Dynamic Aerospace Vehicle Exchange Markup Language (DAVE-ML) Reference

Version 1.5b2

Bruce Jackson, NASA Langley Research Center
<e.b.jackson@nasa.gov>

Revision History

Revision 1.1	2003-07-15	bjax - e.b.jackson@nasa.gov
Initial stab at documentation		
Revision 1.2	2003-08-15	bjax - e.b.jackson@nasa.gov
Fixed typos; added fileVersion and extraDocRef elements to fileHeader and modificationRecord, respectively; added email attribute to author element; added mandatory varID to independentVarPts and dependentVarPts.		

A short reference to DAVE-ML syntax and markup.

Table of Contents

Changes since DAVEfunc.dtd 1.5b

- Fixed typos (thanks, Bill)!
- Added `fileVersion` element to `fileHeader` element, so each version of a particular DAVEfunc model can be uniquely identified. Format of the version identifier is undefined.
- Added an email attribute to the `author` element. The eXtensible Name Service (xns [http://www.xns.org/pages/xns_ov.html]) standard doesn't appear to be catching on as rapidly as hoped, so a static e-mail link will have to for now.

- Added a mandatory varID attribute to both `independentVarPts` and `dependentVarPts` so these can be associated with an input and output signal name (`variableDef`), respectively.
- Added an optional `extraDocRef` element to the `modificationRecord` element so more than one document can be associated with each modification event; if only one document needs to be referenced, use of the optional refID in the `modificationRecord` itself will suffice.

Introduction

This document describes the format for DAVE-ML model definition files. DAVE-ML is a proposed standard method for the interchange of aerospace vehicle flight dynamic models. The intent of DAVE-ML is to significantly expedite the process of "rehosting" a simulation model from one facility to another, as well as an improved method to promulgate changes to a particular model to various facilities.

DAVE-ML is based on the eXtensible Markup Language (XML), a World-Wide Web Consortium (W3C) standard. More information on XML is available here.

Many benefits may be derived from application of XML in general, and DAVE-ML in particular, to the exchange of aerospace vehicle data:

- Human-readable, UNICODE text representation of the model
- Unambiguous machine-readable model description, suitable for conversion into programming language or direct import into object-oriented data structures
- The same source file can be used for computer-aided design and real-time piloted simulation
- Based on open, non-proprietary, standards that are language- and facility-independent
- Statistical properties, such as confidence bounds and uncertainty ranges, can be embedded, suitable for Monte Carlo or other statistical analysis of the model
- Compliant with AIAA draft simulation data standards
- Self-contained, complete, archivable data package, including references to reports, wind-tunnel tests, author contact information, data provenance
- Self-documenting and easily convertible to on-line and hardcopy documetation

A more complete discussion on the benefits and design of DAVE-ML can be found at the DAVE-ML web site: *http://dcb.larc.nasa.gov/utils/fltsim/DAVE* [http://dcb.larc.nasa.gov/utils/fltsim/DAVE/index.html]

Purpose

DAVE-ML is intended to convey an entire flight vehicle dynamic simulation package, as is traditionally done with engineering development and flight training simulations. It is intended to allow a programming language independent representation of the aerodynamic, mass/inertia, landing gear, propulsion, and guidance, navigation and control laws for a particular vehicle.

Traditionally, flight simulation data packages are often a combination of paper documents and data files on magnetic or optical media. This collection of information is very much site-specific, and is often incomplete. Many times, the preparing facility makes assumptions about the knowledge the receiving facility has about the way the preparer's simulation environment is structured; these assumptions are not always true. As a result, the "rehosting" of the dy-

namic flight model can take weeks if not months as the receiving facility staff gets their hands around the contents and arrangement of the data package, the model structure, the various data formats, and then spends additional time running check cases (if they are lucky enough to have received any) and tracking down small differences in implementations.

We would all obviously benefit if this tedious, manual process could be somewhat automated. Often, when a paired set of facilities has exchanged one model, the receipt of another model is much faster, since the receiving facility will probably have built some computer scripts and processes to convert the data (both model and checkcase data).

The purpose of DAVE-ML is to define a common exchange format for this data. The advantage gained is that any simulation facility or laboratory would, after having written a DAVE-ML import and/or export script, could automatically receive and/or generate such packages (and updates to those packages) extremely quickly from other DAVE-ML-compliant facilities.

To accomplish this (lofty) goal, the DAVE-ML project is starting with the bulkiest part of the most aircraft simulation packages: the aerodynamic model. This initial version of DAVE-ML can transport a complete aerodynamics model, including descriptions of the aerodynamic build-up equations and the data tables, as well as include references to the documentation about the aerodynamic model. This format also lends itself to any static subsystem model (i.e. one that contains no state vector) such as the mass & inertia model, or a weapons loadout model, or perhaps a navigational database. The only requirement is that model outputs can be unambigously defined in terms of inputs, with no past history information required.

Background

The idea of a universally understood flight dynamics data package has been discussed for at least two decades, within the American Institute of Aeronautics and Astronautics (AIAA) technical committees. There have been proposals in the past to standarize on FORTRAN as well as proprietary, vendor-specified modeling packages (including graphical ones). The National AeroSpace Plane (NASP) program, under the guidance of Larry Schilling of NASA Dryden, came up with a combination Web- and secure FTP-based system for exchanging NASP subsystem models, as well as a naming convention for variables, file names, and other simulation components. Some simulation standards have been proposed by the AIAA and are under active consideration at this writing.

Existing standards

The AIAA has published a Recommended Practice concerning sign conventions, axes systems, and symbolic notation for flight vehicle models [AIAA92].

The AIAA Modeling & Simulation Technical Committee has prepared a draft standard for the exchange of simulation modeling data. This included a methodology for accomplishing the gradual standardization of simulation model components, a mechanism for standardizing variable names within math models, and proposed HDF as the data format. [AIAA01], [AIAA03]

DAVE-ML proposal

In a 2002 AIAA paper, Jackson and Hildreth proposed using XML to exchange flight dynamic models [Jackson02]. This paper gave outlines for how such a standard could be accomplished, and provided a business justification for persuing such a goal.

This proposal includes several key aspects from the draft standard, including allowing use of the AIAA variable name convention, data table schema, and including tracability for each data point back to a referenced document or change order.

Supporting technologies

DAVE-ML relies on MathML, version 2.0, as a means to describe mathematical relationships. MathML is a low-

level specification for describing mathematics as a basis for machine to machine communication. It is used in DAVE-ML to describe relationships between variables and function tables and may also be used for providing high-quality typeset documentation from the DAVE-ML source files. More information is available at the MathML home web page, found at http://www.w3.org/Math/

Major Elements

At present, only one major element of DAVE-ML has been defined: the function definition element, or `DAVEfunc`. `DAVEfunc` is used to describe static models such as aerodynamic and inertia/mass models, where an internal state is not included.

Other major elements are envisioned to describe dynamic portions of the vehicle model (such as propulsion, alighting gear, control systems, etc.) and check case data. Ultimately DAVE-ML should be capable of describing a complete flight dynamics model with sufficient data to validate the proper implementation thereof.

The `DAVEfunc` major element

The `DAVEfunc` element contains both data tables and equations for a particular vehicle subsystem model, for example, the aerodynamic model or the mass/inertia model. A `DAVEfunc` element is broken into roughly five components: a file header, variable definitions, breakpoint definitions, table definitions, and function definitions. This decomposition reflects common practice in engineering development flight simulation models in which the aerodynamic database is usually captured in multidimensional, linearly interpolated function tables. The input to these tables are usually state variables of the simulation (such as Mach number or angle-of-attack). The outputs from these interpolated tables are combined to represent forces and moments acting on the vehicle due to aerodynamics.

It is possible, using DAVEfunc and MathML elements, to completely define an aerodynamic model without use of function tables (by mathematical combinations of input variables, such as a polynomial model) but this is not yet common in the American flight simulation industry.

A file header element `fileHeader` is included to give background and reference data for the represented model.

Variables, or signals, which are used to route inputs, calculations and outputs through the subsystem model are defined with a `variableDef` element. Variables can be thought of as parameters in a computer program, or signal paths on a block diagram. They can be inputs to the subsystem model, constant values, outputs of the model, and/or the results of intermediate calculations. Variables must be defined for each input and output for any function elements as well as any input or output of the subsystem represented. MathML [http://www.w3.org/Math] markup is used to define constant, intermediate, or output variables in mathematical combination of constant values, function table outputs, and other variables. MathML can also be used to define the symbol to use in documentation for each defined variable. Variables also represent the current value of a function (the "dependent variable" in a function definition) so the output of functions can also be used as inputs to other variables or functions.

Breakpoint definitions, captured in `breakpointDef` elements, consist of a list of montomically-increasing floating-point values separated by commas. These sets are later referenced by "gridded" function table definitions and may be referenced by more than one function definition.

Function table definitions, described by `griddedTableDef` and `ungriddedTableDef` elements, generally contain the bulk of data points in an aero model, and typically represent a (usually) smooth hypersurface representing the value of some aerodynamic non-dimensional coefficient as a function of one or more vehicle states (typically Mach number, angle of attack, control surface deflection, and/or angular body rates). These function tables can be either "gridded," meaning the function has a value at every intersection of each dimension's breakpoint, or "ungridded," meaning each data point has a specified coordinate location in n-space. The same table can be used in several functions, such as a left- and right-aileron moment contribution.

Finally, function definitions (described by `function` elements) connect breakpoint sets and data tables to define how an output signal (or dependent variable) should vary with one or more input signals (or independent variables). The valid ranges of input signal magnitudes, along with extrapolation requirements for out-of-range inputs, can be defined. There is no limit to the number of independent variables, or function dimensionality, of the function.

A simpler version of a `function` is available in which the dependent variable breakpoint values and dependent output values are specified directly inside the `function` body. This may be preferred for models that do not reuse function or breakpoint data.

A third form of `function` is to give the gridded table values or ungridded table values inside the `function` body, but refer to externally defined breakpoint sets. This allows reusability of the breakpoint sets by other functions, but keeps the table data private.

Schematic overview of `DAVEfunc`

Shown below are schematic overviews of the various elements currently available in `DAVEfunc`. Each element is described in detail in the appendix. The following key is used to describe the elements and associated attributes.

```
Key:
        elementname : mandatory_attributes, [optional_attributes]
            mandatory_single_subelement
            optional_single_subelement?
            zero_or_more_subelements*
            one_or_more_subelements+
            (character data) implies Unicode text information
```

The header element, `fileHeader`

The `fileHeader` element contains information about the source of the data contained within the `DAVEfunc` major element, including the author, creation date, description, reference information, and a modification history.

```
    fileHeader :
        author : name, org, [xns, email]
            address? :
                (address character data)
        fileCreationDate : date
        fileVersion? :
            (version identification character data)
        description? :
            (description character data)
        reference* : refID, author, title, date, [accession, href]
        modificationRecord* : modID, [refID]
            author : name, org, [xns, email]
                address? :
                    (address character data)
            description? :
                (descriptive character data)
            extraDocRef? : refID
```

`fileHeader` sub-elements:

`author`	Name, organization, and optional XNS ID and mailing address of the author
`fileCreationDate`	Creation date of this file. See the "Additional DAVE-ML conventions" section later in this document for the recommended format.
`fileVersion`	A string that indicates the version of the document. No convention is reached, but this

	could include an automated revision number from configuration control processes.
`description`	Optional but recommended text description: what does this DAVE-ML file represent?
`reference`	A list of zero or more references with a document-unique ID (must begin with alpha character), author, title, date, and optional accession and URL of the reference.
`modificationRecord`	An optional list of modifications with optional reference pointers, as well as author information and descriptions for each modification record. These modifications are referred to by individual function tables and/or data points, using the AIAA modification letter convention. If more than one document is associated with the modification, multiple subelement extraDocRefs may be used in place of the `modificationRecord` 's refID attribute.

The variable definition element, `variableDef`

The `variableDef` element contains a definition of a constant, parameter, signal, or variable used within or generated by the defined subsystem model. It contains attributes with the variable name, an XML-unique `varID` identifier, the units of measure of the variable, and optional axis system, sign convention, alias, and symbol information. Optional sub-elements include a description and a mathematical description, in MathML 2 markup, of the calculations needed to derive the variable from other variables or function table outputs. A final optional subelement, `isOutput`, serves to indicate an intermediate calculation that should be brought out to the rest of the simulation.

There must be a single `variableDef` for each and every input, output or intermediate signal within the DAVE-func model.

```
variableDef+ : name, varID, units, [axisSystem, sign, alias, symbol]
    description? :
        (description character data)
    calculation? :
        math (defined in mathML2.0 DTD) :
    isOutput? :
```

`variableDef` attributes:

`name`	A UNICODE name for the table (may be same as `gtID`.
`varID`	An XML-legal name that is unique within the file
`units`	The units-of-measure for the signal.
`axsiSystem`	An optional brief indicator of the axis system (body, inertial, etc.) in which the signal is measured
`sign`	An optional brief indicator of which direction is considered positive (+RWD, +UP, etc.)
`symbol`	A UNICODE Greek symbol for the signal [to be superceded with more formal MathML or Tex element in a later release]
`varID`	An XML-legal name that is unique within the file

`variableDef` sub-elements:

`description`	An optional text description of the variable
`calculation`	An optional container for the the MathML markup that describes how this variable is calculated from other variables or function table outputs. This element contains a single `math` element which is defined in the MathML 2 markup language [http://www.w3.org/Math].
`isOutput`	This optional element, if present, identifies this variable needs to be passed as an output. How this is accomplished is up to the implementer. Unless specified by this element, a variable is considered an output only if it is the result of a calculation or function AND is not used elsewhere in this DAVEfunc model.

The breakpoint set definition element, `breakpointDef`

The breakpoint set definition element, `breakpointDef`, is used to define a list of comma-separate values that define the coordinate values along one axis of a gridded linear function value table. It contains a mandatory `bpID`, a file-unique XML identifier attribute, an optional name and units of measure attributes, an optional text description element and the comma-separated list of floating-point values in the bpVals element. This list must be monotonically increasing in value.

```
breakpointDef* : bpID, [name, units]
    description? :
    bpVals :
        (character data of comma-separated breakpoints)
```

`breakpointDef` attributes:

`bpID`	An XML-legal name that is unique within the file
`name`	A UNICODE name for the set (may be same as `gtID`.
`units`	The units-of-measure for the breakpoint values.

`breakpointDef` sub-elements:

`description`	An optional description of the breakpoint set.
`bpVals`	A comma-separated, monotonically-increasing list of floating-point values.

The gridded table definition element, `griddedTableDef`

The `griddedTableDef` element defines a multi-dimensional table of values corresponding with the value of an arbitrary function at the intersection of a set of specified independent inputs. The coordinates along each dimension are defined in separate breakpointDef elements that are referenced within this element by bpRefs, one for each dimension.

The data contained within the dataTable definition are a comma-separated set of floating-point values. This list of values represents a multidimensional array whose size is inferred from the length of each breakpoint vector. For example, a two-dimensional table that is a function of an eight-element Mach breakpoint set and a ten-element angle-

of-attack breakpoint set is expected to contain 80 comma-separated values.

By convention, the breakpointRefs are listed in order such that the last breakpoint set varies most rapidly in the associated data table listing.

An optional confidenceBound scalar value may be provided that reprents the confidence level in the values presented. [This statistical capability needs significant improvement in later DAVEfunc releases.]

```
griddedTableDef* : gtID, [name, units]
    description? :
        (description character data)
    provenance? :
        author : name, org, [xns, email]
            address? :
                (address character data)
        functionCreationDate :
            (date in YYYY-MM-DD format, character data)
        documentRef* : docID
        modificationRef* : modID
    breakpointRefs :
      bpRef+ : bpID
    confidenceBound? : value
    dataTable
      (character data)
```

`griddedTableDef` attributes:

`gtID` An XML-legal name that is unique within the file

`name` A UNICODE name for the table (may be same as `gtID`.

`units` The units-of-measure for the table's output signal.

`griddedTableDef` sub-elements:

`description` The optional description element allows the author to describe the data contained within this `griddedTable`.

`provenance` The optional provenance element allows the author to describe the source and history of the data within this `griddedTable`.

`breakpointRefs` The mandatory breakpointRefs element contains separate bpRef elements, each pointing to a separately-defined breakpointDef. Thus, the independent coordinates associated with this function table are defined elsewhere and only a reference is given here. The order of appearance of the bpRefs is important.

`confidenceBound` The numeric value specified in this element represents the statistical confidence in the values contained within this table.

`dataTable` The numeric values of the function at the function vertices specified by the breakpoint sets are contained within this element, in a single comma-separated list. Parsing this list and storing it in the appropriate array representation is up to the implementor. By convention, the last breakpoint value increases most rapidly.

The ungridded table definition element, `ungriddedTableDef`

The `ungriddedTableDef` element defines a set of non-orthogonal data points, along with their independent values (coordinates), corresponding with the dependent value of an arbitrary function.

An optional confidenceBound scalar value may be provided that reprents the confidence level in the values presented. [This statistical capability needs significant improvement in later DAVEfunc releases.]

```
ungriddedTableDef* : utID, [name, units]
    description? :
        (description character data)
    provenance? :
        author : name, org, [xns, email]
            address? :
                (address character data)
        functionCreationDate :
            (date in YYYY-MM-DD format, character data)
        documentRef* : docID
        modificationRef* : modID
    confidenceBound? : value
    dataTable+ :
```

`ungriddedTableDef` attributes:

`utID` A mandatory XML-legal name that is unique within the file

`name` An optional UNICODE name for the table (may be same as `gtID`.

`units` Optional units-of-measure for the table's output signal.

`ungriddedTableDef` sub-elements:

`description` The optional description element allows the author to describe the data contained within this `ungriddedTable`.

`provenance` The optional provenance element allows the author to describe the source and history of the data within this `ungriddedTable`.

`confidenceBound` The numeric value specified in this element represents the statistical confidence in the values contained within this table.

`dataPoint` One or more sets of coordinate and output numeric values of the function at various locations within it's input space. This element includes one coordinate for each function input variable. Parsing this information into a usable interpolative function is up to the implementor. By convention, the coordinates are listed in the same order that they appear in the using function.

The function definition element, `function`.

The `function` element connects breakpoint sets (for gridded tables), independent variables, and data tables to their respective output variable.

```
function* : name
    description? :
    provenance? :
        author : name, org, [xns, email]
            address?
                (address character data)
        functionCreationDate :
        extraDocRef* : docID
        modificationRef* : modID
    EITHER
    {
      independentVarPts+ : varID, [name, units, sign, extrapolate]
          (input values as character data)
      dependentVarPts : varID, [name, units, sign]
          (output values as character data)
    }
    OR
    {
      independentVarRef+ : varID, [min, max, extrapolate]
      dependentVarRef : varID
      functionDefn : [name]
          CHOICE OF
          {
            CHOICE OF
            {
              griddedTableRef : gtID
            OR
              griddedTable : [name]
                breakpointRefs
                    bpRef+ : bpID
                confidenceBound? : value
                dataTable
                    (gridded data table as character data)
            }
          }
          OR
          {
            CHOICE OF
            {
              ungriddedTableRef : utID
            OR
              ungriddedTable : [name]
                confidenceBound? : value
                    dataPoint+
                    (coordinate/value sets as character data)
            }
          }
    }
```

`function` attributes:

`name` A UNICODE name for the function.

`function` sub-elements:

`description`	The optional description element allows the author to describe the data contained within this `function`.
`provenance`	The optional provenance element allows the author to describe the source and history of the data within this `function`.
`independentVarPts`	If the author chooses, he|she can express a linearly-interpolated functions by specifing the indendent (breakpoint) values sets as one or more `independentVarPts` which are comma-separated, monotonically increasing coordinate values corresponding to the `dependentVarpts` given next. In the case of multiple dimensions, more than one `indepdendentVarPts` must be specified, one for each dimension. The mandatory varID attribute is used to connect each independentVarPts with an input signal.
`dependentVarPts`	This element goes along with the previous element to specify a function table. Only one `dependentVarPts` may be specified. If the function is multidimensional, the convention is the last breakpoint dimension changes most rapidly in this comma-separated list of output values. The mandatory varID attribute is used to connect this table output to an output signal.
`independentVarRef`	One or more of these elements refer to a separately-defined `variableDef`. For multidimensional tables, the order of specification is important and match the order in which breakpoints are specified or the order of coordinates in ungridded table coordinate/value sets.
`dependentVarRef`	One `dependentVarRef` must be specified to marry the output of the function to a particular `variableDef`.
`functionDefn`	This mandatory element (if not a simple function) identifies either a separately-specified data table or specifies a private table, either gridded or ungridded.
`griddedTableRef`	If not defining a simple function table, the author may use this element to point to a separately-specified `griddedTableDef` element.
`griddedTable`	As an alternative to reutilization of a previously defined table, this element may be used to define a private output gridded table. See the writeup on `griddedTableDef` for more information.
`ungriddedTableRef`	If not using a simple function table, the author may use this element to point to separately-specified `ungriddedTableDef` element.
`ungriddedTable`	As an alternative to reuse of a previously defined table, this element may be used to define a private output ungridded table. See the writeup on `ungriddedTableDef` for more information.

Additional DAVE-ML conventions

To facilitate the interpretation of DAVE-ML packages, the following conventions are proposed. Failure to follow any of these should be noted prominently in the data files and any cover documentation.

Locus of action of moments

It is recommended that all force and moments be considered to act around a defined reference point, given in aircraft coordinates. It is further recommended that all subsystem models (aerodynamic, propulsive, alighting gear) provide total forces & moments about this reference point and leave the transfer of moments to the center of mass to the equations of motion.

Decomposition of flight dynamic subsystems

It is recommended that a vehicle's flight dynamic reactions be modeled, at least at the highest level, as aerodynamic, propulsive, and landing/arresting/launch gear models. This is common practice in most aircraft simulation environments we've seen.

Date format in DAVE-ML

The recommended way of representing dates in DAVE-ML documentation, especially date attribute and creation date elements, is numerically in the order yyyy-mm-dd. Thus, July 15, 2003 is given as 2003-07-15. This is suggested since such dates sort with minimal effort, and is a common format on UNIX computer systems.

Common sign convention notation

The following list of sign convention notation is recommended for adoption. Note the sign convention for most quantities is already fixed by the AIAA Recommended Practice [AIAA92], so this is actually a list of abbreviations for typical sign conventions:

Common DAVE-ML sign convention notation

Acronym: +AFT
Meaning: Positive aft
Acronym: +ANR
Meaning: Positive aircraft nose right
Acronym: +ANU
Meaning: Positive aircraft nose up
Acronym: +CWFN
Meaning: Positive clockwise from north
Acronym: +DN
Meaning: Positive down
Acronym: +E
Meaning: Positive eastward
Acronym: +FWD
Meaning: Positive forward
Acronym: +LFT
Meaning: Positive left
Acronym: +N
Meaning: Positive northward
Acronym: +OUT
Meaning: Positive outward
Acronym: +POS
Meaning: Always positive
Acronym: +RCL
Meaning: Positive right of centerline
Acronym: +RT
Meaning: Positive right
Acronym: +RWD
Meaning: Positive right wing down
Acronym: +TED
Meaning: Positive trailing edge down
Acronym: +TEL
Meaning: Positive trailing edge left
Acronym: +THR
Meaning: Positive beyond threshold
Acronym: +UP
Meaning: Positive up

Lots more to identify

[FIXME: more conventions are lurking]

...like how to define units-of-measure notation

Planned major elements

Additional major elements will have to be defined to support the goal of rapid exchange of simulation models, including

- Validation check case definitions & data files
- Dynamic elements

Further information

Further information, background, the latest `DAVEfunc.dtd` and example models of some aircraft data packages can be found at the DAVE-ML web site: http://dcb.larc.nasa.gov/utils/fltsim/DAVE/index.html

References

[Jackson02] : Jackson, E. Bruce; and Hildreth,Bruce L.: *Flight Dynamic Model Exchange using XML [http://techreports.larc.nasa.gov/ltrs/PDF/2002/aiaa/NASA-aiaa-2002-4482.pdf]* . AIAA 2002-4482, presented at the AIAA Modeling and Simulation Technology Conference, 5 August 2002, Monterey, California.

[AIAA92] : American Institute of Aeronautics and Astronautics: *American National Standard: Recommended Practice for Atmospheric and Space Flight Vehicle Coordinate Systems*. ANSI/AIAA R-004-1992

[AIAA01] : AIAA Flight Simulation Technical Committee: " Standard Simulation Variable Names [http://dcb.larc.nasa.gov/utils/fltsim/DAVE/SimParNames_Dec2001.pdf] ", Preliminary Draft, December 2001

[AIAA03] : AIAA Modeling and Simulation Technical Committee: " Standards for the Exchange of Simulation Modeling Data [http://dcb.larc.nasa.gov/utils/fltsim/DAVE/SimDataExchange_Jan2003.pdf] ", Preliminary Draft, Jan 2003

A. Element references and descriptions

A description of each element of DAVE-ML is given below.

Element list

`address` - Street address or other contact information of an author
`author` - Gives name and contact information for originating party
`bpRef` - Reference to a breakpoint list
`bpVals` - String of comma-separated values of breakpoints
`breakpointDef` - Defines breakpoint sets to be used in model
`breakpointRefs` - Reference to a breakpoint definition
`calculation` - Used to delimit a MathML v2 calculation

`confidenceBound` - Defines the confidence in a function
`dataPoint` - Defines each point of an ungridded table
`dataTable` - Gives a name to a table of function data
`DAVEfunc` - Root level element
`dependentVarPts` - Defines output breakpoint values
`dependentVarRef` - Identifies the signal to be associated with the output of a function
`description` - Verbal description of an entity
`documentRef` - Reference to an external document
`extraDocRef` - Allows multiple documents to be associated with a single modification event
`fileCreationDate` - Gives date of creation of entity
`fileHeader` - States source and purpose of file
`fileVersion` - Indicates the version of the document
`function` - Defines a function by combining independent variables, breakpoints, and tables.
`functionCreationDate` - Date of creation of a function table
`functionDefn` - Defines a function by associating a table with other information
`griddedTable` - Definition of a gridded table; associates breakpoint data with table data.
`griddedTableDef` - Defines an orthogonally-gridded table of data points
`griddedTableRef` - Reference to a gridded table definition
`independentVarPts` - Simple definition of independent breakpoints
`independentVarRef` - References a predefined signal as an input to a function
`isOutput` - Flag to identify non-obvious output signals from model
`modificationRecord` - To associate a reference single letter with a modification event
`modificationRef` - Reference to associated modification information
`provenance` - Describes origin or history of data
`reference` - Describes an external document
`ungriddedTable` - Definition of an ungridded set of function data
`ungriddedTableDef` - Defines a table of data, each with independent coordinates
`ungriddedTableRef` - Reference to an ungridded table
`variableDef` - Defines signals used in DAVE-ML model

Name

address -- Street address or other contact information of an author

`address`

Content model

```
address :
     (#PCDATA)
```

Attributes

NONE

Possible parents

`author`

Allowable children

NONE

Name

author -- Gives name and contact information for originating party

`author`

Content model

```
author :  name,  org,  [xns],  [email]
     address?
```

Attributes

`name` - the name of the author
`org` - the author's organization
`xns` (optional) - the eXtensible Name Service identifier for the author
`email` (optional) - the e-mail address for the primary author

Description

author includes alternate means of identifying author using XNS or normal e-mail/address

Possible parents

```
fileHeader
modificationRecord
provenance
```

Allowable children

`address`

Name

bpRef -- Reference to a breakpoint list

`bpRef`

Content model

```
bpRef :  bpID
     EMPTY
```

Attributes

`bpID` - the internal XML identifier for a breakpoint set definition

Description

The bpRef element provides references to breakpoint lists so breakpoints can be defined separately from, and reused by, several data tables.

Possible parents

`breakpointRefs`

Allowable children

NONE

Name

bpVals -- String of comma-separated values of breakpoints

`bpVals`

Content model

```
bpVals :
     (#PCDATA)
```

Attributes

NONE

Description

bpVals is a set of breakpoints; that is, a set of independent variable values associated with one dimension of a gridded table of data. An example would be the Mach or angle-of-attack values that define the coordinates of each data point in a two-dimensional coefficient value table.

Possible parents

`breakpointDef`

Allowable children

NONE

Name

breakpointDef -- Defines breakpoint sets to be used in model

`breakpointDef`

Content model

```
breakpointDef :  [name],  bpID,  [units]
     (description?, bpVals)
```

Attributes

`name` (optional) - the name of the breakpoint set
`bpID` - the internal, document-unique XMLname for the breakpoint set
`units` (optional) - the units of measure for the breakpoint set

Description

A breakpointDef is where gridded table breakpoints are given. Since these are separate from function data, may be reused.

Possible parents

`DAVEfunc`

Allowable children

`description`
`bpVals`

Name

breakpointRefs -- Reference to a breakpoint definition

`breakpointRefs`

Content model

```
breakpointRefs :
     bpRef+
```

Attributes

NONE

Description

The breakpointRefs elements tie the independent variable names for the function to specific breakpoint values defined earlier.

Possible parents

`griddedTableDef`
`griddedTable`

Allowable children

`bpRef`

Name

calculation -- Used to delimit a MathML v2 calculation

`calculation`

Content model

```
calculation :
     math
```

Attributes

NONE

Description

Optional calculation element is MathML 2 content markup describing how the signal is calculated.

Possible parents

`variableDef`

Allowable children

`math`

Name

confidenceBound -- Defines the confidence in a function

`confidenceBound`

Content model

```
confidenceBound :  value
     EMPTY
```

Attributes

`value` - percent confidence (like 95%) in the function

Description

The confidenceBound element is used to declare the confidence interval associated with the data table. This is a placeholder.

Possible parents

`griddedTableDef`
`ungriddedTableDef`
`griddedTable`
`ungriddedTable`

Allowable children

NONE

Name

dataPoint -- Defines each point of an ungridded table

`dataPoint`

Content model

```
dataPoint :  [modID]
     (#PCDATA)
```

Attributes

`modID` (optional) - the internal XML identifier for a modification record

Description

The dataPoint element is used by ungridded tables to list the values of independent variables that are associated with each value of dependent variable. For example: <dataPoint> 0.1, -4.0, 0.2 <!- Mach, alpha, CL -> </dataPoint> <dataPoint> 0.1, 0.0, 0.6 <!- Mach, alpha CL -> </dataPoint> Each data point may have associated with it a modification tag to document the genesis of that particular point. No requirement on ordering of independent variables is implied. Since this is a ungridded table, the intepreting application is required to handle what may be unsorted data.

Possible parents

`ungriddedTableDef`
`ungriddedTable`

Allowable children

NONE

Name

dataTable -- Gives a name to a table of function data

`dataTable`

Content model

```
dataTable :
     (#PCDATA)
```

Attributes

NONE

Description

The dataTable element is used by gridded tables where the indep. variable values are implied by breakpoint sets. Thus, the data embedded between the dataTable element tags is expected to be sorted ASCII values of the gridded table, wherein the last independent variable listed in the function header varies most rapidly. Values are comma or whitespace separated values.

Possible parents

`griddedTableDef`
`griddedTable`

Allowable children

NONE

Name

DAVEfunc -- Root level element

`DAVEfunc`

Content model

```
DAVEfunc :
     (fileHeader, variableDef+, breakpointDef*, griddedTableDef*, ungriddedTableDef*, func
```

Attributes

NONE

Description

Root element is DAVEfunc, composed of a file header element followed by 1 or more variable definitions and 0 or more break point definitions, gridded or ungridded table definitions, and function elements.

Possible parents

NONE - ROOT ELEMENT

Allowable children

```
fileHeader
variableDef
breakpointDef
griddedTableDef
ungriddedTableDef
function
```

Name

dependentVarPts -- Defines output breakpoint values

`dependentVarPts`

Content model

```
dependentVarPts :  varID,  [name],  [units],  [sign]
     (#PCDATA)
```

Attributes

`varID` - the XML ide of the output signal this table should drive
`name` (optional) - the name of the function's dependent variable output signal
`units` (optional) - the units of measure for the dependent variable
`sign` (optional) - the sign convention for the dependent variable

Description

A dependentVarPts element is a simple of function values and contains a mandatory varID as well as optional name, units, and sign convention attributes. Data points are arranged as single vector with last-specified breakpoint values changing most frequently. Note that the number of dependent values must equal the product of the number of independent values for this simple, gridded, realization. This element is used for simple functions that don't share breakpoint or table values with other functions.

Possible parents

`function`

Allowable children

NONE

Name

dependentVarRef -- Identifies the signal to be associated with the output of a function

`dependentVarRef`

Content model

```
dependentVarRef :  varID
     EMPTY
```

Attributes

`varID` - the internal XML identifier for the output signal

Description

A dependentVarRef ties the output of a function to a signal name defined previously in a variable definition.

Possible parents

`function`

Allowable children

NONE

Name

description -- Verbal description of an entity

`description`

Content model

```
description :
     (#PCDATA)
```

Attributes

NONE

Description

optional description is free-form text describing something.

Possible parents

```
fileHeader
variableDef
breakpointDef
griddedTableDef
ungriddedTableDef
function
modificationRecord
```

Allowable children

NONE

Name

documentRef -- Reference to an external document

`documentRef`

Content model

```
documentRef :  docID
     EMPTY
```

Attributes

`docID` - the internal XML identifier for of a reference definition element

Possible parents

`provenance`

Allowable children

NONE

Name

extraDocRef -- Allows multiple documents to be associated with a single modification event

`extraDocRef`

Content model

```
extraDocRef :  refID
     EMPTY
```

Attributes

`refID` - If an extraDocRef is used, the idRef attribute is required.

Description

A single modification event may have more than one documented reference. This element can be used in place of the refID attribute in a modificationRecord to record more than one refIDs, pointing to the referenced document.

Possible parents

`modificationRecord`

Allowable children

NONE

Name

fileCreationDate -- Gives date of creation of entity

`fileCreationDate`

Content model

```
fileCreationDate :  date
     EMPTY
```

Attributes

`date` - The date of the file, in YYYY-MM-DD format

Description

fileCreationDate is simply a string with a date in it.

Possible parents

`fileHeader`

Allowable children

NONE

Name

fileHeader -- States source and purpose of file

`fileHeader`

Content model

```
fileHeader :  [name]
     (author, fileCreationDate, fileVersion?, description?, reference*, modificationRecord
```

Attributes

`name` (optional) - the name of the file

Description

The header element requires an author, a creation date and a version indicator; optional content are description, references and mod records.

Possible parents

`DAVEfunc`

Allowable children

```
author
fileCreationDate
fileVersion
description
reference
modificationRecord
```

Name

fileVersion -- Indicates the version of the document

`fileVersion`

Content model

```
fileVersion :
     (#PCDATA)
```

Attributes

NONE

Description

This is a string describing, in some arbitrary text, the version identifier for this function description.

Possible parents

`fileHeader`

Allowable children

NONE

Name

function -- Defines a function by combining independent variables, breakpoints, and tables.

`function`

Content model

```
function :  name
     (description?, provenance?,
          (
               (independentVarPts+, dependentVarPts)
 |
               (independentVarRef+, dependentVarRef, functionDefn)
)
)
```

Attributes

`name` - the name of this function

Description

Each function has optional description, optional provenance, and either a simple input/output values or references to more complete (possible multiple) input, output, and function data elements.

Possible parents

`DAVEfunc`

Allowable children

```
description
provenance
independentVarPts
dependentVarPts
independentVarRef
dependentVarRef
functionDefn
```

Name

functionCreationDate -- Date of creation of a function table

`functionCreationDate`

Content model

```
functionCreationDate :  date
     EMPTY
```

Attributes

`date` - the creation date of the function, in YYYY-MM-DD format

Possible parents

`provenance`

Allowable children

NONE

Name

functionDefn -- Defines a function by associating a table with other information

`functionDefn`

Content model

```
functionDefn :  [name]
     (griddedTableRef | griddedTable | ungriddedTableRef | ungriddedTable)
```

Attributes

`name` (optional) - the name of this function definition

Description

A functionDefn defines how function is represented in one of two possible ways: gridded (implies breakpoints), or ungridded (with explicit independent values for each point).

Possible parents

`function`

Allowable children

```
griddedTableRef
griddedTable
ungriddedTableRef
ungriddedTable
```

Name

griddedTable -- Definition of a gridded table; associates breakpoint data with table data.

`griddedTable`

Content model

```
griddedTable :  [name]
     (breakpointRefs, confidenceBound?, dataTable)
```

Attributes

`name` (optional) - the name of the gridded table being defined

Possible parents

`functionDefn`

Allowable children

`breakpointRefs`
`confidenceBound`
`dataTable`

Name

griddedTableDef -- Defines an orthogonally-gridded table of data points

`griddedTableDef`

Content model

```
griddedTableDef :  [name],  gtID,  [units]
     (description?, provenance?, breakpointRefs, confidenceBound?, dataTable)
```

Attributes

`name` (optional) - the name of the gridded table
`gtID` - an internal, document-unique XMLname for the table
`units` (optional) - units of measure for the table values

Description

A griddedTableDef contains points arranged in an orthogonal (but multi-dimensional) array, where the independent variables are defined by separate breakpoint vectors. This table definition is specified separately from the actual function declaration and requires an XML identifier attribute so that it may be used by multiple functions.

Possible parents

`DAVEfunc`

Allowable children

```
description
provenance
breakpointRefs
confidenceBound
dataTable
```

Name

griddedTableRef -- Reference to a gridded table definition

`griddedTableRef`

Content model

```
griddedTableRef :  gtID
     EMPTY
```

Attributes

`gtID` - the internal XML identifier of a gridded table definition

Possible parents

`functionDefn`

Allowable children

NONE

Name

independentVarPts -- Simple definition of independent breakpoints

`independentVarPts`

Content model

```
independentVarPts :  varID,  [name],  [units],  [sign],  [extrapolate]
     (#PCDATA)
```

Attributes

`varID` - the XML id of the input signal corresponding to this independent variable
`name` (optional) - the name of the function's independent variable input signal
`units` (optional) - the units of measure for the independent variable
`sign` (optional) - the sign convention for the independent variable
`extrapolate` (optional) - extrapolation flags for IV out-of-bounds

Description

An independentVarPts element is a simple list of breakpoints and contains a mandatory varID identifier as well as optional name, units, and sign convention attributes. An optional extrapolate attribute describes how to extrapolate the output value when the input value exceeds specified values. This element is used for simple functions that don't share breakpoint or table values with other functions.

Possible parents

`function`

Allowable children

NONE

Name

independentVarRef -- References a predefined signal as an input to a function

`independentVarRef`

Content model

```
independentVarRef :  varID,  [min],  [max],  [extrapolate]
     EMPTY
```

Attributes

`varID` - the internal XML identifier for the input signal
`min` (optional) - the allowable lower limit for the input signal
`max` (optional) - the allowable upper limit for the input signal
`extrapolate` (optional) - extrapolation flags for IV out-of-bounds

Description

An independentVarRef more fully describes the input mapping of the function by pointing to a separate breakpoint definition element. This allows common breakpoint values for many tables.

Possible parents

`function`

Allowable children

NONE

Name

isOutput -- Flag to identify non-obvious output signals from model

`isOutput`

Content model

```
isOutput :
     EMPTY
```

Attributes

NONE

Description

Optional isOutput element signals a variable that should be forced to be an output, even if it is used as an input elsewhere. Otherwise, using program should assume a signal defined with no calculation is an input; a signal defined with a calculation but not used elsewhere is an output; and a signal defined as a calculation and used subsequently in the model is an internal signal.

Possible parents

`variableDef`

Allowable children

NONE

Name

modificationRecord -- To associate a reference single letter with a modification event

`modificationRecord`

Content model

```
modificationRecord :  modID,  [refID]
     (author, description?, extraDocRef*)
```

Attributes

`modID` - a single letter used to identify all modified data with this mod
`refID` (optional) - an optional document reference for this modification

Description

A modificationRecord associates a single letter (such as modification "A") with a modification author, address, and any optional external reference documents, in keeping with the AIAA draft standard.

Possible parents

`fileHeader`

Allowable children

`author`
`description`
`extraDocRef`

Name

modificationRef -- Reference to associated modification information

`modificationRef`

Content model

```
modificationRef :  modID
     EMPTY
```

Attributes

`modID` - the internal XML identifier of a modification definition

Possible parents

`provenance`

Allowable children

NONE

Name

provenance -- Describes origin or history of data

`provenance`

Content model

```
provenance :
     (author, functionCreationDate, documentRef*, modificationRef*)
```

Attributes

NONE

Description

optional provenance describes history or source of data and includes author, date, and zero or more references to documents and modification records.

Possible parents

`griddedTableDef`
`ungriddedTableDef`
`function`

Allowable children

`author`
`functionCreationDate`
`documentRef`
`modificationRef`

Name

reference -- Describes an external document

`reference`

Content model

```
reference :  xmlns:xlink,  xlink:type,  refID,  author,  title,  [accession],  date,  [xli
     EMPTY
```

Attributes

`xmlns:xlink`
`xlink:type`
`refID` - an internal, document-unique, XML identifier for this reference definition
`author` - the name of the author of the reference
`title` - the title of the referenced document
`accession` (optional) - the accession number (ISBN or organization report number) of the document
`date` - the date of the document, in YYYY-MM-DD format
`xlink:href` (optional) - an optional URL to an on-line copy of the document

Description

A reference element associates an external document with an ID making use of XLink semantics.

Possible parents

`fileHeader`

Allowable children

NONE

Name

ungriddedTable -- Definition of an ungridded set of function data

`ungriddedTable`

Content model

```
ungriddedTable :  [name]
     (confidenceBound?, dataPoint+)
```

Attributes

`name` (optional) - the name of the ungridded table being defined

Possible parents

`functionDefn`

Allowable children

`confidenceBound`
`dataPoint`

Name

ungriddedTableDef -- Defines a table of data, each with independent coordinates

`ungriddedTableDef`

Content model

```
ungriddedTableDef :  [name],  utID,  [units]
     (description?, provenance?, confidenceBound?, dataPoint+)
```

Attributes

`name` (optional) - the name of the ungridded table
`utID` - an internal, document-unique XML name for the gridded table
`units` (optional) - the units of measure for the table values

Description

An ungriddedTableDef contains points that are not in an orthogonal grid pattern; thus, the independent variable coordinates are specified for each dependent variable value. This table definition is specified separately from the actual function declaration and requires an XML identifier attribute so that it may be used by multiple functions.

Possible parents

`DAVEfunc`

Allowable children

`description`
`provenance`
`confidenceBound`
`dataPoint`

Name

ungriddedTableRef -- Reference to an ungridded table

`ungriddedTableRef`

Content model

```
ungriddedTableRef :  gtID
     EMPTY
```

Attributes

`gtID` - the internal XML identifier of a ungridded table definition

Possible parents

`functionDefn`

Allowable children

NONE

Name

variableDef -- Defines signals used in DAVE-ML model

`variableDef`

Content model

```
variableDef :  name,  varID,  units,  [axisSystem],  [sign],  [alias],  [symbol],  [initia
     (description?, calculation?, isOutput?)
```

Attributes

`name` - the name of the signal being defined
`varID` - an internal, document-unique XML name for the signal
`units` - the units of the signal
`axisSystem` (optional) - the axis in which the signal is measured
`sign` (optional) - the sign convention for the signal, if any
`alias` (optional) - possible alias name (facility specific) for the signal
`symbol` (optional) - UNICODE symbol for the signal
`initialValue` (optional) - an initial and possibly constant numeric value for the signal

Description

variableDef elements provide wiring information - that is, they identify the input and output signals used by these function blocks. They also provide MathML content markup to indicate any calculation required to arrive at the value of the variable, using other variables as inputs. Note the breakpoint values are specified separately since one input signal may be normalized to more than one breakpoint set (for gridded function data).

Possible parents

`DAVEfunc`

Allowable children

```
description
calculation
isOutput
```