Flight Simulation Model Exchange

Appendices

Daniel G. Murri/NESC
Langley Research Center, Hampton, Virginia

E. Bruce Jackson
Langley Research Center, Hampton, Virginia
Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA’s STI. The NASA STI program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Report Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- **TECHNICAL PUBLICATION.** Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.

- **TECHNICAL MEMORANDUM.** Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- **CONTRACTOR REPORT.** Scientific and technical findings by NASA-sponsored contractors and grantees.

- **CONFERENCE PUBLICATION.** Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- **SPECIAL PUBLICATION.** Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- **TECHNICAL TRANSLATION.** English-language translations of foreign scientific and technical material pertinent to NASA’s mission.

Specialized services also include creating custom thesauri, building customized databases, and organizing and publishing research results.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at [http://www.sti.nasa.gov](http://www.sti.nasa.gov)

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA STI Help Desk at 443-757-5803

- Phone the NASA STI Help Desk at 443-757-5802

- Write to:
  NASA STI Help Desk
  NASA Center for AeroSpace Information
  7115 Standard Drive
  Hanover, MD 21076-1320
Flight Simulation Model Exchange

Appendices

Daniel G. Murri/NESC
Langley Research Center, Hampton, Virginia

E. Bruce Jackson
Langley Research Center, Hampton, Virginia
The use of trademarks or names of manufacturers in the report is for accurate reporting and does not constitute an official endorsement, either expressed or implied, of such products or manufacturers by the National Aeronautics and Space Administration.
Flight Simulation Model Exchange
Volume II

NRB Review Date: February 17, 2011
Table of Contents

Volume II: Appendices

Appendix A. NESC Flight Simulation Model Exchange Assessment Report from Johnson Space Center ................................................................. 3
Appendix B. American National Standard: Flight Dynamics Model Exchange Standard (draft BSR/AIAA S-119-201x) ............................................. 66
Appendix C. XML Document Type Definition file for S-119 markup:
   DAVEfunc.dtd....................................................................................... 183
Appendix D. Dynamic Aerospace Vehicle Exchange Markup Language (DAVE-ML) Reference Manual............................................................ 201
Appendix A. NESC Flight Simulation Model Exchange Assessment Report from Johnson Space Center

NESC Flight Simulation Model Exchange Assessment
October 21, 2010

Matthew V. Jessick
L3 Technology

John Penn
L-3 Communications

David Hasan
L-3 Communications

Edwin Z. (Zack) Crues
NASA Johnson Space Center
Table of Contents

Executive Summary ........................................................................................................ 4
Acronyms ........................................................................................................................ 6
1.0 INTRODUCTION ........................................................................................................ 7
2.0 SCOPE ...................................................................................................................... 7
3.0 METHODS ................................................................................................................ 7
4.0 SOFTWARE INTEGRATION ..................................................................................... 8
  4.1 Integration of Two Provided DAVE-ML Interpreters with Trick and JEOD2 ........ 8
     4.1.1 Lessons Learned ......................................................................................... 15
  4.2 Simulation Data Mapping, and Testing ................................................................. 15
  4.3 TDM_Map: Trick to DAVE-ML Variable Name Mapping ..................................... 15
     4.3.1 Lessons Learned ......................................................................................... 17
  4.4 Autocoding DAVE-ML to C via XSLT .................................................................. 19
     4.4.1 Feasibility of Transforming a Legal DAVE-ML Model into Executable “C” Code .................................................. 19
        4.4.2 Design - Mapping DAVE-ML Elements to “C” Code ........................................ 20
        4.4.3 Lessons Learned ....................................................................................... 24
  4.5 HL-20 Auto-Landing Simulation ........................................................................... 33
5.0 EXECUTION TIME STUDY OF DAVE-ML INTERPRETERS ................................. 37
  5.1 Trick Job Timing System ...................................................................................... 37
  5.2 Test Computer and Operating System Specifications .......................................... 37
  5.3 Compiler and Compilation Flags ......................................................................... 38
  5.4 Miscellaneous Settings ....................................................................................... 38
  5.5 Weaknesses of the Test Method .......................................................................... 38
  5.6 Timing Results .................................................................................................... 38
  5.7 Interpretation of the Results ................................................................................ 40
6.0 NON-AERODYNAMICS MODELS IMPLEMENTED IN DAVE-ML ..................... 40
  6.1 HL-20 RCS Algorithm .......................................................................................... 40
     6.1.1 Ad hoc “Macro” Substitutions via XSLT ......................................................... 41
     6.1.2 Test Method .................................................................................................. 45
     6.1.3 Results .......................................................................................................... 45
     6.1.4 Lessons Learned .......................................................................................... 48
  6.2 Pneumatic Tire Data Compression Model ............................................................ 49
     6.2.1 Ad hoc “Macro” Substitutions via Script ....................................................... 51
     6.2.2 Lessons Learned .......................................................................................... 52
7.0 DAVE-ML SPECIFICATION COMMENTS/SUGGESTIONS .......................................................... 53
  7.1 Uncertainty Element ........................................................................................................ 53
  7.2 GriddedTableDef Element ............................................................................................ 55
  7.3 CheckData Element ........................................................................................................ 55
8.0 CONCLUSIONS .................................................................................................................. 56
9.0 FUTURE WORK .................................................................................................................. 56
10.0 BIBLIOGRAPHY ............................................................................................................. 58
APPENDIX A Draft DAVE-ML Uncertainty Reference .............................................................. 59

Table of Figures

Figure 1. Layered Software Structure ................................................................. 9
Figure 2. Summary UML class diagram ............................................................... 11
Figure 3. Generic DAVE-ML model integration with sim ..................................... 12
Figure 4. HIL-20 aerodynamic model integration with sim ................................. 12
Figure 5 DaveMLModel Interface in UML ........................................................... 13
Figure 6. TDM_Map Software UML Class Diagram ............................................ 17
Figure 7. TDM_Map Software UML, Sequence Diagram ...................................... 18
Figure 8. DAVEML Elements and Dependencies that are related to Code Generation 19
Figure 9. UML Diagram For MathML "math" Elements ....................................... 21
Figure 10. HIL-20 Trick Sim, Pitch Pulse, Angle of Attack Trace ...................... 34
Figure 11. TM 107580 Pitch Pulse, Angle of Attack Trace .................................. 34
Figure 12. HIL-20 Auto-Landing Trajectory, Overhead Map ......................... 35
Figure 13. HIL-20 Auto-Landing Trajectory, Angle of Attack (AoA) and Mach Profiles ..... 36
Figure 14. CPU Time per Call, HIL-20 Aero Model ........................................... 39
Figure 15. HIL-20 RCS Algorithm of Powell(18) .................................................. 41
Figure 16. DAVE-ML HIL-20 RCS Algorithm Test Results: Angle of Attack .... 46
Figure 17. DAVE-ML HIL-20 RCS Algorithm Test Results: Roll Angle .......... 47
Figure 18. DAVE-ML HIL-20 RCS Algorithm Test Results: Body Rates ........... 48
Figure 19. Tire Data Compression Model Set ......................................................... 49
Figure 20. Tire Data Compression Model Example Output .................................. 51

Table of Tables

Table 1 Generic Interpreter Interface Methods ......................................................... 14
Table 2, Touchdown Conditions Compared For The Four HIL-20 Aero Model Versions ......................................................... 36
Table 3. CPU Time per Call Summary, HIL-20 Aero Model ................................. 39
Table 4, HIL-20 RCS Thruster Model ...................................................................... 45
Table 5. HIL-20 entry Rigid Body Mass Model ....................................................... 45
Executive Summary

This report summarizes an investigation of the DAVE-ML markup language, performed at Johnson Space Center (JSC) during the 2010 fiscal year. The work focused on several areas: (1) Integration of DAVE-ML software with the JSC Trick simulation framework, (2) Analysis of DAVE-ML interpreter performance, (3) Investigation of some non-aerodynamic DAVE-ML models, and (4) Analysis of the S-119 and DAVE-ML XML draft specifications.

Software Integration. Our effort to integrate DAVE-ML into JSC simulation software involved two activities: integration and code generation. The integration activity focused on integrating the two available DAVE-ML interpreter systems (Janus and LaSRS++) into Trick. The code generation activity involved the development of an XML-to-C/C++ code generator to create compilable code from a DAVE-ML model. For both activities (integration and code generation), we focused primarily on an HL-20 lifting body simulation, incorporating the HL-20 DAVE-ML aerodynamic model with the Trick simulation framework and a JSC dynamics package (JEOD) to provide a planet model, coordinate systems, vehicle dynamics and a vehicle trajectory. We implemented a single "generic" software model that integrated our Trick-based simulations with either the Janus or the LaSRS++ DAVE-ML interpreters. Details of this generic design are provided in the report. We also implemented an XSLT-based XML-to-C/C++ code generator that allows the DAVE-ML model algorithms to be executed directly rather than interpreted in the Janus or LaSRS++ DAVE-ML "virtual machines". We found this generated code useful as a baseline against which to compare the runtime performance of the interpreted approach. It could also be used to compare against a hand coded translation of a DAVE-ML transferred algorithm during development.

Execution Time Study of DAVE-ML Interpreters. Our performance analysis of DAVE-ML models involved the investigation of a Trick-based HL-20 auto-landing simulation integrated with (1) the Janus DAVE-ML interpreter, (2) the LaSRS++ DAVE-ML interpreter, (3) the C code auto-coded from our XML-to-C/C++ code generator, and (4) some pre-existing hand-coded HL-20 source code. In all four cases, the simulations generated the same trajectory. We found that interpreted DAVE-ML was of comparable speed to auto-generated compiled C code and hand tuned code for the limited testing we performed. Detailed results are available in the report, including some of the weaknesses of the method we used for performance analysis.

Non-Aerodynamics Models Implemented in DAVE-ML. In addition to our work with the HL-20 aerodynamics model, we looked at two non-aerodynamic DAVE-ML models: (1) An HL-20 reaction control system (RCS) algorithm, and (2) A pneumatic tire force model. Our investigation of the RCS algorithm, in particular our successful representation of it in DAVE-ML and subsequent execution of the model using the Janus and LaSRS++ interpreters and our XML-to-C/C++ code generator, offers some evidence that models beyond the aerodynamics niche can indeed be represented through the current DAVE-ML specification. In particular, dynamic models with saved states can be created in DAVE-ML 2.0 without direct support in the specification, by the caller providing workspace for the states. This worked reasonably well when aided by a convenient method of hooking together corresponding simulation and internal DAVE-ML interpreter variables. Our investigation of a pneumatic tire data compression model showed that the self-documenting properties of DAVE-ML can be used to record provenance, modification, and accuracy data for static data sets; sophisticated models can be created entirely in MathML without recourse to table look-ups; and that models can be created in a hierarchy where the outputs from one are then fed into the next in DAVE-ML 2.0, even though this feature is not supported directly in the specification. Further details are available in the body of report.

DAVE-ML Specification Comments/Suggestions. In the course of our investigations, we collected some observations about the DAVE-ML specification, in particular the XML DTD. These observations are primarily the result of (1) our code generation work and (2) a detailed look we took into the DAVE-ML uncertainty element. Generally, we found that in a few places the DAVE-ML DTD is insufficiently precise to support automatic code generation (e.g., certain XML element attributes are optional leading to the possibility that a DAVE-ML compliant XML model might not allow code generation without some manual intervention). We also found that the
documentation of the uncertainty element in the Reference Manual could be improved, and we drafted a proposed replacement for the relevant section of the Reference Manual to fix some (but not all) of the weaknesses we found. Our detailed observations are available in the body of the report.

Conclusions and Suggestions for Future work. We suggested several clarifications and changes to the DAVE-ML specification that are described in more detail later in this report, and summarized in the conclusions section. We feel these changes would improve the rigor and clarity of the specification, making it easier to develop interpreters and code generators for DAVE-ML, and helping to transfer models without ambiguity. The feasibility of an XSLT based DAVE-ML to “C” code generation capability was demonstrated during this assessment. Although incomplete, the system prototyped during this project is useful now and shows considerable potential for future expansion. The utility of the DAVE-ML specification for use by non-aerodynamics models was shown by two test cases, that also investigated hierarchical models, a pseudo-dynamic model with saved states implemented via caller provided memory storage, and demonstrated two ways to use “macros” (essentially) to ease MathML authoring for complex algorithms. We suggested several areas where future work would be useful. These include further exploration of the S-119 and DAVE-ML specifications, continued development of the XSLT code generator toward an operational capability, and testing of the DAVE-ML <uncertainty> element by exercising it to specify dispersion test cases for the Trick Monte-Carlo capability.
## Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI</td>
<td>American Institute of Aeronautics and Astronautics</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>DAVE-ML</td>
<td>Dynamic Aerospace Vehicle Exchange Markup Language</td>
</tr>
<tr>
<td>DSTO</td>
<td>Defense Science and Technology Organization (Australian Dept. of Defense)</td>
</tr>
<tr>
<td>DTD</td>
<td>Document Type Definition</td>
</tr>
<tr>
<td>FSME</td>
<td>Flight Simulation Model Exchange program of the NESC</td>
</tr>
<tr>
<td>GNC</td>
<td>Guidance, Navigation and Control</td>
</tr>
<tr>
<td>HAC</td>
<td>Heading Alignment Cylinder (or Circle)</td>
</tr>
<tr>
<td>JEDO2</td>
<td>JSC Engineering Orbital Dynamics model set, version 2</td>
</tr>
<tr>
<td>JSC</td>
<td>NASA Johnson Space Center</td>
</tr>
<tr>
<td>LaRC</td>
<td>NASA Langley Research Center</td>
</tr>
<tr>
<td>LaSRS++</td>
<td>Langley Standard Real-Time Simulation in C++</td>
</tr>
<tr>
<td>MathML</td>
<td>Mathematics Markup Language</td>
</tr>
<tr>
<td>NASA</td>
<td>National Aeronautics and Space Administration</td>
</tr>
<tr>
<td>NESC</td>
<td>NASA Engineering Safety Center</td>
</tr>
<tr>
<td>RCS</td>
<td>Reaction Control System</td>
</tr>
<tr>
<td>Trick</td>
<td>The Trick Simulation Environment</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
<tr>
<td>XSLT</td>
<td>Extensible Stylesheet Language Transformations</td>
</tr>
</tbody>
</table>
1.0 INTRODUCTION

The goals of this project were to support the NASA Engineering Safety Center (NESC) in their efforts to study the American Institute of Aeronautics and Astronautics (AIAA) draft specification S-119 Flight Simulation Model Exchange (FSME). The work at NASA Johnson Space Center (JSC) was part of a multi-center effort to gain experience in integrating Dynamic Aerospace Vehicle Exchange - Markup Language (DAVE-ML) models into the local simulation systems of each center in order to more efficiently transfer simulation models.

2.0 SCOPE

The major tasks were to:
- Review FSME project documents
- Integrate DAVE-ML models into the local (in this case, NASA/JSC) simulation construction flow
- Demonstrate the integration by using the HL20_aero.dml DAVE-ML file as the basis of an HL-20 landing simulation
- Evaluate the S-119(1) and DAVE-ML(2) draft specifications for their purpose
- Provide feedback on the specifications
- Make a recommendation with the other participating centers for the draft specification.

3.0 METHODS

Our basic strategy for integration of DAVE-ML into JSC simulation construction flow was to integrate use of simulation models, expressed in DAVE-ML format XML files, directly into common JSC simulation construction tool sets via example DAVE-ML interpreters. These included the Trick Simulation Environment and the JSC Engineering Orbital Dynamics (JEOD) model set. The two available DAVE-ML interpreter systems were DaveMLTranslator(3) (NASA/LoR) and Janus(4) (DSTO).

As a second method of integration, we investigated the feasibility of generating C source code from DAVE-ML file algorithms via XSLT(5)(6) (Extensible Markup Language Transformations). Given the ubiquity of XSLT tools, if found to be feasible, this seemed like a widely applicable method of directly translating DAVE-ML models, or at least greatly easing the task of converting models from DAVE-ML to C.

An alternative to this strategy would have been to build tools to convert DAVE-ML file models to local (Trick format) input files and code. One interesting lack in the Trick Simulation Environment, however, is a single preferred method of performing a table look up from data. This is typically left to the model builder. Given the preeminence of such structures in DAVE-ML and without such a baseline code module to target converted input files toward, we felt our time during this assessment was better spent pushing the capabilities of the two provided DAVE-ML interpreter systems to their limits, and also to pursue code generation as better fitting into the usual Trick usage patterns. (Trick simulations typically contain a strong element of code generation).

Evaluation of the S-119 and DAVE-ML specifications ended up being concentrated on the DAVE-ML specification, through an extensive review and suggested re-drafting of the "uncertainty" element, and DAVE-ML Document Type Definition (DTD) issues that cropped up during the XSLT based code generation work.
4.0 SOFTWARE INTEGRATION

4.1 Integration of Two Provided DAVE-ML Interpreters with Trick and JEOD2

Two existing DAVE-ML Interpreters provided through the FSME program by the DSTO and NASA/LaRC were integrated into the Trick Simulation Environment (Trick(7)) and the JSC Engineering Orbital Dynamics (JEOD version 2 (8)) model package in object oriented fashion.

These two toolsets are in widespread use at NASA/JSC and elsewhere for real-time simulation of vehicles and robotics. One of the major results of the JSC participation in the FSME program is the development of a framework to integrate DAVE-ML capability into these common simulation development toolsets via the DAVE-ML interpreters or a new code generation capability (described later).

The new DAVE-ML capability was exercised primarily through construction of a Trick/JEOD2 based HL-20 auto-landing simulation as specified in the FSME program. This new simulation featured runtime interpretation of a DAVE-ML aerodynamics model simulating the flight of the HL-20 as a single rigid body from Mach 4 to touchdown. The HL-20 simulation is able to run in real-time or non-real-time modes, unpiolated, and lands under the control of an auto-coded autolanding guidance, navigation and control (GN&C) algorithm provided by NASA/LaRC.

Additional miscellaneous simulations were constructed using the same structure to exercise other DAVE-ML models for various purposes during the study and will be described later.

Figure 1 shows the layered hierarchy of simulation software modules used to construct the HL-20 auto-landing simulation, which was developed during this program. Each layer uses the capabilities provided by the layer below. The simulation is shown divided into four major layers, and data files:

- Simulation Environment
- User Models
- DAVE-ML Interpreters
- 3rd Party Libraries
- plus the Data Files.

These five areas are separated by dashed lines in Figure 1. The solid lines in the figure indicate software module interactions. The arrows show data flow for DAVE-ML file interpretation during runtime and also code generation of MatLab™ models to code.
The Trick Simulation Environment served as the base framework for the simulation, providing simulation executive functions, model organization, and code generation, as well as real-time capability and input, output, monitoring, and plotting facilities.

The JEOD version 2 model package provided the bulk of the simulation models. These included time management, planet models and ephemerides, environment, coordinate systems, vehicle dynamics and trajectory integration. The JEOD general models were customized for this application using the Trick code generation and input systems. JEOD provides the DefaultAero class as a hook for further specialization by the user. DefaultAero was used as the base class from which the JEOD adapter modules were derived to hook DAVE-ML aero models into the simulation. Two software adapter objects were constructed to integrate the two provided DAVE-ML interpreters into the simulation structure in an object oriented fashion for the purposes of this investigation.

The two DAVE-ML interpreters formed the third layer of the simulation:

- Janus(4) provided by the Australian Defence Science and Technology Organization (DSTO), and
- DaveMITranslator(3), part of the LASRS++ simulation, provided by the NASA/Langley Research Center.
The major third party libraries forming the bottom layer of the simulation are also indicated in the diagram. These include:

<table>
<thead>
<tr>
<th>LibXML2(9)</th>
<th>A cross-platform XML library used by DaveMITranslator  (In wide use as the XML provider for the Linux GNOME desktop)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xerces(10)</td>
<td>A cross-platform XML library used by Janus  (In wide use as the XML provider for the Apache web server)</td>
</tr>
<tr>
<td>Qhull(11)</td>
<td>A Convex Hull library used by Janus for ungridded table look ups.  (In wide use by: MatLab, GNU Octave, Mathematica, Google, etc.)</td>
</tr>
</tbody>
</table>

Additionally, MatLab™/Simulink™ Real Time Workshop™, a visual modeling system with code generation capability, was used to view and generate C code for the HL-20 Auto-landing guidance, navigation and controls (GNC) algorithms from Simulink™ files provided by NASA/LaRC.

This structure allows DAVE-ML models to be loaded as XML files into a Trick/JEOD based simulation during initialization, either as JEOD aerodynamic models or as general models, and interpreted during runtime via either of the two supported DAVE-ML interpreters. The interpreter module to be used may be selected for each model during initialization via Trick input processing.

The circled numbers in the diagram indicate the source of existing and provided modules and files per the Legend. The blue colored "Implemented" items were newly constructed during this program. These include the various adapter modules used to integrate DAVE-ML models into Trick and JEOD, the HL-20 reaction control system (RCS) algorithm implemented in DAVE-ML, and the "glue code" required to integrate the autocoded GNC algorithm into the simulation.

A summary UML(12) (Unified Modeling Language) class diagram showing the details of the software integration is shown in Figure 2.
This implementation uses several object oriented software "Design Patterns"(13):

- The "Strategy" design pattern is used to select amongst similar algorithms that all provide a consistent interface to the using software modules via C++ inheritance.
- The "Factory" design pattern is used to simplify construction of objects of the selected type.
- The "Adapter" design pattern is used to adapt the differing interfaces of DaveMIM Translator and Janus into a single interface capable of operating both embedded interpreters at the minimal level required of the simulation.
- The DaveMIM Translator base class uses the "Template Method" pattern to access the underlying interpreter APIs in a generic way.

Current DAVE-ML version 2.0 models can be summarized as multi-input multi-output functions with no internal states. To use the models, the values of simulation input variables (e.g., angle of attack) must be passed to the corresponding variables within the DAVE-ML model. The values of the output variables of the DAVE-ML model...
must then be passed back to corresponding output variables owned by the simulation. This basic structure is shown in Figure 3, and a specific example for the HL-20 aero model in Figure 4.

![Diagram of DAVE-ML model integration with sim](image)

**Figure 3. Generic DAVE-ML model integration with sim**

![Diagram of HL-20 aerodynamic model integration with sim](image)

**Figure 4. HL-20 aerodynamic model integration with sim.**

Two capabilities of the implemented software are potentially of general interest. These include the:

- **TDM** Map class, providing a way to read an XML file specifying a mapping between the names of sim variables and corresponding DAVE-ML model variable names.

- **DataMapper** class and sub-classes. This system provides a general interface to obtain a pointer to a simulation variable. When integrating with Trick, the TrickDataMapper subclass uses the Trick API to provide this service. The TestDataMapper subclass simply allows unit test code to enter `name,pointer` pairs to hook the model up to test code without requiring the Trick systems.

Both of these systems could be used or extended for use in other simulation environments. These systems are discussed in individual sections further below.

The DaveMModel class serves to adapt the differing application programming interfaces (APIs) of DaveMTranslator and Janus to a single generalized interface. (Note: In doing so, the additional flexibility of Janus allowing calculation of less than the full set of outputs was suppressed.)

Figure 5 shows the DaveMModel interface in UML. Methods shown with a “*” are public, those with “#” are private, and those in italics are pure abstract methods that are required to be implemented by all subclasses.
Once during initialization processing, the static member function factoryBuild is called to create the selected type of DAVE-ML interpreter – DaveMLTranslator or Janus – and to specify the file names of the DAVE-ML file defining the model and the TDM_Map XML file specifying the correspondence between simulation and internal DAVE-ML model variable names. The init() member function is then called once. (Optionally, for unit testing the overloaded init version that allows injecting the DataMapper object may be called). The init method then uses the names provided by the TDM_Map system to set up a mapping between simulation variable pointers (provided by the DataMapper) and the internal DAVE-ML model variable pointers (provided by member function getDaveMLVariablePointerFromName).

During simulation run-time, the DaveMLModel::update method is called repeatedly to perform the following steps in turn:

a) Copy values from the input simulation variables to the corresponding internal DAVE-ML model variables, using the pointers set up during initialization.

b) Update the internal DAVE-ML Model.

For the DaveMLTranslator API, this is calling the update() method. (The bulk of the DaveMLTranslator processing occurs here. This interpreter recalculates all of the output variables values using the values of the input variables in this one update call.)

For the Janus API, this function is empty (no action - see below).

c) Copy values from the internal DAVE-ML model variables to the corresponding simulation output variable pointers as setup during initialization.

For the DaveMLTranslator API, this is a data copy between pointers.

For the Janus API, this step is where the bulk of its processing takes place. The value of each output variable is requested in turn. If Janus determines that intermediate variables needed to determine this value are stale, they are recalculated. The returned result is then copied to the simulation variable pointer (Note that the Janus capability of only calculating some of the outputs is not used in this generalized access strategy.)
The generic interface provided by the DaveMIModel class was determined to be sufficient for access to the minimum features of both the DaveMIMTTranslator and Janus interpreters for general DAVE-ML models. This generic interface is shown in Table 1 below. Each subclass derived from DaveMIModel implements this interface.

The suggested implementation of the update method is shown below the table.

### Table 1 Generic Interpreter Interface Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>void copyInputs()</td>
<td>Copy input values from sim variables to internal DAVE-ML model variables</td>
</tr>
<tr>
<td>void copyOutputs()</td>
<td>Copy output values from internal DAVE-ML model variables to sim variables</td>
</tr>
<tr>
<td>void interpreterLoadAndCheck (const std::string &amp; filePath, bool performCheckCases)</td>
<td>Command interpreter to: load DAVE-ML file perform DAVE-ML check cases (optional)</td>
</tr>
<tr>
<td>double* getDaveMIVariablePointerFromName (const std::string &amp; name)</td>
<td>Return pointer corresponding to internal DAVE-ML variable &quot;name&quot;</td>
</tr>
<tr>
<td>void interpreterInitialize()</td>
<td>Command interpreter to set input variables to default values</td>
</tr>
<tr>
<td>void interpreterUpdate()</td>
<td>Command interpreter to update using the current input variables already set (by copyInputs) and to copyOutputs</td>
</tr>
<tr>
<td>void interpreterTerminate()</td>
<td>Destroy the interpreter</td>
</tr>
<tr>
<td>void update()</td>
<td>&quot;Template Method&quot; design pattern see below for implementation</td>
</tr>
</tbody>
</table>

The "Template Method" DaveMIModel::update() is a generic recipe that all sub-classes implementing different interpreters can use to update their models. It is built up primarily of calls to the other generic interface functions.

In C++, this recipe is:

```cpp
void update() {
  if (!isInterpreter()) {
    // set zeros into all outputs if a working interpreter is not available
    ... (implementation not shown here)
  } else {
    copyInputs();
    // update the model using the input values
    interpreterUpdate();
    copyOutputs();
  }
}
```

Two adapter classes were constructed:

- DaveMIModel is used to wrap a general DAVE-ML model intended to be embedded in user model class source code.
- DaveMIModelObject has a slightly different public interface making it more convenient to place a DAVE-ML model as a child of a top level simulation object of a Trick sim.
4.1.1 Lessons Learned

We found it valuable to have both the Janus and DaveMLTranslator interpreters available. Janus is the more
capable, providing ungridded table capability via the QUIL library. The two interpreters use different
underlying XML libraries, so if parser problems or capability gaps were to be found for one interpreter, the other
can be tried. One may have a faster run time performance for a particular problem than the other.

Execution speed using the interpreters was close to hand coded performance for the models we investigated
(within a factor of 4). Should the interpreted performance prove inadequate for some problem, C source code can
be auto-generated from DAVE-ML files using XSLT with some hand coded finishing work, or the algorithm can
be fully hand coded from the DAVE-ML definition. The DAVE-ML internal checkData available with each model
was found to be particularly valuable while integrating each DAVE-ML model.

4.2 Simulation Data Mapping and Testing

During the course of this project, we developed a general DataMapper interface that was used to encapsulate the
Trick Simulation Environment’s system for returning memory pointers from variable names. The Trick system
was accessed through this general interface so that a test-specific sub-class could be easily substituted for it during
unit testing. This allowed testing of the DAVE-ML models separate from the Trick environment.

This system worked along with the TDM_Map system (see the next section). The DataMapper provided pointers
from simulation variable names, while TDM_Map provided the correspondence between simulation variable
names and internal DAVE-ML model variables. These two capabilities allowed the DaveMLModel class
(discussed previously), to hook up DAVE-ML models to the simulation via an XML file in a general way useful
for all models.

4.3 TDM_Map: Trick to DAVE-ML Variable Name Mapping

A TDM_map class simply stores pairs of variable names. One name in a pair refers to a DAVE-ML model
variable, while the other refers to a Trick model variable. An instance of a TDM-map is constructed from an XML
file like the following, by calling the method TDM_make_map():
The following is an example usage of TDM_make_map(), a factory class of sorts, to create a TDM_map object:

```c
#include "TDM_make_map.hh"

// Load and process the DAVE-ML to Trick mapping file
// and setup the mappings
TDM_map* tdmapi = NULL;

tdmap = TDM_make_map(mapFilePath.c_str());

if (tdmap != NULL) {
    int i;
    int n_inputs, n_outputs;
    const char* dvar;
    const char* tvar;

    n_inputs = tdmapi->NumberOfInputs();
    n_outputs = tdmapi->NumberOfOutputs();

    for (i = 0; i < n_inputs; i++) {
        tdmapi->getInputVarPair( i, dvar, tvar);
        // save the name
    }

    for (i = 0; i < n_outputs; i++) {
        tdmapi->getOutputVarPair( i, dvar, tvar);
        // save the name
    }

    // done with the map
    delete tdmapi;
    tdmapi = NULL;
}
```

Figure 6 below shows the relationships of the components of a TDM_map. Note that it closely matches the structure of the XML file. A TDM component corresponds to an XML element. The TDM_map, TDM_varpairlist, and TDM_varpair each correspond to elements of the XML file.
Figure 6. TDM_Map Software UML Class Diagram

A UML sequence diagram of the interactions is shown in Figure 7.

The purpose of TDM_make_map() is to parse the XML file and pass the root node of the parse tree to the TDM_map constructor so it can walk the tree, creating a in memory representation of the same information.

4.3.1 Lessons Learned

The TDM_Map system is independent of the simulation environment being used. It can provide XML file model hook up capability for any environment, as long as the variable names it correlates can be used by following software like our DataMapper class to find pointers to sim variables given their names, and the DAVE-ML interceptors are wrapped in a class with similar capabilities to DaveMLModel such that the available interpreter(s) can return pointers to internal DAVE-XML model variables from input variable names in a common way.
Figure 7. TDM_Map Software UML Sequence Diagram
4.4 Autocoding DAVE-ML to C via XSLT

4.4.1 Feasibility of Transforming a Legal DAVE-ML Model into Executable "C" Code.

DAVE-ML is an XML markup language for describing vehicle models.

DAVE-ML's grammar rules (syntax), which specify how elements can be legally ordered, are described by its DTD. The semantics of a DAVE-ML model file are conveyed by the definitions of its language-specific elements and attributes (specified in the DAVE-ML Reference(2)) and of course by the user-specific choices of those elements and attributes, ordered according to the grammar rules.

This investigation is an attempt to determine the extent to which the syntax and semantics of DAVE-ML sufficiently and unambiguously describe a model that can be transformed into executable "C" code.

The XML elements defined by DAVE-ML organize a lot of information about a model, including its definition, configuration history, data references, authorship, uncertainty and check data. The elements that we are concerned with in this particular exercise are those related to the definition of the executable model, in other words, those elements that would need to be translated into "C" to properly represent the model. Figure 8 shows these DAVE-ML elements and their dependencies.

Figure 8. DAVEML Elements and Dependencies that are related to Code Generation
Scope Limitations:
Because we have a limited amount of time, we'd like to concentrate on getting the most "bang for buck" with what time we have, therefore:
- Only "gridded" data will be supported.
- Only enough of the MathML(14) to transform the F-16 and HL-20 aero models will be processed.

Choice of XSLT:
Since DAVE-ML is an XML(15) based language, XSLT (Extensible Style-sheet Language for Transformations) is a natural means of transforming it.
- XSLT is widely used for transforming XML.
- It is an official recommendation of the World Wide Web Consortium (W3C).
- There are many freely available XSLT processors.

Working on Linux, we chose xsltproc, a tool in the GNOME XSLT library, because of its common availability.

4.4.2 Design - Mapping DAVE-ML Elements to "C" Code

The following sections map DAVE-ML elements to corresponding "C" code. They refer often to XSLT code in Listing 1 – DMLtoC.xsl by line number.

DAVEfunc Element
We will assume that the DAVEfunc element maps to an executable "C" model, because of our boundless optimism and because otherwise we would be admitting failure before we even got started. The general approach will be:
Generate the following in the following order:

<table>
<thead>
<tr>
<th>Line Number</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>33..34</td>
<td>Necessary include files.</td>
</tr>
<tr>
<td>41..43</td>
<td>Variable Definitions generated from variableDef elements.</td>
</tr>
<tr>
<td>50..52</td>
<td>Breakpoint Definitions generated from breakpointDef elements.</td>
</tr>
<tr>
<td>59..67</td>
<td>Interpolation Data generated from griddedTableDef elements and function elements.</td>
</tr>
<tr>
<td>70..78</td>
<td>In a function called update(), perform the following:</td>
</tr>
<tr>
<td>78..80</td>
<td>Variable Initialization also generated from variableDef elements.</td>
</tr>
<tr>
<td>87..89</td>
<td>Function calls generated from function elements</td>
</tr>
<tr>
<td>96..98</td>
<td>Assignment statements generated from variableDef elements.</td>
</tr>
</tbody>
</table>

(Line numbers refer to those in Listing 1 – DMLtoC.xsl)

variableDef Element
A variableDef element maps to the variable definition in "C". The name of the "C" variable can be taken from the @varID attribute (line 121). The initializer for the "C" variable can be taken from the @initialValue attribute. The type specifier for the "C" variable definition can probably, safely be assumed to be double.

double vt = 1.0;

Rather than creating initializers in the declaration, we initialized the variables with assignment statements (lines 78..80, 135).
calculation Element

A calculation element maps to the right side of an assignment statement, in which the “C” variable associated with the parent variableDef element, is assigned the value resulting from the evaluation of a “C” arithmetic expression, corresponding to the enclosed MathML expression (lines 346, 360).

Example C assignment statement:

cq2v = (cbar + (q / tvt));

The expression within a <math> element is defined recursively. An expression is defined as:

- `<cn>` element (a number) OR
- `<ci>` element (an identifier) OR
- `<apply>` element, consisting of:
  - an operator ( `<plus>` | `<minus>` | `<times>` | ... ) AND
  - an aggregation of expressions.

![UML Diagram For MathML <math> Elements](image)

If the expression is a `<cn>` element, then we need to omit the corresponding number.
If the expression is a `<ci>` element, then we need to omit the corresponding identifier.
If the expression is an `<apply>` element, then we need to first identify the operator and then process the remaining “sibling” operands, based on the operator that we found.

XSLT supports recursive template calls, so the transformations of plus, minus, times and divide (sub)expressions was straightforward. The one problem we ran into was how to transform a piecewise element into an expression suitable for the right hand side of an assignment. Though we haven’t yet had time to figure out how to do it, we still think it might be do-able, possibly by transformation of the piecewise element into a function which is then called on the left hand side of an assignment.

So that we didn’t run afoul of C’s operator precedence rules, we put parentheses around all of the sub-expressions generated from transformations of the `<apply>` element.

Whether or not all possible MathML expressions, that might legally appear in a calculation element, can be translated to a corresponding “C” expression, is still an open question.

function Element

A function element also maps to an assignment statement (line 184). In this assignment statement, the dependent variable, specified by the dependentVarRef element, is assigned the value returned by an interpolator function call. The interpolator function will take as its arguments:

- References to the independent variable(s), specified by the independentVarRef element and
- A reference to the interpolation data, associated with the functionDefn element.
dependent_var = interpolation_fn(interpolation_data_ref,
    independent_var_1,
    , independent_var_n);

**IndependentVarRef Element**

An `IndependentVarRef` element maps to a “C” variable reference (line 209), that is: its name, which will be the same as the @varID attribute of the corresponding `variableDef` element.

**DependentVarRef Element**

An `IndependentVarRef` element likewise maps to a “C” variable reference whose name is the same as the @varID attribute of the corresponding `variableDef` element. Dependent variable references appear on the left side of the assignment from the interpolator function calls (line 186).

**functionDefn Element**

A `functionDefn` element would generally map to some data structure that would consolidate the data necessary to specify either a gridded or an un-gridded interpolator. The data structure would contain the following information:

- gridded | ungridded
- depending on the above type, either
- the data structure that specifies a gridded interpolator
- the data structure that specifies an ungridded interpolator.

Because of our initial decision to limit the scope to gridded interpolation, we don’t need to create this consolidation structure. We only use the structure necessary for gridded data.

**griddedTable Element**

A `griddedTable` element maps to an instance of some structured data type that consolidates the data necessary to specify a gridded interpolator.

The data type would need to contain the following information:

- An ordered collection of (size/size) pairs, one for each breakpoint array associated with the gridded interpolator,
- The size (cardinality) of the above collection,
- A reference to the data within which we are interpolating.

A unique identity for an instance of this “consolidation” data structure is needed. It might conceivably be derived from the @name attribute of the `griddedTable` element. A problem with this is that the @name attribute is not guaranteed (by the DTD) to exist, and it is not guaranteed to be unique.

A workaround is that its identity can be derived from our knowledge that a `griddedTable` element is only contained in a `functionDefn` element, which is only contained in a `function` element which contains one `independentVarRef` element, which has a unique name.

**griddedTableDef Element**

A `griddedTableDef` element maps to an instance of a data structure containing exactly the same type of information as the `griddedTable` element (above).

A unique identity for an instance of this “consolidation” data structure is also needed. A `griddedTableDef` may have an @Id attribute, but it is not guaranteed to exist. It cannot derive an identity from its unique existence in a parent element with a unique identity, because it can be a child of the root element `DataHdl`. In this case a unique
identity can’t be derived from a parent element’s identity as it was in the case of the `griddedTable` element, because the parent element is the root element.

**dataTable Element**
A `dataTable` element maps to an array of type double (line 326). The identity of the array is derived from the identity of the parent `<griddedTableDef>` or `<griddedTable>`.

```java
double CX_table_data_table[] = {
   -0.99, -0.8, -0.81, -0.63, -0.25, 0.044, 0.097, 0.113, 0.145, 0.167, 0.174, 0.166,
   -0.048, -0.038, -0.040, -0.021, -0.016, 0.083, 0.127, 0.137, 0.162, 0.177, 0.179, 0.167,
   -0.022, -0.020, -0.021, -0.004, 0.032, 0.094, 0.126, 0.130, 0.154, 0.161, 0.155, 0.138,
   -0.040, -0.038, -0.039, -0.025, 0.006, 0.062, 0.087, 0.085, 0.100, 0.110, 0.104, 0.091,
   -0.083, -0.073, -0.076, -0.072, -0.046, -0.012, 0.024, 0.025, 0.043, 0.053, 0.047, 0.040
};
```

**Workaround for the Gridded Table Identities:**
In lines 59...67, interpolation data structures are selected and transformed by processing both `griddedTableDef` elements (template name = `interp_data_from_gridded_table`) and function elements (template name = `interp_data_from_function`) at the top level (to extract embedded `griddedTableDef` and `GriddedTable` elements).

The “`interp_data_from_gridded_table`” template takes an identity parameter (interpID). From this identity, identifiers for the components of the corresponding “C” structures are derived.

Line 60, which processes top-level `griddedTableDef` elements, calls the “`interp_data_from_gridded_table`” template (line 227) using the @interpID as the identity parameter to generate interpolation data structures.

Line 66 calls the “`interp_data_from_function`” template (line 161) which in turn calls the “`interp_data_from_gridded_table`” template using the dependent variable as the identity parameter for both embedded `griddedTable` and `griddedTableDef` elements.

**breakpointRefs Element**
A `breakpointRefs` element represents a collection of breakpoint references. It maps to two arrays (line 280). The first array is a list of pointers to breakpoint arrays. The second is an array of the sizes of each of the arrays pointed to by the first. The names of the arrays are derived from the identity of the parent of the `<breakpointRefs>` element.

```java
double* CX_table_breakpointRefs[] = {DE1, ALPHA1};
int CX_table_breakpointSizes[] = {
   (sizeof(DE1)/sizeof(DE1[0])),
   (sizeof(ALPHA1)/sizeof(ALPHA1[0]))
};
```
bpRef Element
A breakpoint reference is the value of the @bpID attribute of the bpRef element.

griddedTableRef
A griddedTableRef element maps to a “C” variable reference, i.e., its name, which will be the same as the @glID attribute of the corresponding griddedTableDef element.

breakpointDef Element
A breakpointDef element maps to a “C” array of type double. The name of the array is the value of the @bpID attribute of the breakpointDef element. The contents of the array initializer are the values contained within the bpVals element.

double ALPHAL[] = {-10.,-5.,0.,5.,10.,15.,20.,25.,30.,35.,40.,45.};

bpVals Element
A bpVals element maps the numbers within the initializer for the array corresponding to the breakpointDef element.

4.4.3 Lessons Learned
Because we were able to transform the HL-20 and F-16 DAVE-ML files to executable “C” models, it appears the development of a general process is very feasible.

A griddedTableDef element at the root level of a legal (according to the DTD) DAVE-ML file is not required to have an @glID. This makes it impossible to guarantee that that file can be transformed to “C”. In order to make that possible, a user would currently be required to manually verify that griddedTablesDefs did have glIDs.

Although we’ve only created transformations for a subset of MathML operators, and we’ve not yet determined how to transform piecewise elements, we do believe that there is likely a solution.
Listing 1 – DMLtoC.xsl

<!-- DAVE-ML to "C" Language Translator

Development Task: NESC Flight Simulation Model Exchange
Development Lead: E. Bruce Jackson (bruce.jackson@nasa.gov)
Developer: John M. Penn (john.m.penn@nasa.gov) (3 Communications)
Developing Organization:
Simulation and Graphics Branch
Software, Robotics and Simulation Division
Engineering Directorate
NASA Johnson Space Center

This transform is an artifact of a study. It was created to provide just enough
functionality to transform the two DAVE-ML file examples that we had and is NOT
GUARANTEED to work for all possible DAVE-ML files that can be created. It is not
a finished product, but it might be a useful start. John M. Penn

$Id: DMLtoC.xsl 119 2010-09-28 14:30:23Z m.jessick $
-->
```
61 <xsl:with-param name="interpID" select="@gID"/>
62 </xsl:call-template>
63 </xsl:for-each>
64 </xsl:for-each>
65 </xsl:for-each select="function">
66 <xsl:call-template name="interp_data_from_function"/>
67 </xsl:for-each>
68 </xsl:for-each>
69 </xsl:for-each select="#text">
70 <xsl:call-template name="interp_data_from_function"/>
71 </xsl:for-each>
72 </xsl:for-each>
73 <xsl:for-each select="variableDef">
74 <xsl:call-template name="variableDef"/>
75 </xsl:for-each>
76 <xsl:for-each select="#text">
77 <xsl:call-template name="interp_data_from_function"/>
78 </xsl:for-each>
79 <xsl:for-each select="variableDef">
80 <xsl:call-template name="variableDef"/>
81 </xsl:for-each>
82 <xsl:for-each select="#text">
83 <xsl:call-template name="interp_data_from_function"/>
84 </xsl:for-each>
85 <xsl:for-each select="variableDef">
86 <xsl:call-template name="variableDef"/>
87 </xsl:for-each>
88 <xsl:for-each select="#text">
89 <xsl:call-template name="interp_data_from_function"/>
90 </xsl:for-each>
91 <xsl:for-each select="variableDef">
92 <xsl:call-template name="variableDef"/>
93 </xsl:for-each>
94 <xsl:for-each select="#text">
95 <xsl:call-template name="interp_data_from_function"/>
96 </xsl:for-each>
97 <xsl:for-each select="variableDef">
98 <xsl:call-template name="variableDef"/>
99 </xsl:for-each>
100 <xsl:for-each select="#text">
101 <xsl:call-template name="interp_data_from_function"/>
102 </xsl:for-each>
103 <xsl:for-each select="variableDef">
104 <xsl:call-template name="variableDef"/>
105 </xsl:for-each>
106 <xsl:for-each select="#text">
107 <xsl:call-template name="interp_data_from_function"/>
108 </xsl:for-each>
109 <xsl:for-each select="variableDef">
110 <xsl:call-template name="variableDef"/>
111 </xsl:for-each>
112 <xsl:for-each select="#text">
113 <xsl:call-template name="interp_data_from_function"/>
114 </xsl:for-each>
115 <xsl:for-each select="variableDef">
116 <xsl:call-template name="variableDef"/>
117 </xsl:for-each>
118 <xsl:for-each select="#text">
119 <xsl:call-template name="interp_data_from_function"/>
120 </xsl:for-each>
```

---

Convert a `<variableDef>` element to a "C" language variable declaration.
<?xml version="1.0" encoding="UTF-8"?>
<template name="var_declaration" match="variableDef">
  <text>
    double 
  </text>
</template>

<?xml version="1.0" encoding="UTF-8"?>
<template name="var_initialization" match="variableDef">
  <if test="@initialValue">
    <text>
      @initialValue
    </text>
  </if>
</template>

<?xml version="1.0" encoding="UTF-8"?>
<template name="bp_declaration" match="breakpointDef">
  <text>
    double [] = [ ]
  </text>
</template>

<?xml version="1.0" encoding="UTF-8"?>
<template name="Interp_data_from_function" select="Function">
  <variable name="depVarId" select="dependentVarRef/"varId"/>
  <for-each select="FunctionDef/griddedTableDef">
    <template name="Interp_data_from_griddedTable">
      <with-param name="interpID" select="concat($depVarId,'_',interp')"/>
    </template>
  </for-each>
</template>
Generate interpolator function calls:

```xml
<template name="Interp_call_from_function" select="Function">
  <variable name="depVarId" select="dependentVarRef/@varId"/>
  <value-of select="$depVarId"/>
  <text>call interp($depVarId,'interp')</text>
</template>
```

```xml
<template name="Interp_data_from_griddedTable" select="griddedTableDef@interpgriddedTable">
  <param name="interpID" select="FIXME"/>
  <text>&amp;&#38;#text;</text>
  <text>INTERPOLATION DATA</text>
  <text>interpID</text>
</template>
```

```xml
<template name="breakPointRef"/>
```

```xml
<call-template name="gen_C_hdrfile_array">
  <with-param name="interpID" select="$interpID"/>
</call-template>
```
<!-- Convert <breakpointRefs> element to an array of pointers to breakpoint arrays generated by the "bpDeclaration" template above. 

Parameter: 
interpID -Interpolator ID. 

-->
The data table is part of the interpolation data.

Generate an assignment statement from a <calculation> element and the @var ID attribute of its parent <variableDef> element.
 Flight Simulation Model Exchange
4.5 HL-20 Auto-Landing Simulation

The FSME program leveraged the existing DAVE-ML HL-20 aerodynamic model to practice transferring an extensive aerodynamic model between organizations and building a simulation from it.

At NASA/JSC, we built up a new HL-20 simulation using the Trick Simulation Environment and the JSC Engineering Orbital Dynamics (JEOD) models as a base. We extended JEOD’s aerodynamic model hook and built an object-oriented class system for embedding four different selectable HL-20 aero model implementations into the simulation. Along the way, general methods of connecting internal DAVE-ML interpreter variables to simulation variables were created. (The software integration was discussed in more detail in previous sections.)

The new simulation results were reviewed against previously published results including:

- Internal DAVE-ML checkData cases for the HL-20 aero model (16)(17), run through the two interpreters,
- Visual comparison with the step response plots reported in TM107580(16),
- Visual comparison with the published trajectory of TM107580(16).

An example pitch-step response, comparing the expected result as published in TM107580 to the Trick simulation result is shown in Figures 10 and 11.
The trajectory comparing to the single case documented in TM 107580 was in rough agreement, although the precision of the visual comparison was not high, due to the lack of digital comparison profiles.

Several representative trajectory profile plots for approach along a 135 degree heading with a left turn around the Heading Alignment Cylinder (HAC) to a final approach along a 0 degree heading are shown in Figure 12 and 13 below. Note that this was the trajectory used for the Execution Time Study reported in the next section.
Figure 12. HL-20 Auto-Landing Trajectory, Overhead Map
The GN&C algorithm as auto-coded and integrated into this simulation may be operating differently in the supersonic regime than the legacy TM107580 algorithm (16). The ability to hold the commanded angle-of-attack seems reduced. This may point to a lingering algorithm re-hosting or auto-coding issue. Since the use of the simulation was primarily to integrate and evaluate the DAVE-ML based aerodynamic model, rather than the auto-coded GN&C algorithm, further investigation of this anomaly was suspended in favor of other tasks.

It is interesting to note the high degree of uniformity among the trajectories from all four versions of the DAVE-ML HL-20 aero model. As seen in Table 2, the touchdown points after 400 seconds of flight are identical for the three models interpreting or generated from the same DAVE-ML file, and different from the legacy hand coded C model by less than 2 feet.

Table 2. Touchdown Conditions Compared For The Four HL-20 Aero Model Versions

<table>
<thead>
<tr>
<th>Aero Model</th>
<th>Runway Relative Downtrack, ft</th>
<th>Runway Relative Crosstrack, ft</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dave/MITranslator</td>
<td>420.723151</td>
<td>-20.102880</td>
</tr>
<tr>
<td>Janus</td>
<td>420.723151</td>
<td>-20.102880</td>
</tr>
<tr>
<td>XSLT Generated C code</td>
<td>420.723151</td>
<td>-20.102880</td>
</tr>
<tr>
<td>Legacy Hand Coded C</td>
<td>418.834007</td>
<td>-20.133210</td>
</tr>
</tbody>
</table>
5.0 EXECUTION TIME STUDY OF DAVE-ML INTERPRETERS

This task compared the execution time of four versions of the HL-20 aerodynamic model using a single auto-landing scenario as the duty cycle. The execution time taken to process a single update of the HL-20 aerodynamic model was measured for the following versions of the model:

- DaveMLTranslator DAVE-ML interpreter,
- Janus DAVE-ML interpreter,
- XSLT generated from DAVE-ML, with hand finishing,
- Legacy hand coded C version of the model.

The intent of this study was to measure the execution time of the different model implementations as embedded within a representative Trick/JEDO installation, over a single representative trajectory for a large DAVE-ML model, and a single computer system. (Additional constraints are discussed below under “Weaknesses”.)

Note that although the HL-20 aero model has many table look-ups, they all use the simplest linear interpolation technique. While it can be considered “large”, it likely falls short of “complex” relative to the full spectrum of possible DAVE-ML models.

5.1 Trick Job Timing System

The Trick Simulation Environment job timing system was used to collect the data. This system uses the Linux `clock_gettime(CLOCK_REALTIME)` function and provides results to microsecond resolution. This function uses the Intel hardware “RDTS in time stamp counter” for the test computer used.

The Trick job that includes the JEDO aero drag function was measured. A small amount of administrative code calculating angle of attack and sideslip from body velocity inputs, and copying input values into the DAVE-ML model, and copying output values out of the DAVE-ML model, was included in each measurement. Note that this is several levels above the interpreter. It was assumed that the cost of these overhead routines is small in comparison to the time required to perform the DAVE-ML model calculations in the interpreter, and would be approximately the same for each of the four tested versions of the HL-20 aero model.

Because the Trick job timing system does not allow the collection of data for “derivative” type job calls (which is the job class appropriate for aero model jobs), an additional job was programmed to occur outside the numerical integration collection of aerodynamic forces. Because these additional timing jobs are interspersed with the normal derivative jobs (in this case, 4 calls per simulation dynamics frame for the Runge-Kutta fourth order integration technique), the effect of the derivative jobs on the measurements is unclear. It is conceivable that optimizations inside the interpreters could allow the measured jobs to perform better than the derivative jobs which could not be measured using the standard Trick system.

5.2 Test Computer and Operating System Specifications

The single test computer and operating system used for the testing had the following specifications:

Specifications: 2 processor Intel Core™ 2 6600 CPU at 2.4 GHz, cache size: 4096 KB
Operating system: CentOS 5 (kernel 2.6.18-8.1.8.el5) for x86_64

This computer was procured circa late 2007 to early 2008.
5.3 Compiler and Compilation Flags

This section documents the build and compilation flags used for each library. We used the highest level of optimization recommended for each library in their included documentation, installation and build files.

**Janus**: 
```
g++ -D_REENTRANT -I.. -I./Janus -fpIC -O6 -Wall -march=x86-64 -funroll-loops 
-finline-functions -c Janus.cpp
```

**qbull**: 
```
CCOPTS1 = -O2 -ansi -fno-strict-aliasing
```

**Xerces**, **version 2.8.0**: as of version 2.8.0, the xerces makefile uses the -O2 optimization level for GNU/Linux with the g++ compilers.

**DaveM/Translator**: 
```
-O3 -funroll-loops -finline-functions
```

**Simulation**: 
```
-O3 -funroll-loops -finline-functions (affects JBOD models and sim models)
```

5.4 Miscellaneous Settings

Trick flags locking the CPU of the main simulation process to the second core and locking the process into memory were used. Core 0 was specified for a secondary Thrce process (the Variable Server). Using these settings resulted in fewer outliers in the timing data.

5.5 Weaknesses of the Test Method

Several weaknesses are inherent in the method used to measure the duration of each model update.

These included the following:

- The measured job includes some (assumed constant) overhead needed to hook the model to the simulation:
  - Calculation of angle of attack and sideslip inputs
  - Input hookups copying data from simulation variables to Interpreter variables
  - Output hookups copying data from interpreter to simulation variables

- Only a single computer, architecture, compiler, operating system, and test case duty cycle were measured.

- The duty cycles selected, while the largest available within the capabilities of all the methods tested, does not include use of the most complicated and time consuming elements (e.g., ungridded tables). Of the four methods, only Janus has ungridded table capability.

- This restricts the test case to only the simpler constructs, severely biasing the results.

5.6 Timing Results

Figure 14 shows the raw per call processing duration results of the timing study between the four versions of the HL-20 aero model over the 400 seconds of the simulated HL-20 Auto-Landing trajectory.
Only the XSLT generated code shows any marked difference in the times per call over the trajectory. The more polished algorithms show flat profiles throughout the trajectory. The reason for the jump in the XSLT call duration at 105 seconds is unknown. It doesn't seem to correlate with any event of importance, except perhaps crossing under Mach 2. (The jump occurs well before the start of turning around the Heading Alignment Cylinder.) It was originally theorized that the increased CPU time in this region was due to not optimizing the search for the table look-up step in the independent variable which was naively repeated from the first breakpoint for each table (e.g.: 193 times per call for Mach). However, an experimental version of the simple recursive linear interpolator, which retained and re-used the solution of the most recent lookup, only improved the times for the XSLT generated model by around 2 microseconds all across the trajectory without changing the shape of the profile. Since performing an optimization study wasn't the point of the XSLT code generation feasibility exercise, further study of this interesting characteristic was not performed.

Table 3 shows a summary of the data, with overall representative time values chosen by eye.

<table>
<thead>
<tr>
<th>Aero Model</th>
<th>CPU Time Per Call, micro-secs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Janus</td>
<td>73</td>
</tr>
<tr>
<td>DaveMITranslator</td>
<td>24.5</td>
</tr>
<tr>
<td>XSLT Generated C code</td>
<td>22.5 - 28</td>
</tr>
<tr>
<td>Legacy Hand Coded C</td>
<td>18</td>
</tr>
</tbody>
</table>

NESC Request No.: 09-00598
5.7 Interpretation of the Results

The CPU times required to process this fairly large aero model (240 one and two dimensional table look ups) in interpreted DAVE-ML, auto-generated C/C++ code from DAVE-ML, and in hand coded C code, were seen to be fast (under 100 microseconds), and similar (within a factor of 4) for the single test computer, operating system, and other simplistic conditions of this test.

Due to the weaknesses already described in the limited testing performed, no further general conclusions should be drawn from these results.

We feel it was valuable to have more than one interpreter available, and the prototype XSLT code generator was also quite useful to have. Overall, these methods provided a spectrum of relatively quick and simple ways to implement a DAVE-ML transferred model into a simulation facility while providing performance reasonably close to hand coded algorithms. Even should real-time requirements require a customized hand coded version of a transferred model to be constructed from the DAVE-ML data, interpreted and XSLT code generated versions of the algorithm can be used to provide valuable independent comparison checks on the resulting hand coded algorithm.

6.0 NON-AERODYNAMICS MODELS IMPLEMENTED IN DAVE-ML

Two non-aerodynamic DAVE-ML models with various special features were constructed to exercise DAVE-ML outside this sphere. These models are described in the following sections.

6.1 HL-20 RCS Algorithm

The HL-20 RCS control algorithm described by Powell(18) was implemented in DAVE-ML to investigate several interesting features of this model:

- A “pseudo-dynamic” model with 24 saved states implemented via caller storage
- Use of an XSLT script to expand “macros” collecting common sub-objects, easing MathML authoring;
  - Load and lag compensator digital filters: `<compensatorZ>` element
  - Relay blocks: `<relay>` element
  - Limiter block: `<limiter>` element

The DAVE-ML implementation of the Powell RCS algorithm (see Figure 15) follows closely the MatLab™ Simulink™ implementation provided by NASA/LaRC.

NESC Request No.: 09-00598
The algorithm sub-objects mentioned above were implemented as “macros” allowing expansion to DAVE-ML 2.0 markup. The macros were placed in `<variableDef>` elements as children of `<calculation>` blocks. It seemed reasonable that future DAVE-ML additions would allow any such new element types to be siblings of `<calculation>` blocks, allowing their use in variable definitions.

Several different future forms of the compensator blocks were envisioned, e.g., S-domain and Z-domain coefficients, etc. Since DAVE-ML 2.0 compatibility was important during this investigation, the Z-domain form was chosen, sidestepping the issue of providing a menu of different discretization methods. (When directly implementing the Z-domain transfer function form, only one discretization method needed to be implemented.)

The macro concept allowed the blocks to be coded in a new format, but implemented in DAVE-ML 2.0 compliant XML for use with current interpreters after expanding the macro using XSLT. (See next section.)

### 6.1.1 Ad hoc “Macro” Substitutions via XSLT

Implementation of this model benefited from the idea of extending DAVE-ML 2.0 through using special “macro” elements implementing algorithm sub-objects similar to MatLab® Simulink® “Blocks”. Future DAVE-ML versions might include specific elements for these or similar objects, using them directly. Here, XSLT was used to expand the macros to DAVE-ML 2.0 compliant form (using MathML elements) to allow the use of current DAVE-ML interpreters until such time.
Example limiter block logic expressed in C/C++

```c
// limiter element performs a two-sided limiting operation
if (input > highLimitValue) {
    output = highLimitValue;
} else {
    if (input < lowLimitValue) {
        output = lowLimitValue;
    } else {
        output = input;
    }
}
```

This element requires no saved states.

Example <limiter> macro element

```xml
<limiter inputID="limiterRollInput" lowLimitValue="-5.0" highLimitValue="5.0" />
```

Example digital compensator block logic as difference equation

A difference equation form of the digital compensator is:

\[ Y_i = N_0 X_i + N_1 X_{i-1} - D_1 Y_{i-1} \]

Where:
- \( Y \): output signal
- \( X \): input signal
- \( N \): digital transfer function numerator coefficients
- \( D \): digital transfer function denominator coefficients

This element requires two saved states: \( X_{i-1} \) and \( Y_{i-1} \).

Example <compensator2> macro element

```xml
<compensator2 inputID="compRollInput"
               prevInputID="prev_compRollInput_input"
               prevOutputID="prev_compRollOutput_input"
               numz0="1.0" numz1="-0.9091" denz1="-0.9394" />
```

This compensator implements a digital filter with the corresponding S-domain transfer function:

\[ \frac{s^2 + 2\sigma s + \sigma^2}{s^4 + 2\beta s^3 + \beta^2 s} \]

with the digital filter coefficients calculated via the “zero order hold” discretization method for a time step of 0.08125 seconds (32 Hz).

Example relay block logic expressed in C/C++

```c
// relay element performs an input/output switch mapping with hysteresis:
if (prevOutput > lowSwitchValue) {
    // was on
    if (input < lowSwitchValue) {
        output = lowOutputValue;
    } else {
        output = highOutputValue;
    }
}
```

NESC Request No.: 09-00598
This element requires one saved state (prevOutput).

**Example <relay> macro element**

```xml
<relay inputID="relayRollInput" prevOutputID="prev_relayRollAOoutput_input"
    loSwitchValue="0.5" hiSwitchValue="1.0"
    loOutputValue="0.0" hiOutputValue="2.0" />
```

**Example XSLT script to expand the <relay> element: relay.xsl**

```xml
<?xml version="1.0" standalone="no"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="xml" encoding="iso-8859-1" indent="yes" />
    <xsl:template name="relay_block">
        <xsl:comment> &lt;relay&gt; element:
            inputID="<xsl:value-of select="@inputID"/>">
            prevOutputID="<xsl:value-of select="@prevOutputID"/>">
            loSwitchValue="<xsl:value-of select="@loSwitchValue"/>"
            hiSwitchValue="<xsl:value-of select="@hiSwitchValue"/>"
            loOutputValue="<xsl:value-of select="@loOutputValue"/>
            hiOutputValue="<xsl:value-of select="@hiOutputValue"/>
        </xsl:comment>
    </xsl:template>
    <xsl:template name="apply">
        <xsl:apply-tag>
            <piece>
                <xsl:apply-tag>
                    <piece>
                        <xsl:value-of select="@loOutputValue"/></piece>
                    <xsl:apply-tag>
                        <lt>
                            <xsl:apply-tag>
                                <xsl:apply-tag>
                                    <xsl:value-of select="@inputID"/>
                                </xsl:apply-tag>
                                <xsl:apply-tag>
                                    <xsl:value-of select="@loSwitchValue"/>
                                </xsl:apply-tag>
                            </xsl:apply-tag>
                            </xsl:apply-tag>
                        </xsl:apply-tag>
                        <if>
                            <xsl:apply-tag>
                                <xsl:value-of select="@inputID"/>
                            </xsl:apply-tag>
                            <xsl:apply-tag>
                                <xsl:value-of select="@loOutputValue"/>
                            </xsl:apply-tag>
                        </if>
                        </xsl:apply-tag>
                    </piece>
                    <otherwise>
                        <xsl:value-of select="@hiOutputValue"/></otherwise>
                    </xsl:apply-tag>
                </xsl:apply-tag>
            </piece>
        </xsl:apply-tag>
    </xsl:template>
</xsl:stylesheet>
```
Example Expanded DAVE-ML 2.0 Compliant <relay> Block

Example excerpt showing a sample DAVE-ML variableDef (variable definition) element featuring an expanded relay macro in DAVE-ML 2.0 compliant XML:

```
<variableDef name="relayRoll1A" varID="relayRoll1A" units="nd" initialValue="0.0">
  <description>Roll relay Ac</description>
  <instance>
    <relay element="relayRoll1Input">
      <prevOutputID="prev_relayRoll1AOutput_input" />
      <loSwitchValue="0.5" />
      <hiSwitchValue="1.0" />
      <loOutputValue="0.0" />
      <hiOutputValue="2.0" />
    </relay>
  </instance>
</variableDef>
```
Here, the MathML was simply expanded to one very long unreadable source line, since the purpose of the macro was to abstract and encapsulate this MathML construct. (Minor additions to the XSLT script could be made to format this in human-readable/editable form, if desired.)

6.1.2 Test Method

The HL-20 RCS algorithm was tested using a rigid body simulation featuring the DAVE-ML HL-20 aero model at low dynamic pressure (where the RCS jets can override the natural aerodynamic stability of the vehicle). The RCS thruster model used for the HL-20 was given in Reference (18) and repeated in Table 4 below.

<table>
<thead>
<tr>
<th>Axis</th>
<th>Moment (per thruster)</th>
<th>Number of Thrusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roll</td>
<td>100 ft-lbf</td>
<td>2 or 4</td>
</tr>
<tr>
<td>Pitch</td>
<td>333 ft-lbf</td>
<td>2 or 4</td>
</tr>
<tr>
<td>Yaw</td>
<td>333 ft-lbf</td>
<td>1, 2 or 3</td>
</tr>
</tbody>
</table>

The HL-20 rigid body mass model (also from(18))) is repeated in Table 5.

<table>
<thead>
<tr>
<th>Axis</th>
<th>Inertia</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roll</td>
<td>7512 slug-ft²</td>
</tr>
<tr>
<td>Pitch</td>
<td>33594 slug-ft²</td>
</tr>
<tr>
<td>Yaw</td>
<td>35644 slug-ft²</td>
</tr>
</tbody>
</table>

The table above shows the mass and moment of inertia for each axis.

An artificial test state was constructed that reduced the dynamic pressure by a factor of 1,000. The state was otherwise correctly represented at 100,000 feet altitude and Mach 4. This adjustment was required because the DAVE-ML HL-20 aero models' range of validity extends from Mach 0.0 to only Mach 4, well below the Mach number where the low dynamic pressure region of RCS utility is usually encountered.

6.1.3 Results

The attitude performance of the algorithm in angle of attack and roll was tested using a series of ramp inputs, as shown in Figures 16, and 17. Figure 18 shows the profile of body rates resulting from thruster firings over the same time period. Because the algorithm was designed to maintain zero sideslip angle, no ramp command in yaw was used. However, yaw firings were required as seen in Figure 18 to maintain low sideslip during the combined pitch and roll maneuver of the test.
Figure 16. DAVE-ML HL-20 RCS Algorithm Test Results: Angle of Attack
Figure 17. DAVE-ML HL-20 RCS Algorithm Test Results: Roll Angle
6.1.4 Lessons Learned

Pseudo-dynamic models were constructed under the existing DAVE-ML 2.0 DTD by using the caller to store what would be internal variables in a postulated future dynamic DAVE-ML. For now, the caller sends the states in as additional inputs, accepts them back, as model outputs, and re-routes the values back into the input storage for the next call. Using the TDM_Map system (discussed above in the Software Section) to hook the DAVE-ML models to the Trick simulation, we were able to use the same simulation-side variables hooked to both the DAVE-ML input and output variables, so no explicit copying from model output to input was required between calls. This made the system very convenient to use. The caller essentially just provides a work array of the proper size for the saved states and prepares the TDM_Map XML file indicating the hookup of each work array element to each DAVE-ML internal variable representing the dynamic states for both input and output.

"Macros" expanded via a capable scripting language such as XSLT can be used to considerably simplify the authoring of complex MathML constructions. This allows the model author to make useful ad hoc "additions" to DAVE-ML and then generate DAVE-ML 2.0 compliant markup on-the-fly via XSLT for use with existing interpreters and DTD. If future DAVE-ML versions implement similar blocks, the implementation will likely be much closer to the macro element form than to the DAVE-ML 2.0 MathML based expansions, so using the macro form seems likely to ease any future conversion task.
6.2 Pneumatic Tire Data Compression Model

The Pneumatic Tire Data Compression Model is a multi-input / multi-output set of three hierarchical DAVE-ML model files that coordinate to calculate the lateral forces on a tire from a set of experimental coefficients and the current simulation states of a tire (normal force F_n, slip angle Alpha, and inclination angle Gamma). A high level diagram mapping the hierarchy is shown in Figure 19.

![Diagram of Pneumatic Tire Data Compression Model]

**Figure 19. Tire Data Compression Model Set**

This model set features:
- A zero input / multiple output model
- Hierarchical model sets where the outputs from one model are used as inputs to the next
- An "equation heavy" model: no table look ups, just equations implemented in MathML.
- Use of a script to expand "macros" collecting common terms, easing MathML authoring.

A Data Compression Model, as used in the modeling of pneumatic tires for automobiles, uses a single curve fit of complex shape that can be adjusted through a set of experimentally derived and curve fit coefficients to generate tire force results over a wide range of several important input state parameter values(19). In this case, the curve fit is the "Magic Tire Formula" developed by Pacejka, as normalized by Woods(20).

The Pacejka coefficients comprising the tire lateral force model are stored in a zero input / multiple output DAVE-ML file named tireDcmParam_lateral_pacejka.dml. DAVE-ML is used here for its self documentation features. Unfortunately, some portions of the DAVE-ML 2.0 specification preclude the use of zero input models. (The checkData element, if present, requires at least one input variable as the checkInputs element requires at least one signal element.) A workaround to use this type of model with existing DAVE-ML 2.0 compliant interpreters is simply to provide a single unused dummy input. This output only model can be run once only, with the outputs input to the next model.
The second model, tireDemParam_lateral_pacejka_to_woods.dml, converts the coefficients from Pacejka to Woods format. Woods developed a useful normalization of the Pacejka model resulting in a different coefficient set that can be used in place of the standard Pacejka coefficients, with coefficients calculable from the Pacejka coefficients. This middle model can also be run once only. It takes Pacejka format coefficients as input, with the output being the set of Woods format coefficients needed for the run-time stage. The DAVE-ML checkData feature is used to check the conversion using a comprehensive set of internal test case data reported by Woods.

The DAVE-ML tire data compression model file tireDataCompression.dml makes heavy use of DAVE-ML’s ability to implement algorithms through embedded MathML elements. The model contains no table look ups, only MathML equations. It calculates the output lateral force as complex functions of the coefficients and varying input states:

\[
\text{tire lateral force} = f(\text{slipAngle, normalForce, inclination, coefficients})
\]

Example output from the model is shown in Figure 20. The figure compares the output lateral force for several normal force load levels over a range of slip angles. The variation of the ratio of lateral force to maximum lateral force and the slip angle for peak force both vary smoothly under control of the model. The effect of the asphericity modeled in the example tire model coefficients provided by Woods (20) is evident as the friction coefficients cross zero at a small positive slip angle rather than zero. The small but non-zero offset forces programmed into the model also cause the friction coefficients to exceed the expected zero normal force maximum at low normal force levels.

The slip angle for peak friction coefficient for these tire model parameters becomes smaller as the load increases.
6.2.1 Ad hoc “Macro” Substitutions via Script

Implementation of this model benefited from the concept of extending DAVE-ML 2.0 through the use of special “macro” elements similar to those used in the HL-20 RCS algorithm. The Woods normalized form of the Pacejka Magic Tire Formula makes repeated use of Linear Degradation and Quadratic Degradation constructs which proved straightforward to abstract out as macros. (This is the reason this form of the tire model was used.) However, for this model, XSLT was not used to expand the macros to DAVE-ML 2.0 compliant form as was done for the HL-20 RCS model. Instead, the object oriented high level (scripting) language Ruby (21) was used to expand the macros into DAVE-ML 2.0 compliant markup.

Several restrictions to the format of the macros were imposed to enable use of an extremely simple parsing system to detect and expand the macro:

- Attributes all present
- Attributes in the expected order
- Entire macro element on a single line
- No other elements allowed on the macro line

This allowed the parsing of the macro and attributes to be performed entirely within regular expression queries as the input file was processed line-by-line, greatly simplifying the script.
Example common Quadratic Degradation term in equation form:

\[ 1 - 0.1 \left( \frac{y}{y_{pp}} \right)^2 \]

Macro source form of the Quadratic Degradation term:

```xml
<quadraticDegradation constValue="1.0" gainValue="-0.1" numID="gamma" denID="gamma_y_mu"/>
```

Script output excerpt of the DAVE-ML 2.0 compliant form of the Quadratic Degradation after expansion:

```xml
<times/>
<cn>1.0</cn>
<plus/>
<times/>
<cn>-0.1</cn>
<times/>
<apply>
  <apply>
    <apply>
      <apply>
        <divide/>
        <ci>gamma</ci>
        <ci>gamma_y_mu</ci>
      </apply>
    </apply>
    <cn>2</cn>
  </apply>
</apply>
</apply>
```

6.2.2 Lessons Learned

Macros expanded via a capable scripting language using a simple parsing strategy can be used to considerably simplify the authoring of complex MathML constructions where repeated portions can be collected into macros. This provides a second technique alongside XSLT for expanding on-the-fly ad hoc extensions to the DAVE-ML specification.

Use of DAVE-ML for self-documenting data files is impeded by the DTD restrictions against no input models. Although a trivial workaround is available, the DAVE-ML specification could be modified to allow these sorts of models without employing the workaround of using a dummy input variable.
7.0 DAVE-ML SPECIFICATION COMMENTS/SUGGESTIONS

7.1 Uncertainty Element

As part of our investigation of DAVE-ML, we studied the uncertainty XML element in the DAVE-ML specification. As a result of this study, we (1) collected some observations on the element itself and (2) wrote a proposed replacement for the section of the DAVE-ML Reference Manual that addresses uncertainty.

A draft of our proposed section to the Reference Manual is included in the appendix. It probably warrants some follow-up conversations with the community. Indeed there are places in our draft where we call out issues that we believe warrant further clarification.

Our collected observations are briefly summarized below.

The bounds attribute.

The current DAVE-ML DTD does not adequately constrain the bounds attribute in the several places where it may be used in the model XML. For example, when the uncertainty element is applied to a single data value, then only one numerical bound is meaningful for normally distributed uncertainties; but when the element is applied to a data table, there must be as many bounds as there are data values in the table. The DTD could be redesigned so that XML validity would help ensure that the correct number of bounds values are specified in the model. Also, the discussion of the bounds attribute in the reference manual is incomplete, accidentally omitting the discussion of bounds in the context of data tables. At the very least, the DTD should enforce a single bound when the uncertainty element is applied to a single value for normally distributed data.

The effect attribute.

It is our understanding that the absolute value of the effect attribute is only applicable to uniformly distributed random data. Yet the current XML design allows it for normally distributed data as well. Thus a DAVE-ML model might validate (in the sense of XML "validation") yet not make any sense. The design of the DAVE-ML XML could avoid this if the effect attribute were moved "down" in the XML from the uncertainty element to the normalPDF and uniformPDF elements. We suggest that this be done and that the DTD only allow the absolute value for the effect attribute when that attribute occurs in the uniformPDF element.

Specification semantics.

In the course of our work with the uncertainty element, we struggled to understand how its use in a DAVE-ML file is supposed to be interpreted -- what use cases it is intended to support. Although the intended use of the element on input data is easy to imagine (e.g., support Monte Carlo analysis), the intended use of the element applied to internal and especially to output data is unclear. In some cases, these intended uses can be clarified by additional discussion in the reference manual, but in some cases the standard does not specify unambiguous direction to simulation developers on how to interpret the uncertainty element.

A data interchange standard may seek to solve three different kinds of problems: (1) making the data readable by all parties (i.e., avoiding the problems implicit in proprietary file formats), (2) labeling the various data items so in addition to being able to read the file, consumers can unambiguously identify which data item is meant to serve what role in the model, and (3) defining the semantics of the data so that not only are all data items readable and unambiguously identified, but the consumer knows how to use them without consulting the producer of the data.
Simple ASCII comma separated value (CSV) files provide an example of a solution to problem #1. The data are readable by virtue of the fact that the data are in an ASCII format and are delimited by a well-defined character. However, a simple CSV file does not specify which columns have what meaning.

XML files can solve problem #1 as well as problem #2. Not only are the data readable, but they are (in most cases) unambiguously identified.

We feel that there are some applications of the DAVE-ML uncertainty element where the intended use is not currently specified by the standard and hence cannot possibly be implemented by generic simulation frameworks. For example, what does it mean for the statistics of an output data value to be specified in a model? How should a DAVE-ML compliant simulation framework process such elements? It is entirely possible that there are good answers to this problem, in which case this is just a matter of expanding the Reference Manual to clarify the intended interpretations. But if there is not universally agreed upon intended use for such cases, then the DAVE-ML community in general and the Reference Manual in particular should explicitly acknowledge that there are situations where a valid DAVE-ML model cannot be properly processed by a consumer until the engineers on the consuming side have consulted with the engineers on the producing side. It is our understanding that the motivation behind the DAVE-ML model exchange standard was to avoid the need for such consultations. Therefore we feel that the DAVE-ML community should discuss the intended semantics of the uncertainty element in more detail before encoding it in the standard.

7.1.1 Suggestions

Based on our study of the DAVE-ML model of uncertainty, we have the following suggestions. Some of these relate to the XML syntax (i.e., the DTD) and some relate to the DAVE-ML Language Manual.

- effect attribute
  - effect="absolute" only applies to uniform distributions
  - suggestion: move this attribute from the uncertainty element one level down to the normalPDF and uniformPDF elements, and don't allow effect="absolute" in normalPDF.
- percentage vs. multiplicative effects
  - effect="percentage" and effect="absolute" are different ways of saying the exact same thing (except for a factor of 100). This needlessly complicates the standard.
  - suggestion: choose one and eliminate the other.
- bounds element
  - the DTD allows any number of bounds elements even in cases where only one or two are allowed.
  - suggestion: change the DTD to precisely constrain the number of occurrences of the attribute in the 1-occurrence case for normal distributions, the 2-occurrence case for uniform distributions and the N-occurrence case for data tables.
- randomizing table data
  - There is no explicit guidance given to tool developers on how uncertain data points inside data tables should be interpolated. Should the interval boundaries be interpolated first and then a random value generated based on the uncertainty specification? Or should the interval boundaries be randomized according to the uncertainty specification and then interpolated.
- upper/lower uniform bounds in a data table
  - How should uniform distribution upper/lower bounds be encoded in a data table? For a table with N entries, there should be 2N bounds (an upper and lower bound for each table entry). The documentation in the DTD is incorrect for this case. It says that the table dimensions must match.
  - suggestion: Fix the documentation in the DTD. Clarify this point in the Reference Manual.
7.2 GriddedTableDef Element

A griddedTableDef element at the root level of a legal (according to the DTD) DAVE-ML file is not required to have a @gID. This makes it impossible to guarantee that that file can be transformed to “C”. In order to make that possible, a user would currently be required to manually verify that GriddedTableDef's did have gIDs.

7.3 CheckData Element

The current DAVE-ML 2.0 DTD does not allow zero input (or output only) models. These can be useful to take advantage of DAVE-ML's self documenting elements for static data files for "Initialization Load" (I-Load) type data. The checkData/staticSlot/checkInputs hierarchy of elements requires at least one input signal definition, once the checkData element is present. Removing this restriction would allow zero input output-only models without use of the workaround of using a dummy input currently required by the DAVE-ML 2.0 DTD.
8.0 CONCLUSIONS

Several comments and suggestions were developed concerning the DAVE-ML specification included within the S-119 Specification:

- Use of DAVE-ML for self-documenting data files is impeded by the DTD restrictions against no input models (checkData hierarchy). Suggest removing DAVE-ML DTD restrictions against no input models.
- Suggest requiring gritD attributes for griddedTableDef elements
- Suggest revising the uncertainty element specification for greater clarity.

See the Appendix for suggested revisions, with comments describing areas where further review is suggested.

“Macro” elements expanded via a capable scripting language can be used now to considerably simplify the authoring of complex MathML constructions, and possibly ease conversion to future DAVE-ML versions that implement similar features directly. XSLT or high level scripting languages can be used to transform the output to current DAVE-ML 2.0 compliant form.

General, reusable methods of hooking simulation variables to interpreted DAVE-ML model input and output variables can be created to specify the corresponding variable names conveniently via an input file.

Pseudo-dynamic models can be constructed now under the existing DAVE-ML 2.0 DTD by using a caller provided work array as additional model inputs and outputs holding what would be internal variables in a postulated future dynamic DAVE-ML. A sim-variable-to-DAVE-ML-model-variable hookup system, similar to the described TDM_Map system, can do this transparently to the user by hooking the model outputs back into the inputs by specifying the hookups via an input file that calls out the same sim variables for both input and output.

Reasonably efficient C language source code can be automatically generated from DAVE-ML via XSLT, at least for the simpler constructs. Although incomplete, the system prototyped during this project is useful now and shows considerable potential.

We feel it was valuable to have more than one interpreter available, and the prototype XSLT code generator was also quite useful to have. Overall, these methods provided a spectrum of relatively quick and simple ways to implement a DAVE-ML transferred model into a simulation facility while providing performance reasonably close to hand coded algorithms. Even should real-time requirements require a customized hand coded version of a transferred model to be constructed from the DAVE-ML data, interpreted and XSLT code generated versions of the algorithm can be used to provide valuable independent checks on the resulting hand coded algorithm.

9.0 FUTURE WORK

We have identified several potential follow-on work topics. These are:

- Further review of the S-119/DAVE-ML specifications
- Exercise an expected revision greatly expanding the HL-20 aero model
  - Even larger model
  - Possibly exercising a wider subset of DAVE-ML features
• Continue development of the XSLT DAVE-ML to C code generator
  o Increase its coverage of DAVE-ML constructs,
  o Increase robustness for production work
  o Low barrier-to-entry code generation from DAVE-ML may assist adoption of the standard

• Integrate DAVE-ML model uncertainty into the Trick Simulation Environment
  o Exercise with Monte-Carlo test cases.
10.0 BIBLIOGRAPHY

APPENDIX A   Draft DAVE-ML Uncertainty Reference

It is our feeling that a more extensive discussion of the uncertainty element in the DAVE-ML Language Reference would help clarify its intended use and thus help software developers write interoperable DAVE-ML processing code. This appendix is a draft of a proposed replacement for Section 6.4 (Statistics) section of the language reference, which we propose be renamed "Uncertainty". It is undoubtedly incomplete but nevertheless captures our thoughts on what kind of discussion is warranted.

In some places we have inserted "Author’s comments" where our understanding of the proposed XML syntax is incomplete. Additionally, please note that the section numbers in this appendix do not pertain to this document – they are actually the relevant section numbers from the DAVE-ML Language Reference.

6.4 Uncertainty

6.4.1 Introduction

6.4.1.1 Motivation

DAVE-ML supports the specification of statistical uncertainty using the uncertainty element. Its purpose is to allow models to characterize model uncertainty in two primary use cases.

Use Case 1. Support Monte-Carlo simulation by allowing statistical variation of parameter values. DAVE-ML supports specification of the uncertainty associated with constant parameters in addition to the nominal value of the parameter itself. Thus DAVE-ML compliant simulations may run multiple versions of a particular scenario, each with different parameter values within the statistical bounds specified by the associated uncertainty element associated with each parameter.

Use Case 2. Support Monte-Carlo simulation by allowing statistical variation of table lookup values. DAVE-ML supports this use case with the specification of uncertainties of dependent values whose nominal values are defined in grided or ungrided tables. Thus DAVE-ML compliant simulations may run multiple versions of a particular scenario, each with different values of the lookup data within the statistical bounds specified in the tables.

In addition, DAVE-ML also allows the specification of uncertainties associated with any signal in a model, including inputs, outputs or intermediate values. This capability is included as a “catch all” for model developers whose use cases do not fall into the two mentioned above. In these cases, DAVE-ML compliant software tools should substitute appropriately generated random values according to the semantics described below at the specified locations.

6.4.1.2 Approach

The DAVE-ML approach to statistical uncertainty is the uncertainty element which may be inserted in a model wherever the signals should have a statistical distribution about some nominal value. In particular, the uncertainty element may be a sub-element of the variableDef, gridedTableDef and ungridedTableDef elements.

Of course, DAVE-ML models only specify the statistical properties, not the random values themselves. It is up to DAVE-ML-compliant software tools to generate appropriately randomized data according to the specifications in the uncertainty element. Likewise it is up to the frameworks to seed the random number generators in some problem-appropriate fashion.¹

¹ For example, if the particular simulation run must be repeatable (albeit with randomized variable values), then the random number generator must be seeded from a deterministic value. Such seeding is beyond the scope of the DAVE-ML standard.
The following sections summarize the syntax and semantics of the uncertainty element and present some examples of its use.

6.4.2 Syntax

DAVE-ML supports specification uncertainties distributed normally or uniformly about nominal values. (These are the only possible distributions.) Which of these is used depends on the content of the uncertainty element, in particular whether the uniformPDF or normalPDF sub-elements are present. Both of these elements in turn characterize their distribution using several attributes and elements (described below). The most important of these is the bounds element which specifies the numerical characteristics of the uniform or normal distribution.

6.4.2.1 The bounds element

At root, this element is just a scalar numerical value. For example, it could be the upper or lower bound of a uniform distribution, or it could quantify the 3-sigma value for a normal distribution. However, DAVE-ML uncertainties may also be applied to data tables in which case there must be scalar values specified for each table entry.

As a result, the form of a bounds element depends on the context in which it occurs. When used in scalar contexts, it contains a single scalar value. When applied to tables, it contains a dataTable sub-element which itself contains the several bounds values the correspond to the data table values.

Accordingly, a scalar use of the bounds element might appear as follows.

```
<bounds>
  100.0
</bounds>
```

And its use inside a three-element gridded or ungridded table might appear as follows.

```
<bounds>
  <dataTable>
    100.0, 200.0, 300.0
  </dataTable>
</bounds>
```

Note: The current DAVE-ML DTD does not enforce this proper use of bounds elements. It is possible for a DAVE-ML file to validate (in the XML sense) but have incorrectly specified bounds (e.g., specify a table of bounds in a scalar context).

Author's note: This description is incomplete. We have left off the use of variableDef and variableRef sub-elements. We do not understand these.

6.4.2.2 Uniform Distributions

The DAVE-ML syntax for uniformly distributed uncertainties has the following general form.

```
<uncertainty>
  <uniformPDF effect="...">
    <bounds>...</bounds>
  </uniformPDF> <!-- optional -->
</uncertainty>
```

The effect attribute may be one of the following: additive, multiplicative or absolute.

Author's note: Our understanding of the multiplicative effect is that it is redundant with the percentage effect.
Accordingly, we recommend that one or the other be dropped from the DAVE-ML DTD. Our description here reflects that. If we misunderstand the intent of these two effects, or if the DAVE-ML community disagrees with our preference to avoid such duplication, this description must be fixed.

The bounds sub-elements specify the bounds of the interval from which the uniformly distributed random values are chosen. For symmetric distributions only one bound is needed. Asymmetric distributions require two. Thus the second bounds sub-element is optional, but at least one must be specified.

The specific meaning of the effect attribute and bounds element are explained in the semantics section below.

6.4.2.3 Normal Distributions

The XML syntax for normally distributed uncertainties has the following general form.

```xml
<uncertainty>
  <normalPDF effect="..." numSigmas="...">
    <bounds>...</bounds>
    <correlatedWith varID="..." />
    <correlation varID="..." corrcoef="..." />
  </normalPDF>
</uncertainty>
```

The effect attribute may be one of the following: additive, or multiplicative. The numSigmas attribute may have the value of 1, 2 or 3. There must be one (and only one) bounds sub-element. And there may be zero or more correlatesWith or correlation sub-elements.

The values of the varID attributes of the correlatesWith or correlation sub-elements are references to other signals which must be defined in the model, and the value of the corrCoef attribute is a numerical value.

Author's note: See the note above regarding multiplicative and percentage effects.

The specific meaning of these attributes and elements are explained in the semantics section below.

6.4.3 Semantics

The uncertainty element specifies statistical characteristics associated with uncertainties about certain nominal values. These nominal values might be explicitly defined constant parameter values, tabulated data specified in gridded and ungridded tables or implicit (input, output or intermediate) signals calculated during the execution of a DAVE-ML tool. The distribution characteristics (e.g., mean, standard deviation, upper/lower bounds) are based on the following three numerical values: a bound, the nominal value and (for normal distributions) the numSigmas value.

A bound is specified in bounds elements\(^2\). The nominal value is in general only known at runtime when the model is being executed (although for constant parameters, it is also specified in the DAVE-ML file). The numSigmas value is only relevant to normal distributions and is specified in the numSigmas attribute of the normalPDF element.

\(^2\) In the simple case, the bounds element contains the numerical bound. However, the situation is more complex for tabulated. See below for a discussion for how to determine the bound in the context of data tables.
6.4.3.1 Uniform distributions

The use of the uniformPDF element inside an uncertainty element indicates that the corresponding signal should be distributed uniformly in some interval about the nominal value. The interval may be symmetrically or asymmetrically oriented relative to the nominal value. The content of the uniformPDF element is used to specify the distribution interval.

DAVE-ML-compliant software must randomize the signal in question differently depending on the value the effect attribute. The following table summarizes how the software must calculate the randomized signal in each of the three allowable cases.

<table>
<thead>
<tr>
<th>value of the effect attribute</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>additive</td>
<td>The desired signal is the sum of the nominal value and a uniformly distributed random value.</td>
</tr>
<tr>
<td>multiplicative</td>
<td>The desired signal is the product of sum of the nominal value and the product of a uniformly distributed random value time the nominal value.</td>
</tr>
<tr>
<td>absolute</td>
<td>The desired signal is a random value uniformly distributed in some interval. (The nominal value of the signal is irrelevant in this case.)</td>
</tr>
</tbody>
</table>

For uniformly distributed random values, the relevant distribution characteristics are the upper and lower bounds on the interval from which the random values are drawn. In DAVE-ML, uniform distributions are parameterized by one or two bounds sub-elements in the uniformPDF element. When two sub-elements are present, let the greater and lesser of these be called big and little, respectively. When only one sub-element is present, let big and little both be that same value. The method of determining both bounds depends on the value of the effect attribute as explained in the following table. (All calculations are assumed to be floating point.)

DAVE-ML-compliant software will randomize the signal in question differently depending on the value the effect attribute. The following table summarizes how the software must calculate the randomized signal in each of the three allowable cases.

<table>
<thead>
<tr>
<th>value of the effect attribute</th>
<th>upper/lower bounds of the interval</th>
</tr>
</thead>
</table>
| additive                     | upper = nominalValue + big  
                              | lower = nominalValue - little  |
| multiplicative               | upper = nominalValue * big       
                              | lower = nominalValue * little  |
| absolute                     | upper = big                       
                              | lower = little                  |

6.4.3.2 Normal distributions

The use of the normalPDF element inside an uncertainty element indicates that the corresponding signal should be distributed normally about the nominal value. The content of this element is used to specify the standard deviation of the distribution around the nominal value.

DAVE-ML-compliant software will randomize the signal in question differently depending on the value the effect attribute. The following table summarizes how the software must calculate the randomized signal in each of the three allowable cases.
<table>
<thead>
<tr>
<th>value of the effect attribute</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>additive</td>
<td>The desired signal is the sum of the nominal value and a normally distributed random value.</td>
</tr>
<tr>
<td>multiplicative</td>
<td>The desired signal is the product of sum of the nominal value and the product of a normally distributed random value time the nominal value.</td>
</tr>
</tbody>
</table>

For normally distributed random values, the relevant distribution characteristics are the mean and standard deviation. In DAVE-ML, the mean is always equal to the nominal value. The method of determining the standard deviation depends on the value of the effect attribute as explained in the following table. (All calculations are assumed to be floating point.)

<table>
<thead>
<tr>
<th>value of the effect attribute</th>
<th>standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>additive</td>
<td>stddev = bound / numSigmas</td>
</tr>
<tr>
<td>multiplicative</td>
<td>stddev = ( (bound-1) * nominal-value ) / numSigmas</td>
</tr>
</tbody>
</table>

6.4.3.3 Tables of bounds

**Author’s note**: A description of how the DAVE-ML-compliant software is to determine the appropriate bounds values in the context of tables is needed. The algorithms above are not sufficient. There is ambiguity in the case of tables, because it is unclear whether the algorithms are applied before or after interpolation. To clarify, if the algorithms were applied before interpolation, then a random value would be generated at each table data point, and the desired data value would be the result of interpolation between those two random values. On the other hand, if the algorithms were applied after interpolation, then the bounds values themselves would first be interpolated to give corresponding bounds values in between the endpoints, and from that value the algorithms would be used to generate a randomized value. In general, these two approaches will yield different results, and it is not clear which is intended by the DAVE-ML standard. It is very important (for interoperability) that this be specified unambiguously. This section is where we should put an description of the intended approach.

6.4.4 Examples

**Author’s note**: We have not explicitly asserted examples here. Our intent is that the examples from the current Language Reference would fit perfectly here.

Permission granted on April 6, 2011:
AIAA grants you permission to publish the draft of S-119 in your NASA report as an appendix.
Michael Baden-Campbell
AIAA Publications

American National Standard

Flight Dynamics Model Exchange Standard

Draft 2010-04-23

Sponsored by
American Institute of Aeronautics and Astronautics

Approved XX Month 201X
American National Standards Institute

Abstract
This is a standard for the interchange of simulation modeling data between facilities. The initial objective is to allow easy, straightforward exchanges of simulation model information and data between facilities. The standard applies to virtually any vehicle model (ground, air or space), but most directly applies to aircraft and missiles.
Foreword

This standard was sponsored and developed by the AIAA Modeling and Simulation Committee on Standards. Mr. Bruce Jackson of NASA Langley conceived Dynamic Aerospace Vehicle Exchange Markup Language (DAVE-ML). DAVE-ML is the embodiment of the standard in XML. The DAVE-ML reference document, including examples of its use, and the document type definition for the XML implementation are included in this standard (Annex B).

This implementation was then tested by trial exchange of simulation models between NASA Langley Research Center (Mr. Bruce Jackson), NASA Ames Research Center (Mr. Thomas Alderete and Mr. Bill Cleveland), and the Naval Air Systems Command (Mr. William McNamara and Mr. Brent York). Numerous improvements to the standard resulted from this testing.

At the time of approval, the members of the AIAA Modeling and Simulation Committee were:
- Bruce Hildreth, Chair, Science Applications International Corporation (SAIC)
- Bruce Jackson, NASA Langley Research Center
- Michael Madden, NASA Langley Research Center
- Geoff Brian, Defence Science and Technology Organisation (DSTO)
- Brent York, Indra Systems, Inc.
- Michael Silvestro, Charles Stark Draper Laboratory, Inc.
- Victoria Chung, NASA Langley Research Center
- Bimal Aponso, NASA Ames Research Center
- Jean Slane, Engineering Systems Inc.
- Peter Grant, University of Toronto
- William Bezdek, Boeing Phantom Works
- Jon Berndt, Jacobs

The above consensus body approved this document in Month 200X.

The AIAA Standards Executive Council (VP-standards Name, Chairman) accepted the document for publication in Month 201X.

The AIAA Standards Procedures dictates that all approved Standards, Recommended Practices, and Guides are advisory only. Their use by anyone engaged in industry or trade is entirely voluntary. There is no agreement to adhere to any AIAA standards publication and no commitment to conform to or be guided by standards reports. In formulating, revising, and approving standards publications, the committees on standards will not consider patents that may apply to the subject matter. Prospective users of the publications are responsible for protecting themselves against liability for infringement of patents or copyright or both.
Introduction

The purpose of this standard is to clearly define the information and format required to exchange air vehicle simulation models between simulation facilities (see Figure 1). This standard simulation interchange format is implemented in XML and is described fully in Annex B of this document.

![Diagram showing model exchange via a standardized format](image)

Figure 1 — Model exchange via a standardized format

The standard interchange format includes:

a) Standard variable name definitions — to facilitate the transfer of information by using these standard variables as a "common language." The interchange format can be used without using standard variable names. However, it will be more difficult because the exported model will have to include explicit definitions of all variables instead of just a subset unique to the particular model.

b) Standard function table definition — to allow easy transfer of non-linear function tables of arbitrary dimension.

c) Standard coordinate system and reference frame definitions — used by the variable names and function tables to clearly define the information being exchanged.

d) Standard static math equation representation — for definition of static equations forming part of aerodynamic, propulsive or other models.

A specialized grammar of XML provides a format for the exchange of this information, therefore each organization is required to design import/export tools that comply with the standard one time only.

Use of this standard will result in substantially reduced cost and time necessary to exchange aerospace simulations and model information. Test cases have indicated an order of magnitude reduction in effort to exchange simple models when utilizing this standard. Even greater benefits could be attained for large or complicated models.
Trademarks

The following commercial products that require trademark designation are mentioned in this document. This information is given for the convenience of users of this document and does not constitute an endorsement. Equivalent products may be used if they can be shown to lead to the same results.

Simulink®
MATLAB®
1 Scope

This standard establishes definitions of the information and format used to exchange air vehicle simulations and validation data between disparate simulation facilities. This standard is not meant to require facilities to change their internal formats or standards. With the concept of an exchange standard, facilities are free to retain their well-known and trusted simulation hardware and software infrastructures. The model is exchanged through the standard, so each facility only needs to create import/export tools to the standard once. These tools can then be used to exchange models with any facility at minimal effort, rather than creating unique import/export tools for every exchange.

The standard includes a detailed convention for representing simulation variables. The purpose of this is to unambiguously describe all variables within the model when it is exchanged between two simulation customers or facilities. The variable representation includes explicit specification of all coordinate systems, units, and sign conventions used. XML is used as the mechanism to facilitate automation of the exchange of the information. Based on the definitions in the standard, a list of recommended but non-obligatory simulation variable names is included in Annex A. This list of standard variable names should further simplify the exchange of information, but is not required for use of the standard.

The standard includes capabilities for a model to be self-validating and self-documenting, with the provenance of a model’s components included within the model and transferred with it. Statistical descriptions of the quality of a model may also be included.
2 Tailoring

The requirements defined in this standard may be tailored to match the actual requirements of any particular program or project. Tailoring of requirements should be undertaken in consultation with the procuring authority where applicable.

NOTE Tailoring is a process by which individual requirements or specifications, standards, and related documents are evaluated and made applicable to a specific program or project by selection, and in some exceptional cases, modification and addition of requirements in the standards.

The following sections provide further guidance on specific tailoring situations.

2.1 Partial Use of the Standard

2.1.1 General

Not all aspects of this standard may be applicable to all models or simulation applications. The following guidelines are provided to encourage appropriate use of the standard in a number of example situations.

2.1.2 Creating a New Simulation Environment

This situation calls for use of the complete standard. It is hoped that the team developing the new simulation environment would, if necessary, add to the list of standard variables and coordinate systems.

2.1.3 Creating a New Simulation Model in an Existing Simulation Environment

This situation is defined as creating a new system model (aircraft dynamic model for example) that will run in an existing simulation environment. It is expected that this is the most commonly performed work that will see benefit by application of this standard.

In this case the following tailoring guidelines are applicable.

a) Apply the standard to the new development aspects of the project and all the function tables.

b) Assuming that most or all of the standard variable names and coordinate systems are applicable to the simulation, use them for the new code developed for the simulation.

c) In the existing simulation environment that is being reused, for example the equations of motion, there is no need to rewrite the code to use the standard variable names or coordinate systems. However, in most cases the coordinate systems used in existing simulation environments will be covered in the standard coordinate system definitions herein (Section 5). Therefore the standard coordinate systems can easily be referenced when documenting the simulation and interfaces between the new simulation components and those reused.

2.1.4 Creating or Updating a Simulation with a Long Life Expectancy

A pilot training simulator is an excellent example of this type of simulation. This simulation may only be updated every 3-10 years, so at first glance the standard may seem to be less applicable.

In fact the opposite is true. It is because of the infrequent maintenance that application of the standard is critical. In this case, in each new software update, the original developers (or previous updaters) are probably no longer available, and the update is being performed by different personnel. Software developed using the standard should be easier for the new software team to understand. They are working with clear variable definitions with which they are familiar. The function table format is understood and the functions themselves are better documented. The coordinate system definitions are clear. Changes are recorded for the benefit of any future software update.

In simulations with a long expected life, use of the state, state derivative control and output conventions as part of the variable naming convention becomes critical as these variables form the core of the model.
and the significant inputs. It is important that the personnel modifying the simulation are able to easily identify the states, state derivatives and controls.

2.2 Implementing the Standard in a non-flat or non-scalar namespace

The variable naming convention defined within the standard makes no assumption as to the hierarchy of data components, such as object-oriented model implementation or multi-dimensional storage of matrices and vectors. The standard can accommodate these implementations through the use of a period (.) inserted before the optional domain name (e.g. aerobodyForceCoefficient[0] or aerobodyForceCoefficient[1]) or through the use of an appropriate indexing mechanism for the chosen implementation language (e.g. aerobodyForceCoefficient[0] or aerobodyForceCoefficient[1]). However, it should not be expected that other members of the simulation community maintain implementation-specific conventions. Therefore, on export these variable constructs should be converted to the flat, scalar namespace defined herein.

2.3 New and Reused Software Tailoring Guidance

The longer the expected life of the simulation, the more helpful the application of the standard becomes. The above tailoring guidelines may be categorized into two common situations; new and reused code.

New simulation code should:

a) use coordinate system definitions (Section 5) that coincide with the definitions in the standard;

b) use standard variable names (Section 6) to facilitate consistency and simplify documentation requirements;

c) apply the convention for states, state derivatives and controls wherever possible; and

d) use standard function tables (Section 7) for all function tables.

NOTE: This facilitates consistency in the data, the documentation of the data, and collaboration with other organizations to improve or deploy the data.

Reused simulation code should reference the standard only when convenient to document interfaces with new code.

2.4 Creating New Variable Names and Coordinate Systems

The standard variable names and coordinate system definitions are included in the standard to facilitate communication. They provide a "common language" for the exchange. For example, it is not enough to exchange the values of the lift coefficient function. As a minimum, the independent variables used to define the function and their units, sign convention, and reference coordinate system must be defined. This need to precisely define variables is facilitated by having standard variable names and coordinate systems. Of course, new variable names, definitions, and other convenient coordinate systems may be used to exchange models between simulation facilities. However, in such cases, the exporters and importers must carefully define these variables and coordinate systems, or the exchanged model may not produce the desired results. Use of standard variable names and coordinate systems facilitates the exchange.

This standard includes a methodology for creating new standard variables. Its use is encouraged. Annex C provides the URL for submitting additional standard variable names and coordinate systems or comments on existing standard variable names and coordinate systems.
3 Applicable Documents

The following documents contain provisions which, through reference in this text, constitute provisions of this standard. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies.

AIAA R-004-1992 Atmospheric and Space Flight Vehicle Coordinate Systems
www.w3.org/XML Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008-11-26
www.w3.org/Math Mathematical Markup Language (MathML) Version 2.0 (Second Edition), 2003-10-21
ISO 19111:2007 Geographic information – Spatial referencing by coordinates
ISO 1151-2:1985 Flight dynamics – Concepts, quantities, and symbols – Part 2: Motions of the aircraft and the atmosphere relative to the Earth
ISO 1151-6:1992 Terms and symbols for flight dynamics – Part 6: Aircraft geometry
ISO 80000-1:2009 Quantities and Units – General
4 Vocabulary

4.1 Acronyms and Abbreviated Terms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIAA</td>
<td>American Institute of Aeronautics and Astronautics</td>
</tr>
<tr>
<td>AND</td>
<td>aircraft nose down</td>
</tr>
<tr>
<td>ANR</td>
<td>aircraft nose right</td>
</tr>
<tr>
<td>ANSI</td>
<td>American National Standards Institute</td>
</tr>
<tr>
<td>CM</td>
<td>center of mass</td>
</tr>
<tr>
<td>DAVE-ML</td>
<td>Dynamic Aerospace Vehicle Exchange Markup Language</td>
</tr>
<tr>
<td>DIS</td>
<td>Distributed Interactive Simulation</td>
</tr>
<tr>
<td>DTD</td>
<td>Document Type Definition</td>
</tr>
<tr>
<td>FE</td>
<td>Flat Earth coordinate system</td>
</tr>
<tr>
<td>GE</td>
<td>Geocentric Earth fixed coordinate system</td>
</tr>
<tr>
<td>HLA</td>
<td>High Level Architecture</td>
</tr>
<tr>
<td>IC</td>
<td>initial condition</td>
</tr>
<tr>
<td>ISO</td>
<td>International Organization for Standardization</td>
</tr>
<tr>
<td>ISQ</td>
<td>International System of Quantities</td>
</tr>
<tr>
<td>MathML</td>
<td>Mathematical Markup Language</td>
</tr>
<tr>
<td>MRC</td>
<td>moment reference center</td>
</tr>
<tr>
<td>MSL</td>
<td>Mean sea level</td>
</tr>
<tr>
<td>RWD</td>
<td>right wing down</td>
</tr>
<tr>
<td>SI</td>
<td>Système Internationale d’Unités</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>WGS</td>
<td>World geodetic system</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
</tbody>
</table>

4.2 Terms and Definitions

For the purposes of this document, the following terms and definitions apply.

Center of mass
This standard uses center of mass (CM) as the default location for several coordinate systems. The long-standing aeronautical tradition is to refer to this location as the center of gravity (CG). Center of gravity and center of mass are interchangeable for vehicle modeling on or above a large gravitational body in hydrostatic equilibrium like the Earth. However, the difference between CM and CG can become significant when a vehicle maneuvers near small, irregularly shaped gravitational bodies (e.g. asteroids).
Thus, center of mass is the more correct term for this important aerodynamic and kinematic reference point.

**Coordinate system**
A measurement system for locating points in space and attached to a reference frame (Stevens and Lewis, 2003). In this standard they will be orthogonal right-handed triads unless specifically noted.

**Ground**
Smooth surface of the earth at the nadir, not necessarily MSL.

**Mean Sea Level**
The zero elevation reference in the simulation, normally the geoid. Although many simulations treat MSL and the smooth surface of the earth as equivalent this is not always true. For example, the WGS84 model equates MSL with the geoid, not the smooth surface.

**Presentation coordinate system**
The specific coordinate system in which a variable or vector is presented (or expressed). For example, a specific vector may be presented (expressed) in the body axis coordinate system, the geocentric Earth (ge) coordinate system, or one of the alternatives presented in Section 5. The value of the vector's components (e.g. X, Y, & Z) differs depending on the presentation coordinate system. The presentation frame only determines how the vector is expressed as X, Y, and Z components, as the specific vector of an object is invariant with respect to any arbitrary coordinate frame (i.e. they are contravariant rank one tensors). In other words, the specific vector in two presentation frames differs only by a linear transformation (i.e. a rotation matrix). When one "presents" or "expresses" the vector in a presentation coordinate system, that presentation coordinate system is treated as if it is instantaneously fixed relative to the observer for the given time t (even if the presentation coordinate system is translating and rotating relative to the observer).

The presentation coordinate systems is identified in variable names by the initial coordinate system prefix.

**Reference frame**
Frames for short. A general definition for a reference frame is: three or more non-collinear points on a rigid body define a reference frame (Stevens and Lewis, 2003). Unlike a coordinate system, a reference frame has no fixed origin, it is in essence a rigid body wherein all points are fixed in position relative to each other. The location of a point or vector in a frame is expressed using a specified coordinate system. Any number of points or vectors may be expressed with any number of coordinate systems (with no relative motion) in the same frame. For example, the Earth is often a reference frame and may have the geocentric Earth (ge) coordinate system attached to the center, and any number of user defined topocentric coordinate systems (such as runways or launch sites) used to locate and orient fixed objects on the Earth. Note however an object moving on or above the surface of the Earth would be in a different reference frames.

**Reference coordinate system**
The coordinate system that defines the frame of interest of a rotational measurement such as attitude, angular rate or angular acceleration. Identified in variable names by the "cr" component. If not present, the default reference coordinate system is the body coordinate system.

**Reference point**
The point of interest of a translational measurement such as position, velocity or acceleration. Identified in variable names by an "rcr" component. If not specified, the default reference point is the ownership's center of mass.

**Relative coordinate system**
A coordinate system that defines the origin of a measurement such as position, velocity or acceleration (translational or rotational). Identified in variable names by the "rcr" component. If not specified, the default relative coordinate system is the same as the presentation coordinate system for translation and
the locally-level coordinate system for rotations.

**Observer coordinate system**
A coordinate system from which motion of a point (a velocity, acceleration or higher derivative) is observed (or measured). In many cases this coordinate system is in the same reference frame as the relative coordinate system, however in the most general case, may exist in a different frame. The magnitude and direction of velocity (and higher derivatives) differ depending on the observer coordinate system due to the fact that the relative coordinate system may be in motion relative to the observer. Identified in variable names by the "ObsFr" component. If not specified, the observer coordinate system defaults to the same coordinate system as the relative coordinate system.

**Velocity**
The first derivative of position; in the general case, can be applied to either translational or rotational rate-of-change of position. This term normally applies to translational motion; the rotational equivalent is normally called "angular rate."

**Inertial velocity**
The special case of a velocity for which the relative reference and observer coordinate systems are an inertial frame.

**Breakpoint**
A value of an independent variable at which the value of its dependent variable is specified, or the x coordinate (or abscissa) of a one dimensional table

**Confidence interval**
An estimate of the computed or perceived accuracy of the data

**Dependent variable**
An output that is obtained by evaluation of a tabulated function or a MathML expression

EXAMPLE For \( C_\alpha(\alpha,\beta) \), \( C_\alpha \) is the dependent variable, also called the output.

**Independent variable**
The input(s) to a function table or a MathML expression

EXAMPLE For \( C_\alpha(\alpha,\beta) \), \( \alpha \) and \( \beta \) are independent variables.

**One-dimensional table**
A table whose values are based upon only one independent variable

EXAMPLE \( C_\alpha(\alpha) \) may be represented by a one-dimensional table.

**Two-dimensional table**
A table whose values are based upon two independent variables

EXAMPLE \( C_\alpha(\alpha,\beta) \) may be represented by a two-dimensional table.

**Static equation**
A mathematical statement where the output (left hand side) does not have direct dependence (right hand side) on a simulation state

**Simulation states (and state derivatives)**
In the formulation of a non-linear simulation model shown as

\[ \dot{x} = f_1(x(t), u(t), w(t)) \]
\[ y = f_2(x(t), u(t)) \]

where
\( x \) represents a vector of the simulation states.
\( \dot{x} \) represents a vector of the simulation state derivatives.
\( u \) represents a vector of the simulation controls
\( y \) represents a vector of the simulation outputs
\( w \) represents a vector of disturbances

**Function Table**
The numeral set of data points used to represent the value of an independent variable as a function of the value(s) of one or more independent variables

EXAMPLE \( C_1(\alpha, \beta) \) may be represented by a function table.

**Gridded Table**
A multi-dimensional function table in which all independent variable breakpoints are constant across the function range, but not necessarily evenly spaced

NOTE 1 This is sometimes called an orthogonal table.

NOTE 2 All one-dimensional tables are gridded tables.

EXAMPLE A gridded two-dimensional function

**Ungridded Table**
A multi-dimensional function table in which the independent variable breakpoints need not be constant across the function range

NOTE This is sometimes called a non-orthogonal table.

EXAMPLE An ungridded two-dimensional function
5 Standard Simulation Coordinate Systems

5.1 Background / Philosophy


Coordinate system standards are also reflected in the variable naming convention. When applicable, the coordinate system is included in the variable name (see Section 6).

5.1.1 Coordinate System Conventions

In general, ANSI/AIAA R-004-1992 is the normative reference for coordinate system definitions. These coordinate systems are discussed in Table 1. However, it is important to emphasize the correlation of the AIAA document and the Distributed Interactive Simulation (DIS) coordinate systems. The geocentric Earth fixed coordinate system and body coordinate system are both used in DIS and High Level Architecture (HLA) simulations.

5.1.1.1 Geocentric Earth Fixed Coordinate System

The Geocentric Earth Fixed Coordinate System (coordinate system 1.1.3 of Table 1) is identical to the DIS “Geocentric Cartesian Coordinate System” (also referred to as “World Coordinate System” in the DIS).

All variables referenced to this coordinate system use “ge” as part of their name for the Geocentric Earth Fixed Coordinate System. This coordinate system is also frequently called “Earth-centered, Earth-fixed.”

5.1.1.2 Body Coordinate System

Another standard coordinate system is the Body Coordinate System (coordinate system number 1.1.7 in ANSI/AIAA R-004-1992). This is identical to the DIS “Entity Coordinates System.” The body coordinate system is identified in the variable names by “body”.

5.1.1.3 Additional Coordinate Systems

In addition to the coordinate systems defined in ANSI/AIAA R-004-1992, this standard has added the Moment Reference Center, Flat Earth, Locally Level, and Vehicle Reference (or Structural) coordinate systems. The Moment Reference Center coordinate system is a special case body coordinate system (number 1.1.7 in ANSI/AIAA R-004-1992, number 1.1.5 in ISO 1151-1:1998). The Flat Earth and Locally Level coordinate systems are respectively variants of the Normal Earth-fixed coordinate system (number 1.1.4 in ANSI/AIAA R-004-1992, number 1.1.2 in ISO 1151-1:1998) and the Vehicle-carried Earth coordinate system (number 1.1.6 in ANSI/AIAA R-004-1992, number 1.1.4 in ISO 1151-1:1998). Both of these coordinate systems are normally used in conjunction with an assumption that the Earth forms an inertial reference frame. They are used for simple simulations, and for creating vehicle model validation data. The Vehicle Reference coordinate system is a body coordinate system that may be used to locate vehicle components within the structure of the vehicle.

The moment reference center coordinate (MRC) system may be used to locate objects in the vehicle. Its axes are aligned with the body coordinate system, however, its origin is fixed at the moment reference center of the vehicle while the body coordinate system origin is at the center of mass (CM) and moves as the center of mass (CM) moves.

The moment reference center coordinate system is identified in the variable names by “mrc”.

The flat Earth coordinate system is based on a fixed, non-rotating, flat Earth with no mapping to a round Earth coordinate system, and therefore, latitude and longitude are inappropriate (but can be scaled for
small maps). The purpose of this coordinate system is to allow, if desired, vehicle checkout simulation to be performed in this coordinate system. This simplifies the use of this standard by simulation facilities that do not normally use a round or oblate spheroidal, rotating Earth model.

The flat Earth coordinate system is identified in the variable names by "ffe".

The locally level coordinate system is the reference coordinate system for angles and angular motion. Its origin is fixed at the vehicle center of mass.

The locally level coordinate system is identified in variable names by "lll".

The vehicle reference coordinate system is used to locate vehicle components. It is fixed to the vehicle structure and does not move. The specific definition differs for each vehicle. Sometimes the vehicle system may be the weight and balance reference system for the vehicle. The X origin is often in front of the vehicle, the Y origin in the centerline of the vehicle and the Z origin below the vehicle. The X-axis is often called the fuselage station and is often positive aft, the Y-axis is called butt line and is often positive to right, and the Z-axis is the waterline and is often positive up. However, these definitions may change with the vehicle and a manufacturing reference system may instead be used.

The vehicle reference system is identified in variable names by "vref".

### 5.2 Complete List of Coordinate Systems

The coordinate systems that are referenced are taken largely from paragraph 1.1 of ANSI/AIAA R-004-1992. The moment reference center, flat Earth and locally level coordinate systems for atmospheric flight simulation approximation are additional to that reference. A vehicle reference coordinate system is added for the purpose of locating systems and subsystems in the vehicle. Table 1 is the comprehensive list of coordinate systems that may be used under this standard.

The first column in Table 1 provides the abbreviation recommended for each coordinate system. The coordinate system may be referenced in a variable name by use of its abbreviation. See Section 6 on the variable naming convention.

<table>
<thead>
<tr>
<th>Reference Abbreviation</th>
<th>R-004-1992 Paragraph Number</th>
<th>Term</th>
<th>Definition</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>e1</td>
<td>1.1.1</td>
<td>Geocentric inertial coordinate system</td>
<td>An inertial reference system of the FK5 mean equator and equinox of J2000.0 has the origin at the center of the Earth, the X-axis being the continuation of the line from the center of the Earth through the center of the Sun and the vernal equinox, the Z-axis pointing in the direction of the mean equatorial plane's north pole, and the Y-axis completing the right-hand system. (See Figure 1A in R-004)</td>
<td>X_eY_eZ_e</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(for Earth centered inertial)</td>
<td>(See Appendix D.2 of R-004 for a modification of this system used for launch vehicles.)</td>
<td></td>
</tr>
<tr>
<td>Not used, this forms a basis for other definitions</td>
<td>1.1.2</td>
<td>Earth-fixed coordinate system</td>
<td>A right-hand coordinate system, fixed relative to and rotating with the Earth, with the origin and axes directions chosen as appropriate.</td>
<td>X_oY_oZ_o</td>
</tr>
<tr>
<td>g0</td>
<td>1.1.3</td>
<td>Geocentric Earth-fixed coordinate system</td>
<td>A system with both the origin and axes fixed relative to and rotating with the Earth (1.1.2). The origin is at the center of the Earth, the X-axis being the continuation of the line</td>
<td>X_oY_oZ_o</td>
</tr>
</tbody>
</table>

NESC Request No.: 09-00598
<table>
<thead>
<tr>
<th>Reference Abbreviation</th>
<th>R-004-1992 Paragraph Number</th>
<th>Term</th>
<th>Definition</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Earth-fixed</td>
<td></td>
<td>from the center of the Earth through the intersection of the Greenwich meridian and the equator, the ( z_0 )-axis being the mean spin axis of the Earth, positive to the north, and the ( y_0 )-axis completing the right-hand system. (See Appendix D.3 in R-004-1992)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.1.4</td>
<td></td>
<td>Normal Earth-fixed coordinate system</td>
<td>An Earth-fixed coordinate system (1.1.2) in which the ( z_0 )-axis is oriented according to the downward vertical passing through the origin (from the origin to the nadir). (See Figure 1C in R-004-1992)</td>
<td>( x_0 y_0 z_0 )</td>
</tr>
<tr>
<td>va</td>
<td>1.1.5</td>
<td>Vehicle-carried orbit-defined coordinate system( ^8 )</td>
<td>A system with the origin fixed in the vehicle, ( \textit{the default being the center of mass} ), in which the ( z_0 )-axis is directed from the spacecraft toward the nadir, the ( y_0 )-axis is normal to the orbit plane (positive to the right when looking in the direction of the spacecraft velocity), and the ( x_0 )-axis completes the right-hand system. (See Figure 1A in R-004-1992)</td>
<td>( x_0 y_0 z_0 )</td>
</tr>
<tr>
<td>ve</td>
<td>1.1.6</td>
<td>Vehicle-carried normal Earth coordinate system( ^8 )</td>
<td>A system in which each axis has the same direction as the corresponding normal Earth-fixed axis, with the origin fixed in the vehicle, ( \textit{the default being the center of mass} ).</td>
<td>( x_0 y_0 z_0 )</td>
</tr>
<tr>
<td>body</td>
<td>1.1.7</td>
<td>Body coordinate system( ^8 )</td>
<td>A system fixed in the vehicle, ( \textit{with the default origin being the center of mass} ), consisting of the following axes: An axis in the reference plane or, if the origin is outside that plane, in the plane through the origin parallel to the reference plane, and positive forward. In aircraft or missiles, this is normally from the CM forward towards the nose in the vertical plane of symmetry. It is also normally parallel to the waterline of the vehicle. An axis normal to the reference plane and positive to the right of the ( x )-axis (henceforth, positive to the right). An axis that lies in or parallel to the reference plane, whose positive direction is chosen to complete the orthogonal, right-hand system ( xyz ).</td>
<td>( x ), ( y ), ( z )</td>
</tr>
</tbody>
</table>

NESC Request No.: 09-00598
<table>
<thead>
<tr>
<th>Reference Abbreviation</th>
<th>R-004-1992 Paragraph Number</th>
<th>Term</th>
<th>Definition</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>mrc</td>
<td>none</td>
<td>A body coordinate system</td>
<td>This is a body coordinate system. The default origin is not fixed at the center of mass, but at the moment reference center (mrc) and therefore does not move. It consists of the following axes: An axis in the reference plane or, if the origin is outside that plane, in the plane through the origin parallel to the reference plane, and positive forward. In aircraft or missiles, this is normally from the MRC forward towards the nose in the vertical plane of symmetry. It is also normally parallel to the waterline of the vehicle. An axis normal to the reference plane and positive to the right. An axis that lies in or parallel to the reference plane, whose positive direction is chosen to complete the orthogonal, right-hand system $xyz$.</td>
<td>$\dot{x}_{mrc}$</td>
</tr>
<tr>
<td>wind</td>
<td>1.1.8</td>
<td>Air-path system$^8$</td>
<td>A vehicle carried system with the origin fixed in the vehicle, located at the center of mass, consisting of the following axes: An axis in the direction of the vehicle velocity relative to the air An axis normal to the air-path axis and positive to the right of the plane formed by the $x_s$ and $z_s$ axes. An axis in the reference plane or, if the origin is outside that plane, parallel to the reference plane, and normal to the air-path axis. The positive direction of the $z_s$ axis is chosen so as to complete the orthogonal, right-hand system $x_{s}\ y_{s}\ z_{s}$</td>
<td>$x_{a}$ $y_{a}$ $z_{a}$</td>
</tr>
<tr>
<td>ea</td>
<td>1.1.9</td>
<td>Intermediate coordinate system$^8$</td>
<td>A system with the origin fixed in the vehicle, located at the center of mass, consisting of the following axes.</td>
<td>$x_{s}\ y_{s}\ z_{s}$</td>
</tr>
<tr>
<td>Reference Abbreviation</td>
<td>R-004-1992 Paragraph Number</td>
<td>Term</td>
<td>Definition</td>
<td>Symbol</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------</td>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>x&lt;sub&gt;b&lt;/sub&gt;-axis</td>
<td></td>
<td>The projection of the air-path x axis on the reference plane, or, if the origin is outside that lane, on the plane through the origin, parallel to the reference plane.</td>
<td>x&lt;sub&gt;b&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>y&lt;sub&gt;b&lt;/sub&gt;-axis</td>
<td></td>
<td>An axis normal to the reference plane and positive to the right, coinciding with or parallel to the lateral axis (1.1.7).</td>
<td>y&lt;sub&gt;b&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>z&lt;sub&gt;b&lt;/sub&gt;-axis</td>
<td></td>
<td>An axis that coincides with or is parallel to the normal air-path axis so as to complete the orthogonal right-hand system.</td>
<td>z&lt;sub&gt;b&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>fp</td>
<td>1.1.10</td>
<td>Flight-path coordinate system&lt;sup&gt;a&lt;/sup&gt;</td>
<td>A system with the origin fixed in the vehicle (at the center of mass) and in which the x&lt;sub&gt;b&lt;/sub&gt;-axis is in the direction of the flight-path velocity relative to the Earth. The y&lt;sub&gt;b&lt;/sub&gt;-axis is normal to the plane of symmetry and positive to the right. The z&lt;sub&gt;b&lt;/sub&gt;-axis completes the orthogonal right-hand system.</td>
<td>x&lt;sub&gt;fp&lt;/sub&gt;, y&lt;sub&gt;fp&lt;/sub&gt;, z&lt;sub&gt;fp&lt;/sub&gt;</td>
</tr>
<tr>
<td>aa</td>
<td>1.1.11</td>
<td>Total-angle-of-attack coordinate system&lt;sup&gt;a&lt;/sup&gt; (USA practice: aeroballistic coordinate system.)</td>
<td>A system with the origin fixed in the vehicle, at the center of mass, in which the x&lt;sub&gt;a&lt;/sub&gt;-axis is coincident with the x&lt;sub&gt;b&lt;/sub&gt;-axis in the body coordinate system (1.1.7). The y&lt;sub&gt;a&lt;/sub&gt;-axis is perpendicular to the plane formed by the x&lt;sub&gt;a&lt;/sub&gt;-axis and the velocity vector, positive to the right. The z&lt;sub&gt;a&lt;/sub&gt;-axis is formed to complete the orthogonal, right-hand system.</td>
<td>x&lt;sub&gt;aa&lt;/sub&gt;</td>
</tr>
<tr>
<td>fe</td>
<td>None</td>
<td>Flat Earth system (not in R-004)</td>
<td>The Flat Earth coordinate system origin is situated on the Earth's surface directly under the center of mass of the vehicle at the initialization of the simulation. The x&lt;sub&gt;FE&lt;/sub&gt;-axis points northwards and the y&lt;sub&gt;FE&lt;/sub&gt;-axis points eastward, with the z&lt;sub&gt;FE&lt;/sub&gt;-axis down. The x&lt;sub&gt;FE&lt;/sub&gt; and y&lt;sub&gt;FE&lt;/sub&gt; axes are parallel to the plane of the flat Earth.</td>
<td>X&lt;sub&gt;FE&lt;/sub&gt;Y&lt;sub&gt;FE&lt;/sub&gt;Z&lt;sub&gt;FE&lt;/sub&gt;</td>
</tr>
<tr>
<td>ll</td>
<td>None</td>
<td>Locally Level coordinate system (not in R-004)</td>
<td>A vehicle related coordinate system (1.1.6) with the origin instantaneous at the ownership center of mass. The x&lt;sub&gt;LL&lt;/sub&gt;-axis passes through the vehicle center of mass and points towards the nadir. The x&lt;sub&gt;LL&lt;/sub&gt;-axis is parallel to the smooth surface of the Earth and oriented toward true north in the geometric Earth model. The y&lt;sub&gt;LL&lt;/sub&gt;-axis is</td>
<td>X&lt;sub&gt;LL&lt;/sub&gt;Y&lt;sub&gt;LL&lt;/sub&gt;Z&lt;sub&gt;LL&lt;/sub&gt;</td>
</tr>
</tbody>
</table>
Reference Abbreviation | R-004-1992 Paragraph Number | Term | Definition | Symbol
--- | --- | --- | --- | ---
VRS | None | Vehicle Reference system (not in R-004) | A vehicle fixed coordinate system used to locate items in the vehicle. It is often the weight and balance coordinate reference system for the vehicle, or the manufacturing coordinate reference system. The VRS may not be a right-handed coordinate system. X-axis is the longitudinal reference. It may be the Fuselage Station line, normally 0 being in front of the vehicle with the coordinate increasing aft. Y-axis is the lateral reference. It may be the Butt line perpendicular to the vertical symmetric plane of the vehicle and in the geometric center of the vehicle. Positive to the right facing forward (Starboard) Z-axis is the vertical reference. It may be the Waterline and its origin is normally under the vehicle, positive up. | \textit{By default, the origins of the coordinate systems selected from ANSI/AIAA-R-004-1992 1.1.5 through 1.1.11 coincide and are at the center of mass. If that is not the case, it is necessary to distinguish the different origins by appropriate suffixes and additional coordinate system references.}

\textit{The reference plane should be a plane of symmetry, or a clearly specified alternative. This may be specified by the vehicle reference system (VRS).}

\textit{Italics indicate clarifications to ANSI/AIAA-R-004-1992.}

5.3 Summary

This coordinate system standard should be followed for all future equations of motion. It is necessary for unambiguous reference to coordinate systems in simulation variable names.

5.4 References


moments and their coefficients, 01 April 1969.
6  Standard Simulation Variables

6.1  Background / Philosophy

6.1.1  Rationale for Having Standard Variable Name and Naming Conventions

The standard variable names and coordinate system definitions are part of this standard to facilitate communication. They provide a "common language" for information exchange. For example, to unambiguously exchange a function representing a lift coefficient, the minimum information required to be transmitted includes the independent variables used to define the function (such as angle-of-attack, angle-of-sideslip, Mach number, Reynolds number and aircraft configuration), their units, their sign conventions, and their reference coordinate systems. Such an exchange will be facilitated by using standard variable names and coordinate systems.

If a model uses standard variable nomenclature the information defining the model data may be exchanged entirely by reference to this standard. Additionally, adherence to the variable naming convention included herein will allow the list of standard variables to grow as needed by the user community. Use of the convention to maintain consistent variable names will ease user workload and maximize the benefits to be obtained from this standard.

Positions, angles, velocities and angular velocities referred to in this standard are defined in accordance with ANSI/AIAA R-004-1992.

6.2  Variable Naming Convention

The purpose of the naming convention is to provide guidance for the creation of variable names consistent with the standard variable names (Annex A). This will allow expansion of Annex A over time, further expanding the set of names available to facilitate model exchanges.

Variable names are constructed from components that jointly serve to fully define the variable in its particular application. A combined mixed case and underscore variable name convention is used. In variable component names that consist of multiple words, the first letter of each word is capitalized (medial capitals). Where the simulation language in use allows it, and where the logic of the simulation requires it, an underscore may be replaced by a period (.) to indicate an object member, or by parentheses or brackets to indicate an array member.

The following general rules for naming all variables shall be followed:

a) Variables shall have meaningful names.

b) Variable names shall not exceed 63 characters in length. Brief, but complete, names are most effective.

c) Names shall be constructed using US-ASCII 7-bit character encodings.

6.3  Variable Name Methodologies

There are three methods specified for defining variables consistent with this standard. Different methods are described for:

a) Position variables (linear or angular), arrays or structures

b) Motion variables (velocities, accelerations, or higher derivatives, both linear and angular), arrays or structures

c) All other variable names

The naming convention for position variables and for variables describing motion are different than other types of variables because of the general requirements to specify coordinate systems that uniquely define...
position and velocity.

In additional, the following guidelines for capitalization when creating variable names are provided:

d) The first letter in the variable name is lower case. Similarly, the first letter in the prefix and the first 
   component following the prefix are lower case.

e) The first letters in acronyms and abbreviations are capitalized.

f) Distinct components in variable names, after the first component, shall begin with a capital letter.

g) Units are not capitalized unless the unit abbreviation itself is.

6.3.1 The Physical Basis for the Position, Velocity, Acceleration (and Derivatives thereof) 
Naming Convention

To ground the discussion of variable names it is useful to refer to a standard dynamics text and to review 
the equation of Coriolis. Figure 2 below shows the derivation of the three-dimensional linear velocity of 
point $p$ with respect to coordinate system $\mathbb{M}$ as observed from coordinate system $\mathbb{O}$. Coordinate system $\mathbb{M}$ 
has translational and rotation motion relative to $\mathbb{O}$; in the figure, $\omega$ depicts the angular rate of $\mathbb{M}$ with 
respect to $\mathbb{O}$.

![Figure 2 — Rotating reference frames and relative geometry](image)

The velocity of $p$ with respect to $\mathbb{M}$ as observed from $\mathbb{O}$ is the vector difference of the velocity of $p$ with 
respect to $\mathbb{O}$ and the velocity of $\mathbb{M}$ with respect to $\mathbb{O}$ (both of which are observed from $\mathbb{O}$):

$$
^{\mathbb{O}}\dot{r}_{p/\mathbb{M}} = ^{\mathbb{O}}\dot{r}_{p/\mathbb{O}} - ^{\mathbb{O}}\dot{r}_{\mathbb{M}/\mathbb{O}}
$$

(1)

For this discussion, the vector notation used by Stevens and Lewis is used (see Figure 3 below) but other 
notations are equally valid. Using the equation of Coriolis (Stevens and Lewis equation 1.2-10), the 
velocity of $p$ with respect to $\mathbb{O}$ observed from $\mathbb{O}$ is given by:

$$
^{\mathbb{O}}\dot{r}_{p/\mathbb{O}} = ^{\mathbb{O}}\dot{r}_{\mathbb{M}/\mathbb{O}} + ^{\mathbb{M}}\dot{r}_{p/\mathbb{M}} + \omega_{\mathbb{M}/\mathbb{O}} \times r_{p/\mathbb{M}}
$$

(2)

The left-hand side term and the first right hand side term in equation (2) are the velocity terms in the right-
hand side of equation (1). Combining the two equations produces the following relationship:
Equation (3) shows that the velocity of \( p \) with respect to \( M \) as observed from \( O \) does not equal the velocity of \( p \) with respect to \( M \) as observed from \( M \) in the general case. This illustrates the need to clearly specify, for derivatives of a linear position vector, the coordinate systems that define the point (this standard designates the point on the vehicle by "Of"), the origin from which the position is measured (this standard designates this by "Wrt"), and the frame in which the observation is made ("ObsFr"). An additional challenge is to achieve this specification using just the ASCII set of characters that are used to compose variable names.

Figure 3 shows how the notation used by Stevens and Lewis are mapped into the coordinate system components of the variable name scheme. Within this standard, point \( p \) will be illustrative of the Of variable name component, coordinate system \( M \) will be illustrative of the Wrt variable name component and coordinate system \( O \) will be illustrative of the ObsFr variable name component.

Another coordinate system that must be specified is that into which the three vector components are resolved in (this standard uses the term 'presentation coordinate system'). This is indicated in the Figure 3 as the right superscript \( B \), which would indicate which coordinate system is used to resolve the vector into three scalar components.

In order to define position, velocity, acceleration or higher derivative variables (both translational and rotational), it is often necessary to specify each of these various coordinate systems. The kinematic requirements to clearly define these variables are presented below.

**Positions:** For positions (including rotational attitudes), the variable name must specify the origin from which the position is being measured. The name also must specify the coordinate system in which the vector is being resolved.

Take, for example, the core variable name

position

This name alone is meaningless. Therefore, it is necessary to describe what the position is representing:

positionOfPilotEye

This is still ambiguous as it necessary to specify both the point from which the pilot's eyepoint is being measured and into which coordinate system the position vector is being resolved.

positionOfPilotEyeWrtCm

This indicates what point is being located relative to what reference point, and therefore a relative position vector may be defined in 3-space. However, without knowing in which coordinate system the vector is being resolved, any number of coordinate systems could be used. Thus

bodyPositionOfPilotEyeWrtCm
This is a complete and unambiguous variable name representing a three-dimensional position. To define the name of any of the three body coordinate system axes, we need to append the name of the appropriate axis and units of measurement:

`bodyPositionOfPilotEyeWrtCm_ft_X`

This defines a scalar variable representing the X-body axis offset of the eyepoint from the center of mass, measured in feet.

**Attitudes:** Attitude (rotational position) is the orientation of one coordinate system relative to another; therefore, two coordinate systems are required (either explicit or implied) in the variable name to define an attitude. These axis systems are specified using the 'of' and 'wrt' components. The presentation coordinate system is not used because attitude (either as a quaternion, Euler angles, or rotation cosine matrix) is not a vector quantity resolved into X, Y, and Z components.

If the attitude is expressed as a set of Euler angles, the aeronautical convention of yaw-pitch-roll (3-2-1) rotation sequence is the default. To specify a different rotation sequence, the sequence should be appended to the Euler angle core name, e.g. eulerAngle313 for 3-1-3 rotations. To avoid confusion, for any rotation sequence other than 3-2-1, _first, _second, _third should be used for angle selectors in lieu of Roll, Pitch, Yaw.

For example:

```
eulerAngleOfIsoWrtEi1_rad_Pitch
```

This variable represents the pitch attitude (rotation about the Y axis) of the user-defined International Space Station (Iss) coordinate system relative to the Earth-centered inertial (e1) coordinate system, measured in radians. This rotation uses the default yaw-pitch-roll (3-2-1) rotation convention.

```
eulerAngle313OfIssWrtOrion_rad_Second
```

This variable represents the second rotation angle of the International Space Station (Iss) coordinate system relative to another user-defined (orion) spacecraft coordinate system measured in radians. This variable uses yaw-roll-yaw (3-1-3) rotation convention.

```
eulerAngleOfImuWrtBody_rad[[_Roll, _Pitch, _Yaw]
```

This variable represents the three Euler angles of a user-defined inertial measurement unit (Imu) coordinate system relative to the body coordinate system.

```
eulerAngleOfimuWrtimu2_rad[[_Roll, _Pitch, _Yaw]
```

This variable represents the three Euler angles of a user-defined inertial measurement unit (imu2) coordinate system relative to an Imu2 coordinate system, using the default yaw, pitch, roll (3-2-1) rotation convention.

**Derivatives of position (translational or rotational):** For variables representing derivatives of translational positions (velocities, accelerations and higher derivatives thereof) the observer coordinate system must be specified by the naming methodology. The observer coordinate system exists in the reference frame from which the movement (a velocity, acceleration or higher derivative) is observed (or measured). In many cases this is the same reference frame as the relative coordinate system (defined by the wrt component) used to specify the position of the object being observed, but in the most general
case may be a different frame. The magnitude and direction of velocity (and higher derivatives) varies with selection of the observer's coordinate system since the relative coordinate system may be moving relative to the observer's coordinate system.

For rotational derivatives, it is unnecessary to specify an observer's coordinate system, but both coordinate frames that describe the rotational derivatives are necessary (both cf and frt components).

For example:

<table>
<thead>
<tr>
<th>eiAngularRateOfIssWrtOrion_rad_s_Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>This variable represents the angular rate of the ISS coordinate system relative to the Orion spacecraft coordinate system. This is the angular velocity vector is resolved to the Earth-centered inertial (e1) coordinate system to measure the Y component in that system.</td>
</tr>
</tbody>
</table>

6.3.2 New Position Variables Naming Convention

The methodology for creating and defining a position variable (linear or angular) that is consistent with the requirements of this standard are as follows.

a) Each variable name may have up to nine components.

b) With the exception of the core name, all components are optional and should only be used if required by the application. Units must be specified unless the variable is non-dimensional and then the _nd units specification is encouraged.

The variable name components are listed immediately below. Descriptions of all the components follow in this section.

1. `<variable domain>`
2. `_<dynamic equation formulation prefix>_`
3. `<presentation coordinate system>`
4. `<core name>` — the only required component
5. of `<a point for positions or coordinate system for angles, normally on the vehicle>`
   If omitted in the variable name, of defaults to the Cm for translational position and the body coordinate system for rotational position.
6. wrt `<With respect to a point or coordinate system>`
   If omitted in the variable name, wrt defaults to the presentation coordinate system for linear positions and to the locally level coordinate system (11) for angular position.
7. ic <initial condition designation>
8. `_<units>`
9. `<specific axis of the presentation coordinate system>`

Rarely are all 9 components of a name used.

For example:
6.3.3 New Velocity, Acceleration or Higher Derivative Motion Variables Naming Convention

The methodology for creating and defining velocity and acceleration variables (or higher derivatives) consistent with the requirements of this standard are as follows:

a) Each variable name may have up to ten components.

b) With the exception of the core name, all components are optional and should only be used if required by the application. Units must be specified unless the variable is non-dimensional and then the _und units specification is encouraged.

The variable name components are listed immediately below. Descriptions of all the components follow in this section.

1. <variable domain>
2. _<dynamic equation formulation prefix>_
3. <presentation coordinate system>
4. <core name> — the only required component
5. _@<point for translation or coordinate system for rotation, normally on the vehicle>
   If omitted, defaults to the Cm for translation derivatives and the body coordinate system (body) for rotational derivatives.
6. _wrt<"With respect to" a point, frame or coordinate system>
   The wrt component (relative coordinate system) may be omitted; if so, the relative coordinate system defaults to the presentation coordinate system for translational variables and the locally-level coordinate (l1) system for rotational variables.
7. _ObsFr<"Observed From" coordinate system, only used for translational motion>
   If "ObsFr" is not specified, it defaults to the same coordinate system specified by the relative coordinate system (the wrt component). If the relative coordinate system is not present, ObsFr defaults to the presentation coordinate system.

8. _ic — initial condition designation
9. _<units>
10. _<specific axis of the presentation coordinate system>

Rarely are all 10 components of a name used.

For example:
6.3.4 New General Variables Naming Convention

The methodology for defining variables other than positions and derivatives thereof that is consistent with the requirements of this standard are as follows.

a) Each variable name may have up to seven components.

b) With the exception of the core name, all components are optional and should only be used if required by the application. Units must be specified unless the variable is non-dimensional and then the _nd units specification is encouraged.

The variable name components are listed immediately below. Descriptions of all the components follow in this section.

1. `<variable domain>`
2. `<dynamic equation formulation prefix>`
3. `<presentation coordinate system>`
4. `<core name>` — the only required component
5. `Io` — initial condition designation
6. `<units>`
7. `<specific axis of the presentation coordinate system>`
6.3.5 Adapting the Naming Convention to Hierarchical and Nested Data Representations

The naming methodology provides all the essential information about a variable in its name. This convention is concise for flat data representations (e.g., single-data variables, arrays, common blocks). However, the convention can lead to repetition of information if applied to the members of hierarchical and nested data structures (e.g., classes, structures, records) since structure organization might parallel one or more of the variable name components. For example, a developer could create a structure to hold all of the variables in an aerodynamics model. The developer could then declare an instance of the structure with the name “aero.” The structure name would repeat information in the variable source domain component of its member variables.

An example of a “flat” variable name is:

\[ \text{aero\_bodyForceCoefficient\_X} \]

Prepending the name of the structure to the standard variable name could result in an expression like the one below:

\[ \text{aero\_aero\_bodyForceCoefficient\_X} \]

Such repetition can lead to unnecessarily long expressions. To avoid such repetition, developers may use the following guidelines to adapt the naming convention to hierarchical and nested data structures. First, when a level of a structure represents an organization of data that is equivalent to a variable name component, the developer should use the naming rules for that component to name instances of that structure level. Second, if a component of the variable name appears at a higher level or lower level, the developer should not include that component in the variable names at the current level. Using these guidelines, the example above can be changed to:

\[ \text{aero\_bodyForceCoefficient\_X} \]

If the developer made a further change to represent a vector as a structure and replaced the X, Y, and Z variables for the body force coefficient with that structure, the above expression would change to:

\[ \text{aero\_bodyForceCoefficient\_X} \]

To meet the intent of the guidelines, it is not necessary that the structure levels address the variable name components in the same order that the convention specifies for variable names. The intent of the naming convention is to unambiguously identify the information represented by a variable; it is not intended to shape data design. Thus, the name components can appear in a different order for a hierarchical or nested data expression. For example:

\[ \text{bodyForce\_int\_aero\_X} \]

In this example, the data design is such that all the external force on a vehicle (expressed in body coordinates and in units of pound-force) are collected in a structure whose instance is named “bodyForce” and whose members represent each generator of force as an instance of a structure representing a vector. The variable source domain appears after the core name and units due to the chosen data design. Even so, this data expression unambiguously identifies the variable as effectively as the equivalent scalar variable name, \text{aero\_bodyForce\_1bf\_X}. 
6.4 Components Used to Create Variable Names

6.4.1 Variable Domain Component

This represents the domain in which the variable is calculated. In object-oriented design, it could logically be the object. The domain is normally not included if it (or the object) is the vehicle or aircraft being simulated, for example, airspeed.

In some cases the domain name component only provides background information when exchanging models. For example, in one simulation architecture the domain for ambientPressure_N_m2 might be "environment" and another architecture the domain might be 'atmosphere'. The core component of the name is the key. For example:

\[
\text{environment\_ambient\_Pressure\_N\_m2} \quad \text{in one simulation architecture is identical to} \\
\text{atmosphere\_ambient\_Pressure\_N\_m2} \quad \text{in an another simulation architecture.}
\]

However in some cases the domain component is critical. For example:

\[
\text{aero\_body\_force\_lbf\_X} \quad \text{and} \quad \text{thrust\_body\_force\_lbf\_X} \quad \text{are two different variables, both are body axis forces but one comes from the propulsion system model and one from the aerodynamic model. It is this type of variable where domain must be included.}
\]

Some domain examples are presented in Table 2. The domain names presented here are not part of any standard; instead they are presented here as examples.

Table 2 — Examples of domain names

<table>
<thead>
<tr>
<th>domain</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aero</td>
<td>aerodynamic models</td>
</tr>
<tr>
<td>airLaunchedWeapon</td>
<td>modeling of munitions launched into the air that have their own dynamics, includes Missile as a sub-domain</td>
</tr>
<tr>
<td>cautionAndWarning</td>
<td>caution and warning simulation</td>
</tr>
<tr>
<td>cockpit</td>
<td>input/output from/to cockpit instruments and controls</td>
</tr>
<tr>
<td>controlLaw</td>
<td>simulation of a control algorithm</td>
</tr>
<tr>
<td>controlLoading</td>
<td>models of the control system feel</td>
</tr>
<tr>
<td>controlSurface</td>
<td>simulation of an aerodynamic control effector</td>
</tr>
<tr>
<td>controlSystem</td>
<td>collective model of control laws and control effectors on a vehicle</td>
</tr>
<tr>
<td>electrical</td>
<td>models of the electrical system</td>
</tr>
<tr>
<td>engine (or thrust or propulsion)</td>
<td>thrust generation models</td>
</tr>
<tr>
<td>environment</td>
<td>atmospheric models (ambient properties, wind, clouds, etc.)</td>
</tr>
<tr>
<td>failureSystem</td>
<td>failure modeling and fault injection</td>
</tr>
<tr>
<td>fltDirect</td>
<td>flight director models</td>
</tr>
<tr>
<td>fuelSystem</td>
<td>fuel system models</td>
</tr>
<tr>
<td>gun</td>
<td>model of vehicle mounted guns</td>
</tr>
<tr>
<td>hydraulics</td>
<td>hydraulic system models</td>
</tr>
<tr>
<td>Term</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>landingGear</td>
<td>landing gear models</td>
</tr>
<tr>
<td>massProperties</td>
<td>tree-based modeling of vehicle mass and moments of inertia</td>
</tr>
<tr>
<td>missile</td>
<td>missile models. Domain could be more specific, for example missileAim9x.</td>
</tr>
<tr>
<td>motion</td>
<td>motion system models and algorithms</td>
</tr>
<tr>
<td>navigationDatabase</td>
<td>mapping of waypoints, airports, runways, legs, procedures, and navigation transmitters (all of which are subdomains)</td>
</tr>
<tr>
<td>navigationReceiver</td>
<td>modeling of signal-based navigation sensors</td>
</tr>
<tr>
<td>navigationTransmitter</td>
<td>modeling of navigation signal generators (e.g. radios, GPS)</td>
</tr>
<tr>
<td>parachute</td>
<td>parachute models</td>
</tr>
<tr>
<td>propulsion</td>
<td>models the collection of thrust generators (engines) on a vehicle</td>
</tr>
<tr>
<td>radar</td>
<td>models the radar system. Domain could be more specific, for example radarApq99</td>
</tr>
<tr>
<td>relGeom</td>
<td>relative state (position, velocity, and acceleration) of each vehicle to each other vehicle</td>
</tr>
<tr>
<td>sensor</td>
<td>models of sensors</td>
</tr>
<tr>
<td>sensorSystem</td>
<td>modeling the collection of sensors on a vehicle</td>
</tr>
<tr>
<td>sim</td>
<td>Domain encompassing control of the simulation, configuration of a simulation run, control of mathematical techniques such as integration type, etc.</td>
</tr>
<tr>
<td>vehicle</td>
<td>modeling of the vehicle as a cooperating system of other domain models</td>
</tr>
<tr>
<td>weaponSystem</td>
<td>collective model of guns and air-launched weapons on a vehicle</td>
</tr>
<tr>
<td>wheel</td>
<td>landing gear wheel models</td>
</tr>
<tr>
<td>world</td>
<td>world model (shape, dynamics, and reference time(s) plus navigation database and environment domains)</td>
</tr>
<tr>
<td>universe</td>
<td>domain encompassing world, vehicle, and relative geometry domains</td>
</tr>
</tbody>
</table>

NOTE: Users may add as many domains as needed to clearly identify the variable.

Variable name examples using “aero” and “thrust” include:

a) aero_bodyForce_lbf_X
b) thrust_bodyForce_lbf_X
c) aero_bodyForceCoefficient_X
d) thrust.bodyForceCoefficient_X — this is an example of thrust as a structure.

e) thrust.bodyForceCoefficient(X)

### 6.4.2 Dynamic Equation Formulation Prefix Component

The dynamic equation formulation prefix is used to identify the most important dynamic variables in the simulation, the states ($x$) and their derivatives ($\dot{x}$), inputs ($u$), outputs ($y$), and disturbances ($w$) as presented in the equation below. These variables characterize the resultant dynamic response of a vehicle as shown in the equations below. In addition to these variables, the standard allows the prefix to separately designate simulation control variables ($o$). Simulation control variables are used to modify the behavior of the model during simulation and are not part of the vehicle model, while inputs ($u$) are variables that represent the inputs to the vehicle model which may include pilot control positions. Finally, in simulation or analysis where noise and environmental disturbances are modeled, the disturbances ($w$) are the final component in the simulation of the total system dynamics.

\[
\begin{align*}
\dot{x} &= f_1(x(t), u(t), w(t)) \\
y &= f_2(x(t), u(t))
\end{align*}
\]

The prefix shall be separated from the body of the variable by an underscore (_) and from the domain name by an underscore (or a period if preceded by a member of a structure or class). The leading underscore is not permitted if a domain name is not present.

#### 6.4.2.1 Identification of Simulation Model States and State Derivatives

The states ($x$) and state derivatives ($\dot{x}$) are those variables that make the simulation dynamic and are the key variables in a flight simulation model. Basically, any variable that is mathematically integrated is a state derivative. The result of integration of a state derivative over a period of time is a change in the value of the corresponding state over that time. This is true for any integration in a simulation. If the user controls the changes in all the states, they control the trajectory of the simulated model. The time histories of the states and inputs are the key variables required for validation. All outputs are computed directly or indirectly from states and inputs.

The formulation of the equations of motion and the model itself determines what variables are states. This naming convention is not meant to standardize on any variable as a state, but allows the simulation engineer to explicitly identify states in the model implementation, making it easier to document and exchange the models.

Examples:

- x_bodyVelocityWrtEi_ft_s_x
- dx_bodyAccelerationWrtEi_ft_s2_x

- x_ prefix indicates that this variable is a state
- dx_ prefix indicates that this variable is a state derivative

#### 6.4.2.2 Identification of Simulation Model Inputs

The simulation model inputs ($u$) are those variables that provide the pilot or autopilot inputs to the vehicle model. These are also called controls in many references. As with the states and state derivatives, the model inputs are key variables for validation. All model outputs are computed directly or indirectly from model states and inputs.

The formulation of the model itself determines which variables are inputs. This naming convention is not meant to standardize on any variable as an input, but allows the simulation engineer to explicitly identify
them, making it easier to document models, exchange them, and verify them.
Examples:

<table>
<thead>
<tr>
<th>u_controlSurfacePos_deg_avgAileron</th>
<th>u_ prefix indicates that this is a model input</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlLaw_u_controlSurfacePos_deg_avgAileron</td>
<td>same variable name with domain prefix</td>
</tr>
<tr>
<td>controlLaw.u_controlSurfacePos_deg_avgAileron</td>
<td>same variable name in a hierarchical architecture</td>
</tr>
<tr>
<td>u_pilotControlPos_rad_long</td>
<td>Another example of the longitudinal pilot input</td>
</tr>
</tbody>
</table>

6.4.2.3 Identification of Simulation Model Disturbances

The disturbances (\(w\)) are those variables that provide environmental disturbances or system noise to the simulation models.

Disturbances may be inserted into the vehicle model, environment, or equations of motion, depending upon implementation schemes. This naming convention is not meant to standardize on any variable as a disturbance, but allows the simulation engineer to explicitly identify disturbances, making it easier to document models, exchange them, and verify them.

Examples:

<table>
<thead>
<tr>
<th>w_bodyAngularRateTurbulenceWrtGo_deg_s_Yaw</th>
<th>w_ prefix indicates that this variable is a disturbance</th>
</tr>
</thead>
<tbody>
<tr>
<td>environment_w_bodyAngularRateTurbulenceWrtGo_deg_s_Yaw</td>
<td>same variable name with domain (environment) added</td>
</tr>
<tr>
<td>environment.w_bodyAngularRateTurbulenceWrtGo_deg_s_Yaw</td>
<td>same variable name in a hierarchical architecture</td>
</tr>
</tbody>
</table>

6.4.2.4 Identification of Simulation Model Outputs

The simulation model outputs (\(y\)) are those variables that are the outputs of the physics of the simulation models as formulated by the state equations. This is meant to assist in the specification of the state equations, mainly to help simplify model exchanges between simulations used for analysis and those used for real-time man in the loop or hardware in the loop simulations.

Example:

| y_leftHorizontalActuatorRamPosition_deg     | y_ prefix indicates that this variable is a model output |

NESC Request No.: 09-00598
6.4.2.5 Identification of Simulation Controls

The simulation controls (c) are those variables that provide the simulation operator control of the simulation [not to be confused with simulation model inputs (u)]. The simulation controls should not affect any vehicle states, state derivatives or outputs.

The software and hardware architecture of the simulation determines what variables are simulation controls. This naming convention is not meant to standardize on any variable as an control, but allows the simulation engineer to explicitly identify simulation controls. Clear definition of simulation controls makes validation of a simulation much easier after a model is exchanged.

Examples:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>c_simDuration_s</td>
<td>c_prefix indicates that this variable is a simulation control</td>
</tr>
<tr>
<td>cDeltaTime_s</td>
<td>another example</td>
</tr>
<tr>
<td>sim_cDeltaTime_s</td>
<td>same variable name with domain added</td>
</tr>
<tr>
<td>sim_cDeltaTime_s</td>
<td>same variable name in a hierarchical architecture</td>
</tr>
</tbody>
</table>

6.4.3 Presentation Coordinate System Component

This is the coordinate or reference system to which the variable is referenced or in which it is measured (it is indicated by the 'B' in Figure 3). Table 1 specifies the standard coordinate system abbreviations that should be used. If no coordinate system pertains to the variable or the core variable name needs no reference system to be unambiguous (e.g. Airspeed), this part of the variable name may be omitted.

6.4.3.1 Conventions Used

Earth fixed and local coordinate systems by convention use X, Y, Z, for both translation and (X, Y, Z), (Pitch, Roll, and Yaw), or (First, Second, Third) for rotation axis indices. The origin and attitude of local coordinate systems (flat Earth for example) may be user defined (such as N, E, D). Local coordinate systems are meant for runway, test range, target reference, navigational aids, etc.

6.4.3.2 Variable Name Examples

The following variable names are provided as examples.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_bodyAngularRate_rad_s_Roll</td>
<td>These are all equivalent variable names where body is the coordinate system and roll is the specific axis in the body axis system, roll indicating angular motion about the longitudinal axis relative to the locally-level frame (wrt defaults to locally-level for rotational variables). NOTE In this example the variable is designated as a state.</td>
</tr>
<tr>
<td>x_bodyAngularRate_rad_s_Roll</td>
<td>These represent translational inertial velocity in the body coordinate system along the longitudinal axis (the translational variable analogous to the rotational variable above). These are all equivalent variable names.</td>
</tr>
<tr>
<td>x_bodyAngularRate_rad_s_X</td>
<td></td>
</tr>
<tr>
<td>x_bodyVelocityWrtEl_m_s_X</td>
<td></td>
</tr>
<tr>
<td>Variable / Expression</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>geVelocity_m_s_Y</td>
<td>This represents a translational velocity with ( g ) specified as the coordinate system and ( Y ) as the specific axis. This non-inertial velocity of the CM is relative to and measured in the geocentric earth-fixed (ge) coordinate system.</td>
</tr>
</tbody>
</table>
| bodyTurbulenceVelocityWrtGe_ft_s_Z  
bodyTurbulenceVelocityWrtGe_ft_s [3]  
bodyTurbulenceVelocityWrtGe_ft_s.Z | These are all equivalent variable names where \( \text{body} \) is the coordinate system and \( Z \) is the specific axis in the body coordinate system, \( z \) indicating vertical translational motion. Also illustrated as a vector and structure. |
| runway22VelocityOfLeftWheelWrtTd_ft_s_Z | Here \( \text{runway22} \) is the coordinate system (user defined) and \( Z \) is the specific axis, also indicating translational motion. \( \text{LeftWheel} \) is the point on the vehicle and \( \text{Td} \) (touchdown point) is the reference point. |
| bodyAccelOfPilotEyeWrtEi_m_s2_Y | Here \( \text{body} \) is the coordinate system and \( Y \) is the specific axis, also indicating translational motion. Design pilot eyepoint is the point on the vehicle. |
| x_eiVelocity_ft_s_X     | This is a case where the equations of motion are formulated such that the variable is a state, resolved in the earth centered inertial (ei) coordinate system. |
| eiVelocity_ft_s_X       | This is a case where the equations of motion are formulated such that the variable is not a state. |
| x_llVelocity_ft_s_X    | Locally Level coordinate system |
| x_feVelocity_ft_s_X    | Flat Earth coordinate system |
| bodyAngularRate_rad_s_Pitch | |
| bodyAngularRate_rad_s_Roll | |
| bodyAngularAccel_rad_s2_yaw | |

Note that the standard allows \((X, Y, Z)\) or \((\text{Roll}, \text{Pitch}, \text{Yaw})\) or \((\text{First}, \text{Second}, \text{Third})\) as selectors for rotational positions and derivatives, since that is widely conventional. However, since the overall objective of the standard is to form a framework for clear communication between simulation facilities, the use of \(X, Y, Z\) selectors is acceptable. The appropriate core variable name shall be used to indicate whether the variable is a translational or rotational variable.

### 6.4.4 Core Variable Name Component

This is the most specific (hence core) name for the variable. All variable names shall include this component of the name.

Core variable name examples are as follows.

- \textit{velocity}\hspace{1cm} convention for velocities
- \textit{angularRate}\hspace{1cm} convention for angular rates
Flight Simulation Model Exchange

- accel
- angularAccel
- pilotControlPos
- pilotControlRate
- pilotControlAccel
- pilotControlForce
- copilotControlPos
- copilotControlRate
- copilotControlAccel
- copilotControlForce
- controlSurfacePos
- controlSurfaceRate
- controlSurfaceAccel
- controlSurfaceHingeMoment
- liftCoefficient
- dragCoefficient
- forceCoefficient
- turbulenceVelocity
- angleOfAttack
- angleOfSideslip
- cosineOfAngleOfSideslip
- thrust
- torque

The following extended variable names are provided as examples.

- x_bodyAngularRate_rad_s_Roll
- bodyTurbulenceVelocityWrtGe_ft_s_X
- geVelocity_ft_s_Z
- geVelocity_m_s_Z
- pilotControlPos_deg_long
- pilotControlPos_deg_lat
- pilotControlRate_deg_s_pedal
- pilotControlAccel_deg_s2_long
copilotControlPos_deg_long
copeilotControlPos_deg_lat
copilotControlRate_deg_s_long
copilotControlAccel_deg_s2_long
controlSurfacePos_deg_elevator[number of surfaces]
controlSurfaceRate_deg_s2_aileron[number of surfaces]
controlSurfaceAccel_deg_s2_aileron[number of surfaces]
controlSurfaceHingeMoment_ftlb_fcanard[number of surfaces]
angleOfAttack_rad
angleOfSideslip_deg
cosineOfAngleOfSideslip
controlSurfacePos_deg_aileron
totalPressure_N_m2
ambientPressure_N_m2
totalLiftCoefficient
aerodynamicForceCoefficient
aerodynamicForce_lbf
aerodynamicForce_N
thrustBodyForce_N

6.4.5 Reference Point or Coordinate System ("of")

This component of the name is designed to clarify positions, velocities and accelerations and is normally omitted if the variable is not a position, velocity or acceleration. However, it may be used for any variable if desired. This component describes which point or object that is being specified. "of" is used to specify the point or object (this is point p in Figure 2).

For those who prefer shorter variable names, the standard adopts the convention that if the point or location on the vehicle is the center of mass for translational motion variables, then the reference point may be omitted. For rotational motion, the default reference coordinate system is the body axis coordinate system.

Reference points may be defined by the user and depend on the object the variable is describing.

Examples of reference points are as follows.
- Ocfm: the center of mass is the default point, so "ofcm" is normally omitted in any variable name.
- Ofimu: OOfimu1, OOfimu2, OOfimuLtn200, etc.
- OfSensor: OOfSradar, OOfFlir, OOfSradarApG57, etc.
- OfMc: for moment reference center

NESC Request No.: 09-00598
Flight Simulation Model Exchange

- `OfPilotEye` for the pilot eye point
- `OfRadAlt` for radar altimeter
- `OfTerrain` a normal Earth-fixed coordinate system with origin where the vehicle nadir intersects the terrain

The following variable names are provided as examples.

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bodyPositionOfImuWrtCm_m[3]</code></td>
<td>This three-element vector locates the imu relative to the CM in the body axis coordinate system. Note that [3] indicates (for this example) that the referenced variable is a three-element vector.</td>
</tr>
<tr>
<td><code>bodyPositionWrtImu_m[3]</code></td>
<td>Both of these vector names refer to the same quantity; it is the opposite of the vector above (they locate the CM relative to the imu). In the first name &quot;ofcm&quot; is omitted since it is default.</td>
</tr>
<tr>
<td><code>bodyPositionOfCmWrtImu_m[3]</code></td>
<td>This is the angular equivalent of the first variable above. The 3-2-1 rotation convention is implied.</td>
</tr>
<tr>
<td><code>eulerAngleOfImuWrtBody_rad[3]</code></td>
<td>Here element 1 would be about the X-axis (roll), element 2 would be about the Y-axis (pitch) and element 3 would be about the Z-axis (yaw). These are inertial rates since they are measured with respect to the Earth inertial (EI) coordinate system.</td>
</tr>
<tr>
<td><code>x_bodyAngularRateWrtEi_rad_s[3]</code></td>
<td>OfCm is implied</td>
</tr>
<tr>
<td><code>bodyVelocityWrtAir_ft_s_X</code></td>
<td>Same meaning as above</td>
</tr>
<tr>
<td><code>bodyVelocityOfCmWrtAir_ft_s_X</code></td>
<td>Inertial velocity of the CM along the X-body axis</td>
</tr>
<tr>
<td><code>bodyVelocityWrtEi_ft_s_X</code></td>
<td>Inertial velocity of the CM along the X-body axis in SI units</td>
</tr>
<tr>
<td><code>bodyVelocityWrtEi_m_s_X</code></td>
<td>OfCm may be omitted since it is the default</td>
</tr>
<tr>
<td><code>heightOfCmWrtTerrain_ft</code></td>
<td>Height of nadir intersection with terrain above the reference ellipsoid</td>
</tr>
<tr>
<td><code>heightOfRadAltWrtTerrain_ft</code></td>
<td>Height of nadir intersection with terrain above the reference ellipsoid</td>
</tr>
<tr>
<td><code>heightOfTerrainWrtNgs84_ft</code></td>
<td>Height of nadir intersection with terrain above the reference ellipsoid</td>
</tr>
<tr>
<td><code>bodyPositionOfPilotEyeWrtCm_ft_X</code></td>
<td>These are the same scalar quantity.</td>
</tr>
<tr>
<td><code>getLongitudeOfImu_deg_s</code></td>
<td>Inertial acceleration vector in the body axis</td>
</tr>
<tr>
<td><code>longitudeOfImuWrtGe_deg</code></td>
<td>Inertial acceleration vector in the body axis</td>
</tr>
</tbody>
</table>

6.4.6 Component Indicating Relative Reference Point or Relative Reference Coordinate System

The relative reference component is generally used in conjunction with the "reference point or location on the vehicle" component described in section 6.4.5. It is primarily used in variables describing position,
velocities and accelerations. This component defines the reference that the motion or position is relative to. This component, preceded by "Wrt" (with respect to) in the variable name, is the equivalent of coordinate system M in Figure 2, as noted in Figure 3.

For position variables, Wrt refers to the reference point for linear positions. For angular positions, Wrt refers to the relative coordinate system. For derivatives of position (velocities, accelerations, etc.) Wrt is used to define relative motion of two objects.

If "Wrt" is omitted then the default points or relative coordinate systems are:

- the presentation coordinate system for linear positions and translational motion. For example, `bodyPositionOfImu_m[3]`. Here WrtBody is implied.
- the locally level coordinate system specified for rotational variables. For example, `eulerAngleOfImu_rad[3]`. Here "WrtLL" is implied.

Note: Since for translational motion the "Wrt" defaults to the presentation coordinate system the variable:

- `bodyVelocity_f_s_X`

while a valid variable, has little usefulness because, fully enumerated is:

- `bodyVelocityOfCmWrtBodyObsFrBody_f_s_X`

Body velocity of the Cm with respect to the body is virtually always near zero. It would only represent the movement of the Cm within the body, due to cargo shift, fuel burn, etc. It would not represent velocity of the Cm with respect to a coordinate system outside the aircraft.

Examples of reference points are as follows:

- **WrtCm**: this is commonly used to clarify definitions of positions within the vehicle
- **WrtB1**: identifies a variable that is referenced to inertial space
- **WrtHrc**: moment reference center
- **WrtTgt**: aim point
- **WrtImpact**: the desired weapon impact point
- **WrtAir**: the local atmosphere, used to define air-relative (or wind-relative) motion
- **WrtMeanSL**: mean sea level
- **WrtGe**: the geocentric Earth fixed coordinate system
- **WrtGround**: a normal Earth-fixed coordinate system with origin where the vehicle nadir intersects the smooth surface of the Earth
- **WrtTerrain**: a normal Earth-fixed coordinate system with origin where the vehicle nadir intersects the terrain

The following linear and angular position variable names are provided as examples:

| bodyPositionOfImuWrtCm_m[3] | This vector locates the IMU relative to the CM in the body axis coordinate system. WrtCm may be omitted since CM is the default reference point for linear position measurements. |

NESC Request No.: 09-00598
Flight Simulation Model Exchange

6.4.7 Component Indicating Observer's Coordinate System (Vehicle translational motion variables only)

For variables representing derivatives of linear positions (velocities, accelerations and higher derivatives thereof), the observer's coordinate system (indicated by ObsFr for "Observed From") must be specified. The observer's coordinate system is in that reference frame from which the movement (a velocity, acceleration or higher derivative) is observed or measured. In many cases this is the same reference frame as the relative coordinate system (given by Wrt) but in the most general case it may be a different frame. The magnitude and direction of velocity (and higher translational motion derivatives) differ depending on the motion of the observer's coordinate system. This is the coordinate system shown as the upper left superscript in the Stevens and Lewis convention shown in Figure 3 and is illustrated by coordinate system O in Figure 2.

It is conventional to omit the observer's coordinate system when it is the same as the reference (Wrt) coordinate system. As noted in Section 6.4.6, when the Wrt coordinate system is omitted it defaults to the presentation coordinate system for translational variables, so when both the Wrt (reference) and the observer's (ObsFr) coordinate systems are omitted, the default observer's coordinate system is the presentation coordinate system.

It is neither necessary nor appropriate to specify the observer's coordinate system for rotational motion variables as rotations are invariant with the location of the observer.

Some examples are:

<table>
<thead>
<tr>
<th>geVelocity_ft_e_X</th>
</tr>
</thead>
<tbody>
<tr>
<td>Velocity of the ownship center-of-mass in the geocentric Earth coordinate system (velocity along the GE X-axis). Note that the reference point (of), relative coordinate system (Wrt) and observer's coordinate system (ObsFr) default to the velocity of the center of mass with respect to and</td>
</tr>
</tbody>
</table>
observed from the geocentric Earth-fixed coordinate system since they are omitted.

`geVelocityOfComWrtGeoObsFrGe_ft_s_X`

This is the same variable as the previous one with the reference point (ofCom), relative coordinate system (WrtGeo) and observer coordinate system (ObsFrGe) all explicitly specified (they are not required since they are the defaults).

`bodyAccelWrtBi_ft_s2_X`

Acceleration of the vehicle center of mass relative to and observed from the Earth-fixed inertial (EI) coordinate system and presented in the body axis coordinate system (acceleration along the body X-axis). Naming convention components that can be implied are omitted.

`bodyAccelOfCmWrtEiObsFrEi_ft_s2_X`

This is the same variable as the previous one with the reference point, relative coordinate system and observer coordinate system all explicitly specified.

`bodyVelocityWrtAir_ft_e_X`

This is the air-relative velocity of the CM expressed in body coordinates, the ObsFr component is omitted to indicate that the observer's coordinate system is in the same frame as the steady state air mass reference frame (Air).

`bodyVelocityOfCmWrtAirObsFrAir_ft_e_X`

Same as the variable above, fully expressed.

`bodyPositionOfPilotEyeWrtCm_ft_X`

The position of the pilot's eyepoint relative to the vehicle center-of-mass along the body X-axis. Note that the sign convention is clear: since the X-axis origin is at the center of mass, and is positive forward, the pilot's eyepoint position is positive when forward of the center of mass.

`bodyPositionOfPilotEyeWrtMrc_ft_X`

The position of the pilot's eyepoint relative to the reference center along the body X-axis. Note that the sign convention is clear: since the X-axis origin is at the center of mass, and is positive forward, the pilot's eyepoint position is positive when forward of the MRC.

`bodyAccelOfPilotWrtBi_ft_s2_[3]`

This represents the inertial acceleration of the pilot, resolved into the vehicle's body axes. The ObsFr component is omitted, implying the motion is observed from the Wrt coordinate system (here, Earth Inertial).

`bodyAccelOfPilotWrtBiObsFrEi_ft_s2_[3]`

Same as above. ObsFr explicitly stated.

`bodyVelocityWrtBi_ft_e_X`

Inertial velocity of the CM along the X-body axis.

`bodyVelocityOfCmWrtBiObsFrEi_ft_e_X`

Same meaning as the previous variable, fully expressed.

`runway22VelocityOfLeftWheelWrtTdObsPrTd_ft_e_e`

This is the velocity of the wheel relative to the TD point observed from the TD coordinate system. The user must explicitly define the TD coordinate system, but logically it is in an Earth-fixed reference frame, probably with the origin at the desired touchdown point and aligned with the
runway.
runway22VelocityOfLeftWheelWrtTd_ft_s_Z
This is the same variable as above, since omitting ObFr implies it is the Wrt coordinate system.
velocityWrtGround_ft_s
This scalar variable is commonly known as groundspeed.

6.4.8 Component Indicating Initial Variables

A convention proposed by this standard is adding "Ic" to the end of any variable name, before any units, to designate that the variable is an initial condition specification. This can be added to virtually any variable without an underscore separator, conceptually creating a constant, for example:

1. x_bodyVelocityWrtEiIc_rad_s_X
2. grossWeightIc_N

6.4.9 Units Suffix

The suffix is used to describe the units of the variable. The convention for the suffix is simple and is followed for all variables. When exchanging simulation models, the units of all variables must be specified and this is the mechanism to do so. This will also allow the user, the programmer, and the reader of the code to check for homogeneity of the units and is self-documenting in this respect. Therefore, units shall be included in all variables except variables that are non-dimensional. If required for clarity, "nd" may be used in the units suffix to indicate a non-dimensional variable. Including units has the added advantage of making this standard consistent and acceptable in countries utilizing the international system of units. For example, airspeed is equally acceptable as a standard both for the U.S. system of units and the International system of units.

The standard uses an underscore (_) to separate the numerator from the denominator an analogy to exponential notation for the specification of units. For example, the unit expression for cubic feet per second squared (for example) would be \( \text{ft}^2\text{s}^{-2} \). Eliminating the superscripts leaves \( \text{ft}\text{s}^{-2} \). Separating the numerator from the denominator results in \( \text{ft}\text{s}^{-2} \), since the negative sign in the denominator exponential term is dropped.

With few exceptions, only base units are supported; it is not allowed to have, for example, milliseconds (ms). Here the proper use would be to express that variable in seconds.

Further examples are as follows:

1. trueAirspeed_ft_s for feet per second (ft/s)
2. trueAirspeed_m_s for meters per second (m/s)
3. trueAirspeed_kt for knots (nautical miles per hour)
4. bodyAccelWrtEi_ft_s2_X for feet per second squared (ft/s^2)

The suffix shall be separated from the body of the variable name by an underscore. The standard unit notations are given in Table 3. SI units and standard abbreviations are included where available.
### Table 3 — Abbreviation for units of measure in standard variable names

<table>
<thead>
<tr>
<th>Time</th>
<th>volt, alternating current</th>
</tr>
</thead>
<tbody>
<tr>
<td>second</td>
<td>A&lt;sup&gt;o&lt;/sup&gt;</td>
</tr>
<tr>
<td>minute</td>
<td>frequency (hertz)</td>
</tr>
<tr>
<td>hour</td>
<td>inductance (henry)</td>
</tr>
<tr>
<td>Length</td>
<td>capacitance (farad)</td>
</tr>
<tr>
<td>inch</td>
<td>charge (coulomb)</td>
</tr>
<tr>
<td>foot</td>
<td>conductance (siemens)</td>
</tr>
<tr>
<td>meter</td>
<td>resistance (ohm)</td>
</tr>
<tr>
<td>nautical mile</td>
<td></td>
</tr>
<tr>
<td>statute mile</td>
<td></td>
</tr>
<tr>
<td>kilometer</td>
<td></td>
</tr>
<tr>
<td>centimeter</td>
<td></td>
</tr>
<tr>
<td>millimeter</td>
<td></td>
</tr>
<tr>
<td>astronomical unit</td>
<td></td>
</tr>
<tr>
<td>Force</td>
<td></td>
</tr>
<tr>
<td>pound force</td>
<td></td>
</tr>
<tr>
<td>Newton</td>
<td></td>
</tr>
<tr>
<td>kilogram force</td>
<td></td>
</tr>
<tr>
<td>Mass</td>
<td></td>
</tr>
<tr>
<td>kilogram</td>
<td></td>
</tr>
<tr>
<td>pound mass</td>
<td></td>
</tr>
<tr>
<td>slug</td>
<td></td>
</tr>
<tr>
<td>Solid Angle</td>
<td></td>
</tr>
<tr>
<td>steradian</td>
<td></td>
</tr>
<tr>
<td>Plane Angle</td>
<td></td>
</tr>
<tr>
<td>degree</td>
<td></td>
</tr>
<tr>
<td>radian</td>
<td></td>
</tr>
<tr>
<td>revolution</td>
<td></td>
</tr>
<tr>
<td>Temperature</td>
<td></td>
</tr>
<tr>
<td>degrees Rankine</td>
<td></td>
</tr>
<tr>
<td>degrees Celsius</td>
<td></td>
</tr>
<tr>
<td>degrees Fahrenheit</td>
<td></td>
</tr>
<tr>
<td>Kelvin</td>
<td></td>
</tr>
<tr>
<td>Power, energy, work, heat</td>
<td></td>
</tr>
<tr>
<td>energy (British thermal unit)</td>
<td></td>
</tr>
<tr>
<td>energy (erg)</td>
<td></td>
</tr>
<tr>
<td>energy (calorie)</td>
<td></td>
</tr>
<tr>
<td>energy (joule)</td>
<td></td>
</tr>
<tr>
<td>power (horsepower)</td>
<td></td>
</tr>
<tr>
<td>power (watt)</td>
<td></td>
</tr>
<tr>
<td>Electrical units</td>
<td></td>
</tr>
<tr>
<td>potential (volt)</td>
<td></td>
</tr>
<tr>
<td>volt, direct current</td>
<td></td>
</tr>
</tbody>
</table>

Other

| pressure, stress (pascal) | ρ<sub><sup>b</sub></sup> |
| standard gravitational acceleration unit | g |
| luminous intensity (candela) | cd<sup>a</sup> |
| luminous flux (lumen) | lm<sup>b</sup> |
| illuminance (lux) | lx<sup>b</sup> |
| amount of substance (mole) | mol<sup>a</sup> |
| magnetic flux density (tesla) | T<sup>b</sup> |
| magnetic flux (weber) | Wb<sup>b</sup> |
| radioactive activity (becquerel) | Bq<sup>b</sup> |
| absorbed dose (gray) | Gy<sup>a</sup> |
| dose equivalent (sievert) | Sv<sup>a</sup> |
| nautical mile per hour | kt |
| non-dimensional | nd |
Notes:
  a. SI base unit (reference ISO 80000-1:2009, §6.5.2)
  b. SI derived unit (reference ISO 80000-1:2009, §6.5.3)
  c. SI base unit with modified abbreviation
  d. SI derived unit with modified abbreviation
  e. ISO recognized non-SI unit (reference ISO 80000-1:2009, §8.5.6)

6.4.10 Units-agnostic models

Some models have identical formulations whether the system of measurement is the SI system or the U.S. customary system. The units of the model's outputs depend only on the units of the values that are provided as inputs. These "units-agnostic" models can be reused in simulations with few or no conversions performed on inputs or outputs by the host simulation. To allow the exchange of units-agnostic models, the standard provides a set of abbreviations for generic units that represent the base units that differ between SI and U.S. customary system, namely length, mass, and temperature.

<table>
<thead>
<tr>
<th>Unit</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>length</td>
<td>L</td>
</tr>
<tr>
<td>mass</td>
<td>m</td>
</tr>
<tr>
<td>temperature</td>
<td>°C</td>
</tr>
</tbody>
</table>

Notes:
  a. ISQ base unit (reference ISO 80000-1:2009, §3.7)
  b. ISQ base unit with modified abbreviation

The two systems of measurement use the same unit for time (second). The U.S. customary system does not define units for the base quantities of electric current, luminous intensity, and amount of substance; the SI units (see ISO 80000-1:2009) fill this omission. Therefore, generic units are not required for time, electric current, luminous intensity or amount of substance; the SI unit for these quantities is used for variable names in unit agnostic models.

Examples of variable names using generic units:
  - bodyForce\_ML\_e2\_Z
  - bodyVelocity\_WrtGe\_L\_s\_X
  - thermalConductivity\_ML\_e3\_dT

Note that the last variable, thermal conductivity, would normally be published in SI units of W/(m\(^2\)K) [equal to kg/m( s\(^2\)-K)] with a conversion factor to the typical English units of BTU/(hr-ft\(^2\)-F), neither of which equals the English unit substitution in the variable name ofslug-ft/(s\(^2\)-R). So, a host simulation based on U.S. customary units would need to convert the published value toslug-ft/(s\(^2\)-R) before passing the value to the model. (A unit conversion is not required in a host simulation based on SI units.) If this model were originally developed for that host simulation, the model developer would likely have placed the necessary conversions from BTU and hours (or from the published SI value) within the model, however, by formulating the model as unit agnostic, responsibility for conversion has been moved to the host simulation. The thermal conductivity example illustrates how a unit agnostic model may still require conversion of inputs or outputs by the host simulation.

6.4.11 Component Indicating Specific Axis, Coordinate Component or Reference

The last component is the specific axis, coordinate component or reference used within the coordinate system (coordinate systems are defined in Section 5). It may also indicate elements of vectors and arrays. It is separated from the units by an underscore (\_), As can be seen in the examples, this component is appended last to keep the naming convention consistent for variables that are scalars or vectors. If the coordinate system is included in the name, the specific axis or reference should also be included.
Standard axes selector sets are:

a) \((X, Y, Z)\) for linear/ translational motion,

b) \((X, Y, Z), (Roll, Pitch, Yaw)\) or \((\text{First, Second, Third})\) for angular motion.

When the specific axis or reference can logically be a vector or an array, the vector or array component may be convenient for a specific implementation. When coordinate system vectors are used, a right-handed triad in order \((X, Y, Z)\) shall be used to avoid confusion. Due to differences between 0- and 1-based array indexing in various implementation languages, use of numerical indices is discouraged.

In the following examples, \(z\) would be defined as a constant of either 2 or 3 depending on the implementation language array indexing convention.

**Variable name examples:**

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
</table>
| x\_bodyAngularRate_rad\_s\_Roll | Here body is the coordinate system and roll is the specific axis in the body coordinate system, roll indicating angular motion. Examples show alternate scalar and vector implementations and implementation as a structure. 

**NOTE** In this example the variable is designated as a state. |
| x\_bodyAngularRate_rad\_s[Roll] | bodyTurbulenceVelocity\_WrtGe\_ft\_s\_Z (standard)                            |
| x\_bodyAngularRate_rad\_s.Roll   | bodyTurbulenceVelocity\_WrtGe\_ft\_s[Z]                                      |
| geVelocity\_m\_s\_Y             | Here ge is the coordinate system and \(y\) is the specific axis, also indicating translational motion. |
| runway22Velocity\_OfLeftWheel\_WrtTd\_ft\_s\_Z | where runway22 is the coordinate system (user defined) and \(z\) is the specific axis, also indicating translational motion. LeftWheel is the point on the vehicle and Td (touchdown point) is the reference point. |
| bodyAcceleration\_OfPilot\_Eye\_WrtBi\_m\_s\_2\_Y | bodyProduct\_Of\_Inertia\_slugf\_2\_YZ |

### 6.5 Additional Discussion

Very rarely, if ever, are all 10 components of a name used. In the case of

\[ x\_\text{bodyAngularRate}\_\text{rad}\_s\_\text{Roll} \]

the following five components were used:

1. prefix (\(x\)) indicating that in this formulation of the equations of motion this variable is a state,
2. coordinate or reference system (body),
3. core name (AngularRate),
4. units suffix (rad_s), for radians per second
5. specific axis or reference (Roll).

In this case "variable source domain" was omitted because $x_{\text{bodyAngularRate}_\text{rad}_s}_\text{Roll}$ is a single quantity defined by the laws of physics; there should not be separate body rates associated with aerodynamics and a propulsion system. If, however, the user wanted to have a multi-body simulation, logically the "variable source domain" could be used to discriminate between different elements of the body, or, perhaps more logically, an array or structure would be used to define different elements in a multi-body or flexible structure problem.

The $\text{OE}$, $\text{Wrt}$, and $\text{ObseFr}$ were omitted because the variable is describing motion about (of) the CM and relative to the locally-level coordinate system. Recall that $\text{InC}$ defaults to the locally-level frame for rotational motion and rotational motion variables do not allow specification of an observer's $(\text{ObseFr})$ coordinate system.

An $\text{IC}$ flag is not present, indicating that this variable does not specify an initial condition.

The intent of these conventions is to provide clear communication when exchanging models, not to force the universal use of these variable names. $x_{\text{bodyAngularRate}_\text{rad}_s}_\text{Roll}$ is intended to be a clear, brief, unambiguous name for the variable.

### 6.6.1 Discarded Conventions and Reasons

One convention considered eliminated the units suffix when the units were from a standard set, but this concept was discarded since always having the units associated with the variable name should help the developer maintain consistent units in the simulation and to reduce programming errors due to improper mixing of units. Consistent application of units in variable names should also reduce the software maintenance effort when a subsequent developer is trying to understand the code to make bug fixes, implement enhancements, or reuse the code.

### 6.6.2 Relationship with Markup Grammar, DAVE-ML

At present, this variable naming convention is targeted for use with the DAVE-ML XML grammar for model exchange (see Section 7). In DAVE-ML, the dynamic equation formulation prefix and the units suffix are stored as separate components (attributes or child elements) of the variable definition. Thus, including these in a variable name encoded in DAVE-ML would be redundant and a potential source of conflicting information.

The recommended practice is therefore to strip these components (the prefix and suffix) from the variable name when encoding to DAVE-ML, and reinsert them into the variable name if code or model data is generated from the DAVE-ML. Following this convention has two advantages.

1) The DAVE-ML grammar does not enforce naming rules; for those variables that do not conform to the naming convention and therefore do not have state/state derivative designation or units, DAVE-ML encourages the inclusion of this information to assist with the clear documentation of a model.

2) The convention allows XML processors to adopt the practice of automatically stripping and adding the prefix and suffix to the variable names, reducing the possibility of human error during translation.

### 6.6 Standard Variable Name Table Example

Using the conventions discussed above, a set of standard variable names has been created. These are
Flight Simulation Model Exchange

presented in Annex A. An excerpt of Annex A is given in Table 4 for illustrative purposes.

Interpretation of the standard variable name annex is best given by example. Table 4 presents
the standard variable defining the roll Euler angle, its axis system and positive sign convention (positive is
RWD, or right wing down). Four name examples are provided. The table includes:

1) The symbol for that variable, \( \Phi \)

2) The short name, PHI - the short name is included to accommodate standard variable definitions
   for legacy compilers with significant name length restrictions

3) One or more full names using the standard units conventions — generally, one full name with
   American convention units and one with SI units. Refer to section 6 for a list of the standard units
   and their abbreviations.

NOTE: While the variable naming convention described in Section 6 encourages the use of the
<variable domain> component, this Annex does not include variable domains as part of the
normative standard names. This is because the variable domain is normally dependent upon the
simulation architecture, and as such is immaterial to the exchange of a simulation model unless
the exchange is between facilities with similar architectures.

NOTE: Any suitable units may be used. In the example for eulerAngle_Roll both the _deg for
degrees and the _rad for radians are given. The "Full Variable Name" column does not
necessarily provide all acceptable units for each variable

4) A description of the variable, if applicable should always specify the coordinate system. Refer to
section 5 for a description of the standard coordinate systems.

5) The POSITIVE sign convention of the variable — RWD indicates that positive
eulerAngle_Roll is right wing down. (See section A.2.1 for a list of sign convention acronyms)

6) Minimum value, normally only specified for angles

7) Maximum values of the variable, normally only specified for angles

In addition this example also illustrates the pitch and yaw Euler angles.

Since roll, pitch and yaw may also conveniently be expressed as an array, the first variable name in Table
4 is the standard definition of the Euler angle array. Again, eulerAngle_rad(3) would be the standard
array using radians as the units and is fully compliant with the standard.

Euler angles are used in virtually any air vehicle simulation. While normally the coordinate system would
be included in the name, it was not included due to the universal definition of Euler angles. A more
rigorous name would be LL_eulerAngleOfBodyXrt11_deg[3] which expresses all the defaults (the
variable is the Euler angles of the body with respect to the locally level [1] coordinate system and is
presented in [or measured in] the locally level coordinate system). Aircraft simulations typically use Euler
angles defined via the 3-2-1 angle rotation sequence (yaw, pitch, roll); other rotation sequences may be
used but should be explicitly identified as in EulerAngle313, for example.

The standard allows use of any of the standard set of units (degrees or radians in this case).

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EUL</td>
<td>eulerAngle_deg(3)</td>
<td>eulerAngle_rad(3)</td>
<td>Array of the ownership roll, pitch, and yaw Euler angles comprised of the elements defined below: LL (locally level) coordinate system.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NESC Request No.: 09-00598
### 6.7 Summary

While it is recommended that this naming convention be adopted for defining future variables, the real key to a standard variable name is not the name, but the definition of the name. To exchange information between two or more organizations, the most important factor is not whether a variable is named “airspeed” or “VRM,” but that there exists a precise, unambiguous definition of the variable (true, indicated, or calibrated airspeed, etc.), including units and coordinate system.

Using the standard variable name simply provides a common language and set of definitions within which to facilitate transfer of the model.

The simulation community is encouraged to propose additional standard variable names. Annex C describes the web site used to support this standard. There is an appropriate URL or email address for submitting additional names or for recommending clarification of existing names.

### 6.8 References


7  Standard Simulation Data Format and XML Implementation of the Standard: DAVE-ML

7.1  Purpose

This section explains the requirements that a standard simulation data format must be able to satisfy. It includes the content of defined functions and configuration management of the content. The definition of the DAVE-ML format includes data for these components.

This document also discusses conceptually how a data table should be accessed in an executable program.

The standard is implemented in XML as specified by DAVE-ML. Annex B is the current version of the DAVE-ML reference document. Annex C provides links to example programs for loading and looking up data conforming to the XML standard.

7.2  Philosophy

Probably the greatest benefit of the standard to the simulation discipline is the definition of formats for the interchange of tabular data. Tabular data is used widely for non-linear function representation of aerodynamic, engine, atmospheric, and many other model parameters. The simplified interchange of such data should improve efficiency in the simulation community.

Most simulation developers and users have addressed this issue locally. In many simulation communities, a family of tools has been built around existing local function table formats. The intent of this standard is not to replace these local standards, but rather to define a format for communication that will allow each site to develop a single format converter to and from their local format. The DAVE-ML data representation is proposed as an exchange standard.

7.3  Design Objective

The first design objective of the standard data table format was to include all relevant information about real multi-dimensional functions, not just the data values. In the general case of a multi-dimensional table, the independent variables have different numbers of breakpoints, different breakpoints, and different valid ranges, which are all relevant to consistent evaluation of the function.

An equally important design objective was to allow the table to contain information on the data source (provenance, via reference), and a confidence interval for the data. Uses of confidence intervals within a model include direct computation of output confidence levels, estimation of output confidence intervals through Monte Carlo simulations, and mathematically combining different estimates of the same parameter at the same input values. Therefore, confidence statistics should be included when updating an existing or creating a new data set. DAVE-ML allows different types of confidence intervals, not all of which can be meaningfully combined.

The data format must also be easily read by computer or human, and be as self-documenting as possible.

7.4  Standard Function Table Data — An Illustrative Example

Figure 4 presents a fairly standard three-dimensional set of aerodynamic data typical of flight test or wind tunnel results. In the example, lift coefficient is a function of angle-of-attack, Mach number, and a control surface position. More generally stated, the function output (dependent variable) CLALFA is dependent on three inputs (independent variables): angleOfAttack_deg, mach, & controlSurfacePos_deg_avgElevator.

The example illustrates the following characteristics.
1) The number of breakpoints may be different for each independent variable. Data is presented for a different number of angle-of-attack (angleOfAttack_deg) points at each combination of Mach number (mach) and control position (controlSurfacePos_deg_avgElevator). For the first combination of Mach number and control position (mach = 0.6, controlSurfacePos_deg_avgElevator = 5) there are 17 angle-of-attack points. For the last combination of Mach number and control position (mach = 0.8, controlSurfacePos_deg_avgElevator = 0) there are 12 angle-of-attack points. There are also different numbers of Mach number points for each control position. The standard requires this to be represented as an ungridded table.

In contrast, a gridded table would require an function value be defined for every combination of a fixed set of Mach, angle-of-attack and control position breakpoint values.

2) At some breakpoints, the values of the other independent variables are different. Again, this is a characteristic of an ungridded table.

3) The valid ranges of the independent variables are different, another ungridded table characteristic.

4) The above three differences are not consistent for all data. For example, in the sample table the angleOfAttack_deg breakpoints for mach = 0.6 and mach = 0.7 and for controlSurfacePos_deg_avgElevator = -5 are identical.
Figure 4 — An illustration of a three-dimensional function table

For function data there is other information that is of importance to the user, without which the data is not very useful. In general this information is as follows.

a) Where did the data come from? For example what wind tunnel test or computational model?

b) How is it defined? For example, is this at a specific altitude? What is the vehicle configuration?

c) What are the engineering units of the output (the dependent variable) and the independent variables?

d) What is the sign convention of the independent and dependent variables? For example, is the control position positive trailing edge up or trailing edge down? Exactly which control surface is it?

e) Who created the table? Not where the data came from, but what person decided that this was the correct data for this table?

f) How has it been modified and for what reason?

g) How accurate is the data estimated to be? Or, mathematically what is the confidence interval of the data?

h) By what method is the data intended to be interpolated? For example, linear interpolation or cubic spline interpolation?

i) By what method is the data intended to be extrapolated when the independent variable values are outside the specified range for the breakpoint data?

The DAVE-ML grammar has data elements that contain all of the above information. It also includes the ability to automate static checks of the function data to allow spot checking of the function after it has been exchanged. It is discussed in detail in the DAVE-ML reference document contained in Annex B. An introduction and overview of DAVE-ML’s seven major elements is provided here.

7.5 DAVE-ML Major Elements (Annex B)

The major elements of DAVE-ML are listed below in the order required by the DAVE-ML DTD. The root element of a DAVE-ML model file, DAVEf func, can have several sub-elements and attributes; most attributes and sub-elements are optional. The only sub-elements a DAVE-ML file must contain are the fileHeader and at least one variableDef.

Information (breakpoints, data points, provenance, etc.) that is used by more than one major element should be defined once and then referenced in any subsequent use.

The sub-elements must appear in the following order (as required by the DTD).

fileHeader — states the source and purpose of the file. It must include the author’s contact details and the file creation date, and may include a description, reference information, and modification history.

variableDef — each variableDef defines one of the constants or signals (variables) used in the DAVE-ML model, whether input, output or internal. The definition includes all the attributes and sub-elements required to fully characterize the variable of interest, including a MathML definition if the variable’s value is equation-based. Standard variables as defined in Section 6 and Annex A are encouraged here.

breakpointDef — defines breakpoint sets to be used in the model. The breakpoints are the coordinate values along one axis of a gridded linear function value table. One breakpointDef may be reused by several functions.

griddedTableDef — defines an orthogonally-gridded multi-dimensional table of the values of a
function at the intersection of a set of specified independent inputs (breakpoints). The coordinates
along each dimension are defined in separate breakpointDef elements.

ungriddedTableDef — defines a table of non-orthogonal values of a function, each with the values of
their independent coordinates.

function — defines a function by connecting independent variables, breakpoints and data tables to
their output value.

checkData — contains one or more input/output vector pairs (and optionally a vector of internal
values) for the encoded model to assist in verification and debugging of the implementation.

Annex B contains the latest version of the DAVE-ML reference document, including a detailed description
and examples of the data element definitions of the DAVE-ML standard. Section 8 of Annex B provides
detailed XML element references and descriptions.

7.6 Simple DAVE-ML Examples

The easiest way to understand the standard is through an example. Annex B contains many more
examples of the DAVE-ML implementation of the standard.

A simple one dimensional relationship example, giving pitching moment coefficient as a function of angle
of attack, is shown in Table 5 and Figure 5.

Table 5 — A simple one-dimensional function table

<table>
<thead>
<tr>
<th>angleOfAttack deg</th>
<th>0</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>22</th>
<th>23</th>
<th>25</th>
<th>27</th>
<th>90</th>
</tr>
</thead>
<tbody>
<tr>
<td>cm(angleOfAttack)</td>
<td>0.1</td>
<td>-0.1</td>
<td>-0.09</td>
<td>-0.08</td>
<td>-0.05</td>
<td>-0.05</td>
<td>-0.07</td>
<td>-0.15</td>
<td>-0.6</td>
</tr>
</tbody>
</table>

Figure 5 — A simple one-dimensional gridded function

A DAVE-ML implementation for this function could be as follows.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE DAVEfunc PUBLIC "-//AIAA//DTD for Flight Dynamic Models - Functions 2.0//EN" "DAVEfunc.dtd">
<DAVEfunc xmlns="http://daveml.org/2010/DAVEML">
  <!--
       ----------------------------
  -->
  <!------------- File Header Components ----------------
  -->
```

NESC Request No.: 09-00598
<!-- This is an example of the file header components of the
 derivate of $C_m$ as a function of angle of attack. It must
define all documents that are later referenced by any function.

Note that there is not much information in this header,
because it is meant to be a simple example. In
reality, probably the most important information is the
author, the reference and the modification record, because
these data describe where the data came from and if it has
been changed (and how). See annex B for more complete
examples.
-->

<author name="Bruce Hildreth" org="JFTI" email="bhildreth@jfti.com"/>
<fileCreationDate date="2006-03-18"/>
<description>
This is made up data to use as an example of a simple gridded function.
</description>
<reference refID="ELHRpt1" author="Joe Smith"
title="A Generic Aircraft Simulation Model (does not really exist)"
accession="ISEN 1-2345-678-9" date="2004-01-01"/>

<!-- no modifications so far, so we don't need a modificationRecord yet -->

</fileHeader>

<!-- Variable Definition Components -->

<!-- Input variable -->

<variableDef name="Angle of attack" varID="angleOfAttack" units="deg">
<iisItAIAA/> <!-- Indicates that this variable is a standard
variable, which is why the author omitted
description and sign convention
and any other info. (it certainly could
be included here) -->
</variableDef>

<!-- Output (function value) -->

<variableDef name="Pitching moment coefficient due to angle of attack"
varID="CmAlfa" units="nondimensional" sign="+ANU">
<description>
The derivative of total pitching moment with respect to
angle of attack.
</description>
</variableDef>

<!-- Breakpoint Definition Set -->

<breakpointDef bpID="angleOfAttack_bp1"/>

<!--
Note that the bpID can be any valid XML string to uniquely identify the breakpoints. The author here chose to use a name related to the independent variable that is expected to be used to look up the function. In fact, if this set of breakpoints were shared by many functions and different independent variables would be used to look up the function, then the bpID of "angleOfAttack_bpl" would be misleading and a more generic name like "AOA" would probably be better.

```xml
<description>
  Angle of attack breakpoint set for Cm, Cda, and ClAlfa
</description>

<bpVals> <!-- Always comma separated values -->
  0, 18, 19, 20, 22, 23, 25, 27, 90
</bpVals>
</breakpointDef>

<!--
   ----------------------------- -->
<!--  Gridded Table Definition  -->
<!--  ----------------------------- -->

<griddedTableDef gtID="CmAlfa_Table1">
  <description>
    The derivative of Cm wrt fuselage AOA in degrees
  </description>

  <provenance>
    <author name="Jake Smith" org="AlCorp"/>
    <functionCreationDate date="2006-12-31"/>
    <documentRef refID="BLHRpt1"/>
    <!-- This points back to the Header, which provides the information about BLHRpt1. -->
  </provenance>

  <breakpointRef>
    <bpRef bpID="angleOfAttack_bpl"/>
  </breakpointRef>

  <uncertainty effect="percentage">
    <normalPDF numSigma="3">
      <bounds>12</bounds>
    </normalPDF>
    <!-- This means that the 3 sigma confidence is +/-12% on the Data. -->
  </uncertainty>

  <dataTable> <!-- Always comma separated values -->
    0.1, 0.1, 0.09, -0.06, -0.05, -0.05, -0.07, -0.15, -0.6
  </dataTable>

</griddedTableDef>

<!--
   ----------------------------- -->
<!--  Function Definition       -->
<!--  ----------------------------- -->

<!-- The function definition ties input and output variables with table definitions. This allows a level of abstraction such that the table, with its breakpoint definitions, can be reused -->
by several functions (such as left and right aileron or multiple
thruster effect tables).

<function name="Cm_alpha_func">
  <description>
  Variation of pitching moment coefficient with angle of attack (example)
  </description>
  <independentVarRef varID="angleOfAttack"/>
  <dependentVarRef varID="CmAlfa"/>
  <functionDefn>
    <griddedTableRef stID="CmAlfa_Table1"/>
  </functionDefn>
</function>

<!--
------------------------
<!--
------------------------
<!-- Check Data Cases
------------------------
<!-- Checkcase data provides automatic verification of the model by
  specifying the tolerance in output values for a given set of
  input values. One 'staticShot' is required per input/output
  mapping; in this case for a single input, single output model,
  we have a single input signal and a single output signal in each
  test point.
-->

<checkData>
  <staticShot name="case 1">
    <checkInputs>
      <signal>
        <varID>angleOfAttack</varID>
        <signalValue>0.0</signalValue>
      </signal>
    </checkInputs>
    <checkOutputs>
      <signal>
        <varID>CmAlfa</varID>
        <signalValue>0.0</signalValue>
        <tol>0.00001</tol>
      </signal>
    </checkOutputs>
  </staticShot>
  <staticShot name="case 2">
    <checkInputs>
      <signal>
        <varID>angleOfAttack</varID>
        <signalValue>5.0</signalValue>
      </signal>
    </checkInputs>
    <checkOutputs>
      <signal>
        <varID>CmAlfa</varID>
        <signalValue>0.00444</signalValue>
        <tol>0.00001</tol>
      </signal>
    </checkOutputs>
  </staticShot>
  <staticShot name="case 3">
    <checkInputs>
      <signal>
        <varID>angleOfAttack</varID>
        <signalValue>10.0</signalValue>
      </signal>
    </checkInputs>
    <checkOutputs>
      <signal>
        <varID>CmAlfa</varID>
        <signalValue>0.00001</signalValue>
        <tol>0.00001</tol>
      </signal>
    </checkOutputs>
  </staticShot>
</checkData>
<signal>
  </checkInputs>
  <checkOutputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>-0.09111</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 5">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>0.06667</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 6">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>0.08</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 7">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>-0.07</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 8">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>0.06667</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 9">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>0.08</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 10">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>-0.07</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 11">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>0.06667</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 12">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>0.08</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 13">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>-0.07</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 14">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>0.06667</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 15">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>0.08</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 16">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>-0.07</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 17">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>0.06667</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 18">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>0.08</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 19">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>-0.07</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 20">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>0.06667</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 21">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>0.08</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 22">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>-0.07</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 23">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>0.06667</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 24">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>0.08</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 25">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>-0.07</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 26">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>0.06667</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 27">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>0.08</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 28">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>-0.07</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 29">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>0.06667</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 30">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>0.08</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 31">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>-0.07</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 32">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>0.06667</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 33">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>0.08</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
<staticShot name="case 34">
  <checkInputs>
    <signal>
      <varID>CmAlfa</varID>
      <signalValue>-0.07</signalValue>
      <tol>0.00001</tol>
    </signal>
  </checkOutputs>
</staticShot>
While the above seems excessively long for a function with only 9 data points, most of its content involves self-documentation and checking. Therefore, as well as the function's data it includes the data's units, coordinate systems, uncertainty descriptions and provenance. It also includes many instructional comments, and verification data for multiple simulation conditions. Also, a very large complex function would only be expanded by the additional data points. The definitions and provenance information included with the function would probably not change much.

In the minimum, the same nominal data can be represented as shown. It is also possible to completely remove all whitespace between elements for more compactness, but this greatly affects readability by humans.

The other principal means of model representation in DAVE-ML is through Math-ML elements, which are used to specify calculations.

For example:

\[
\text{totalThrust}_N = \text{engine1Thrust}_N + \text{engine2Thrust}_N + \text{engine3Thrust}_N
\]

This equation, which is part of the model being exchanged, may be encoded in DAVE-ML and exchanged as data as shown below.
7.7 Summary

The DAVE-ML implementation of the standard enables nearly effortless transfer of simulation aerodynamics models between simulation facilities or architectures. Inclusion of Math-ML elements allows the formulation of algebraic equations, such as aerodynamic, propulsion, inertial, landing gear or control system models, to be included as data in the model. DAVE-ML is also suitable for use or transfer of tabular functions and algebraic equations for any type of data, not just simulation models.

While the above paragraphs provide an overview of the concepts implemented in DAVE-ML, Annex B is the normative authority for this standard. It includes much more detail and examples on how to easily build a DAVE-ML compliant simulation. Annex C provides a reference to the DAVE-ML web site, which includes tools that facilitate use of DAVE-ML based models in many applications.
8 Future Work

The AIAA Modeling and Simulation Technical Committee plans to continue its efforts in facilitating the exchange of simulations and models throughout the user community. Comments and suggestions on this expansion are welcomed on the simulation standards discussion group. Visit http://www.dascml.org for submission information. The following sections describe the two tasks of primary interest.

8.1 Time History Information

The immediate task that is being pursued is the transfer of validation data between facilities. This is for the purpose of sending time response validation data when a model is exchanged.

The approach being taken is to adopt a flight test data standard. This has the advantage of using an existing standard and facilitating the use of flight test data to validate a simulation. Lockheed Martin has an existing internal standard that they have released for use by the community. It is implemented in hierarchical data format (HDF) and has been adopted by the JSF community and other programs. It is the Modeling and Simulation Technical Committee’s intent to adopt this for the transfer of simulation validation data. Some work will be required to define the data elements that are required for the validation of a simulation. This is expected to be a subset of the data elements that comprise flight test data.

8.2 Dynamic Element Specification

The addition of the specification of dynamics (e.g., continuous and discrete states) is being considered to expand the scope of the standard. This expansion would allow more of the domain of a flight vehicle model (flight controls as a good example) to be exchanged in a non-proprietary, facility-neutral way.
9 Conclusion

This is a standard for the purpose of facilitating the exchange of simulation models between users. This purpose cannot be emphasized enough. It is not meant to enforce any standard simulation architecture. DAVE-ML provides the mechanism for exchange of the modeling data and equations; the standard variables and coordinate systems provide a common language to facilitate effective communication. The standard is also valuable for documenting a model, since the names and coordinate system definitions are clearly documented for the user.

A model can be DAVE-ML compliant without using any standard names or coordinate systems, but the exchange of such a model between users will be more difficult, since clear definitions will have to be exchanged also.

It is the earnest desire of the authors of this standard that the user community will employ the current standard for aerodynamic models, continue to suggest improvements to the standard, and develop tools to enhance the standard. http://nars.daveal.org contains information on how to be part of this effort and/or submit change or improvement recommendations.
10 Standard Variable Names (Normative)

(Now a separate .doc for ease of editing)
ANSI/AIAA  
Annex A. Standard Variable Names

Standard Variable Names (Normative)

A.1 General

The standard variable naming convention is described in detail in Section 6. The table in this annex contains a set of standard simulation variables that are independent of the particular vehicle type being simulated. Use of these standard variables provides a "standard language" which will facilitate the communication of the information required to exchange simulation models. These variables are tailored towards aircraft simulation and to a lesser extent, spacecraft. Visit http://DaveNL.org to suggest additional variables or changes to the existing list.

A.2 Table Explanation

Interpretation of the standard variable name table is best given by example. In general the table has 7 columns. These are described below using the eulerAngle.Roll as an example:

1) The symbol for that variable, Φ
2) The short name, PHI
3) One or more full names using the standard units conventions — generally, one full name with American convention units and one with SI units. Refer to section 6 for a list of the standard units and their abbreviations.

NOTE: While the variable naming convention described in Section 6 encourages the use of the <variable domain> component, this Annex does not include variable domains as part of the normative standard names. This is because the variable domain is normally dependent upon the simulation architecture, and as such is immaterial to the exchange of a simulation model unless the exchange is between facilities with similar architectures.

NOTE: Any suitable units may be used. In the example for eulerAngle.Roll both the _deg for degrees and the _rad for radians are given. The "Full Variable Name" column does not necessarily provide all acceptable units for each variable.

4) A description of the variable, if applicable should always specify the coordinate system. Refer to section 6 for a description of the standard coordinate systems.

5) The POSITIVE sign convention of the variable — RWD indicates that positive eulerAngle.Roll is right wing down. (See section A.2.1 for a list of sign convention acronyms)

6) Minimum value, normally only specified for angles

7) Maximum values of the variable, normally only specified for angles

Note:
This example also illustrates the pitch and yaw Euler angles.

Some variables may be used to represent variables referenced to more than one coordinate system. In this case the coordinate system is specified as xx and any coordinate system reference (refer to the body of this standard) may be substituted for the xx. For example, xxVelocity_ft_s_Y may represent:

- e1Velocity_ft_s_Y for the velocity along the Y axis of the e1 coordinate system - Earth centered Inertial (also known as geocentric inertial) coordinate system
- geVelocity_ft_s_Y for the velocity along the Y axis of the geocentric Earth (go) coordinate system. Also referred to as the Earth Centered Earth Fixed (ECEF coordinate system).
ANNEX A

Standard Variable Names

The variable name table below does not specify which variables are states, state derivatives, inputs, disturbances, simulation controls or initial conditions. These specifications may be added to any appropriate variable. Refer to Section 4.2 in the body of this standard for use of dynamic equation prefixes and section 6.4.8 for initial condition specification.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\theta$</td>
<td>PHI</td>
<td>eulerAngle_deg_Roll eulerAngle_rad_Roll</td>
<td>Roll Euler Angle, LL coordinate system</td>
<td>RWD</td>
<td>$-180, \pi$</td>
<td>$180, \pi$</td>
</tr>
<tr>
<td>$\theta$</td>
<td>THET</td>
<td>eulerAngle_deg_Pitch eulerAngle_rad_Pitch</td>
<td>Pitch Euler Angle, LL coordinate system</td>
<td>ANU</td>
<td>$-90, -\pi/2$</td>
<td>$90, \pi/2$</td>
</tr>
<tr>
<td>$\psi$</td>
<td>PSI</td>
<td>eulerAngle_deg_Yaw EulerAngle_rad_Yaw</td>
<td>Yaw Euler Angle, LL coordinate system</td>
<td>ANR</td>
<td>$-180, \pi$</td>
<td>$180, \pi$</td>
</tr>
</tbody>
</table>

A number or pair of numbers between square brackets represents the number of elements in an array or matrix respectively.

Annex A Page 2 of 54

NESC Request No.: 09-00598
A.2.1 Annex A Acronyms

Positive Sign Convention Acronyms

The following acronyms may appear as values for the “positive sign convention” of variables defined in the variable name tables.

+X in the positive direction of the presentation coordinate system’s X-axis

+Y in the positive direction of the presentation coordinate system’s Y-axis

+Z in the positive direction of the presentation coordinate system’s Z-axis

ABV positive above

AFT positive aft

AH positive above horizon

ANR positive aircraft nose right

ANU positive aircraft nose up

BH positive below horizon

BLO positive below

CCFN positive counterclockwise from north

COMP positive compressed

CWFN positive clockwise from north

DEC decrease

DN or Down positive down

E or East positive East

FWD positive forward

INC positive increase

LED positive leading edge down

LT or Left positive left

MAC percent mean aerodynamic chord

N or North positive North

NSC no sign convention (variable is always positive)

OUT positive outward

POS always positive

RT or Right positive right

RTCL positive right of centerline

RWD positive right wing down

Annex A Page 3 of 54
ANSI/AIAA
Annex A. Standard Variable Names

TED  positive trailing edge down
TEL  positive trailing edge left
TER  positive trailing edge right
TEU  positive trailing edge up
UP   positive up
WOW  weight on wheels

Control Surface and Position Acronyms

LEF  leading edge flap
TEF  trailed edge flap
### A.3 Standard Variable Name Tables

#### Table A.1 — Vehicle Positions and Angles

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi$</td>
<td>PHI</td>
<td>eulerAngle_deg_roll</td>
<td>Roll Euler Angle, LL coordinate system</td>
<td>RWD</td>
<td>-180, $\pi$</td>
<td>180, $\pi$</td>
<td></td>
</tr>
<tr>
<td>$\theta$</td>
<td>THET</td>
<td>eulerAngle_deg_pitch</td>
<td>Pitch Euler Angle, LL coordinate system</td>
<td>ANU</td>
<td>-90, $\pi/2$</td>
<td>90, $\pi/2$</td>
<td></td>
</tr>
<tr>
<td>$\psi$</td>
<td>PSI</td>
<td>eulerAngle_deg_yaw</td>
<td>Yaw Euler Angle, LL coordinate system</td>
<td>ANR</td>
<td>-180, $\pi$</td>
<td>180, $\pi$</td>
<td></td>
</tr>
<tr>
<td>sin $\phi$</td>
<td>Sphi</td>
<td>sinEulerAngle_roll</td>
<td>Sine Of Euler Roll Angle</td>
<td>RWD</td>
<td>-1.0</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>cos $\phi$</td>
<td>Cphi</td>
<td>cosEulerAngle_roll</td>
<td>Cosine Of Euler Roll Angle</td>
<td>RWD</td>
<td>-1.0</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>sin $\theta$</td>
<td>Stht</td>
<td>sinEulerAngle_pitch</td>
<td>Sine Of Euler Pitch Angle</td>
<td>ANU</td>
<td>-1.0</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>cos $\theta$</td>
<td>Ctht</td>
<td>cosEulerAngle_pitch</td>
<td>Cosine Of Euler Pitch Angle</td>
<td>ANU</td>
<td>0.0</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>sin $\psi$</td>
<td>Spsi</td>
<td>sinEulerAngle_yaw</td>
<td>Sine Of Euler Yaw Angle</td>
<td>ANR</td>
<td>-1.0</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>cos $\psi$</td>
<td>Cpsi</td>
<td>cosEulerAngle_yaw</td>
<td>Cosine Of Euler Yaw Angle</td>
<td>ANR</td>
<td>-1.0</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>$T_{ren}$</td>
<td>T</td>
<td>feToBodyT[3,3]</td>
<td>The FE to Body transformation matrix composed of the elements defined below</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_{ren[1,1]}$</td>
<td>T11</td>
<td>feToBodyT11</td>
<td>CTHT*CPSI (FE To B) coordinate transformation element</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_{ren[2,1]}$</td>
<td>T21</td>
<td>feToBodyT21</td>
<td>SPHI<em>STHT</em>CPSI - CPHI*SPSI (FE To B) coordinate transformation element</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Annex A Page 5 of 54
### Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(T_{\text{res}[3,1]})</td>
<td>T31</td>
<td>f2ToBodyT31</td>
<td>CPHI<em>STHT</em>CPSI + CPHI*SPSI (FE to E) coordinate transformation element</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(T_{\text{res}[1,2]})</td>
<td>T12</td>
<td>f2ToBodyT12</td>
<td>CTHT*SPSI (FE to B) coordinate transformation element</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(T_{\text{res}[2,2]})</td>
<td>T22</td>
<td>f2ToBodyT22</td>
<td>CPHI<em>STHT</em>SPSI + CPHI*CPSI (FE to B) coordinate transformation element</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(T_{\text{res}[3,2]})</td>
<td>T32</td>
<td>f2ToBodyT32</td>
<td>CPHI<em>STHT</em>SPSI - CPHI*CPSI (FE to B) coordinate transformation element</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(T_{\text{res}[1,3]})</td>
<td>T13</td>
<td>f2ToBodyT13</td>
<td>-STHT (FE to B) coordinate transformation element</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(T_{\text{res}[2,3]})</td>
<td>T23</td>
<td>f2ToBodyT23</td>
<td>CPHI*CTHT (FE to B) coordinate transformation element</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(T_{\text{res}[3,3]})</td>
<td>T33</td>
<td>f2ToBodyT33</td>
<td>CPHI*CTHT (FE to B) coordinate transformation element</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\gamma)</td>
<td>GAMV</td>
<td>flightPathAngle_rad</td>
<td>Flight Path Angle Above Horizon</td>
<td>ANL</td>
<td>-(\pi/2)</td>
<td>(-90)</td>
<td>90</td>
</tr>
<tr>
<td>(\chi)</td>
<td>GAMH</td>
<td>flightPathAzimuth_rad</td>
<td>Flight Path Angle In Horizon Plane, from North</td>
<td>CWFN</td>
<td>-(\pi)</td>
<td>-180</td>
<td>(\pi)</td>
</tr>
<tr>
<td>(h)</td>
<td>ALT</td>
<td>altitudeMsl_ft, altitudeMsl_m</td>
<td>Geometric altitude of vehicle altimeter above Mean Sea Level</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Annex A Page 6 of 54
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Long Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>λ</td>
<td>glongitude_rad</td>
<td>Longitude of Vehicle Cm with respect to the ge (geocentric Earth) coordinate system.</td>
<td>WEST</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>glongitude_deg</td>
<td>Note: The coordinate system may be deleted if Ge coordinate system, resulting in</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>longitude_rad</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>longitude_deg</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>φ</td>
<td>glatitude_rad</td>
<td>Geodetic Latitude of Vehicle Cm with respect to the ge (geocentric Earth) coordinate system.</td>
<td>NORTH</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>glatitude_deg</td>
<td>Note: The coordinate system may be deleted if Ge coordinate system, resulting in</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>latitude_rad</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>latitude_deg</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XLONGIMU</td>
<td>glongitudeOfImu_rad</td>
<td>Longitude of Vehicle Imu with respect to the ge coordinate system. This variable does not include any Imu sensor errors.</td>
<td>West</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>glongitudeOfImu_deg</td>
<td>Note: The coordinate system may be deleted if Ge coordinate system, resulting in</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>longitudeOfImu_rad</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>longitudeOfImu_deg</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XLATIMU</td>
<td>glatitudeOfImu_rad</td>
<td>Geodetic Latitude of Vehicle Imu with respect to the ge coordinate system. This variable does not include any Imu sensor errors.</td>
<td>NORTH</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>glatitudeOfImu_deg</td>
<td>Note: The coordinate system may be deleted if Ge coordinate system, resulting in</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>latitudeOfImu_rad</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>latitudeOfImu_deg</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>geSensedLongitudeOfImu_rad</td>
<td></td>
<td>geSensedLongitudeOfImu_deg</td>
<td>Longitude of Vehicle Imu with respect to the ge coordinate system. This variable includes any Imu sensor errors.</td>
<td>West</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: The coordinate system may be deleted if the ge coordinate system, resulting in sensedLongitudeOfImu_rad and sensedLongitudeOfImu_deg.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>geSensedLatitudeOfImuWrtSzs_rad</td>
<td></td>
<td>geSensedLatitudeOfImuWrtSzs_deg</td>
<td>Geodetic Latitude of Vehicle Imu with respect to the ge coordinate system. This variable includes any Imu sensor errors.</td>
<td>NORTH</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: The coordinate system may be deleted if the ge coordinate system, resulting in sensedLatitudeOfImu_rad and sensedLatitudeOfImu_deg.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HGT_RW</td>
<td></td>
<td>runwayHeightWrtHsl_ft</td>
<td>Height Of Runway w.r.t. mean Sea Level</td>
<td>Above</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RunwayHeightWrtHsl_m

General Definition

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>xxPositionOfTyyWrtSzs_ft</td>
<td></td>
<td>xxPositionOfTyyWrtSzs_m</td>
<td>For Example, xxPosition_ft[3] is the same as xxPositionOfTyyWrtxx_ft[3]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

General Definition for Linear and Angular Positions: Refer to section 6 of this standard for complete guidance on the naming of variables. Vector of positions of Tyy with respect to Szs (a user defined reference point or coordinate system origin) in the xx coordinate system. The lengths of xx, Tyy, Szs are not restricted to 2 and 3 characters respectively.

The coordinate system xx must always be defined. If the TyyWrt is not defined the definition defaults to the vehicle Cm for linear positions and the body coordinate system for angular positions. If the WrtSzs is not defined the reference point defaults to the origin of the xx coordinate system for linear position and the locally level (L) coordinate system for angular position.

Comprised of the three components as defined below.
ANSI/AI A A  S-119  

Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>xPositionOfYyyWrtZzz_ft_X</td>
<td>xPosition_of_YyyWrt_zzz_m_X</td>
<td>X position of Yyy with respect to Zzz (a user defined reference point) in the xx coordinate system. Defaults to the Cm and origin of the xx coordinate system.</td>
<td>(yyy -Zzz)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>or</td>
<td>xPosition_ft_X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>xPositionOfYyyWrtZzz_ft_Y</td>
<td>Y position of Yyy with respect to Zzz (a user defined reference point) in the xx coordinate system. Defaults to the Cm and origin of the xx coordinate system.</td>
<td>(yyy -Zzz)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>or</td>
<td>xPosition_ft_Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>xPositionOfYyyWrtZzz_ft_Z</td>
<td>Z position of Yyy with respect to Zzz (a user defined reference point) in the xx coordinate system. Defaults to the Cg and origin of the xx coordinate system.</td>
<td>(yyy -Zzz)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>or</td>
<td>xPosition_ft_Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

fePosition_ft [3] Vector of positions of the CM in the flat Earth coordinate system. Comprised of the three components as defined below.

| XCG | fePosition_ft_X | X or North position of the CM in the flat Earth coordinate system |
| YCG | fePosition_ft_Y | Y or East position of the CM in the flat Earth coordinate system |
| ZCG | fePosition_ft_Z | Z or Down position of the CM in the flat Earth coordinate system mins equals above the ground |
### Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>xxPositionOfMrcWrtSss_n_x</td>
<td></td>
<td>xxPositionOfMrcWrtSss_n_x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xxPositionOfMrcWrtSss_n_y</td>
<td></td>
<td>xxPositionOfMrcWrtSss_n_y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REF [3]</td>
<td>bodyPositionOfMrc_ft [3]</td>
<td>REF bodyPositionOfMrcWrtSss_ft_z</td>
<td>Vector of positions of the aerodynamic moment reference center in the body coordinate system. Since the origin of the body coordinate system is the Cm, this vector is the position of the Mrc with respect to the Cm.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Comprised of the three components as defined below.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XREF</td>
<td>bodyPositionOfMrc_wrt_XX</td>
<td>XREF bodyPositionOfMrcWrtSss_ft_x</td>
<td>X position of the Mrc in the body coordinate system.</td>
<td>Mrc in front of the origin (Cm)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YREF</td>
<td>bodyPositionOfMrc_wrt_YY</td>
<td>YREF bodyPositionOfMrcWrtSss_ft_y</td>
<td>Y position of the Mrc in the body coordinate system.</td>
<td>Mrc out the right wing with respect to the origin (Cm)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZREF</td>
<td>bodyPositionOfMrc_wrt_ZZ</td>
<td>ZREF bodyPositionOfMrcWrtSss_ft_z</td>
<td>Z position of the Mrc in the body coordinate system.</td>
<td>Mrc below the origin (Cm)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>bodyPositionOfPilotEyeWrtCm_ft_X</td>
<td>bodyPositionOfPilotEyeWrtCm_ft_X</td>
<td>X position of the Cm point w.r.t Cm, in the body coordinate system.</td>
<td>Eye PWd of the Cm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>bodyPositionOfPilotEyeWrtCm_ft_Y</td>
<td>bodyPositionOfPilotEyeWrtCm_ft_Y</td>
<td>Y position of the Cm point w.r.t Cm, in the body coordinate system.</td>
<td>Eye Right of the Cm</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Annex A Page 10 of 54
## Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZPLT2C</td>
<td>bodyPositionOfPilotEyeWrtCm_ft_Z</td>
<td>bodyPositionOfPilotEyeWrtCm_ft_Z</td>
<td>Z position of pilot eye point w.r.t Cm, in the body coordinate system</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**EXAMPLES**

- runway22Position_ft[3]
- runway22PositionOfOmNrtRunway22_ft[3]
- runway22PositionOfPdLeftMainWheel3NrtTd_ft[3]

Indicates position of the Cm (default) with respect to the origin of the Runway 22 coordinate system (default) in the runway22 coordinate system. A more complete and clear name would be:

- runway22PositionOfOmNrtRunway22_ft[3]
- runway22PositionOfPdLeftMainWheel3NrtTd_ft[3]

Indicates position of the forward left main wheel with respect to the touchdown point in the Runway 22 coordinate system. 

**NOTE** All coordinate systems are user defined.

- Cm X-position w.r.t runway touchdown point in the specified (Runway22) coordinate system.
- Cm Y-position w.r.t runway touchdown point in the specified (Runway22) coordinate system.
- Cm Z-position w.r.t runway touchdown point in the specified (Runway22) coordinate system. (This variable is normally negative.)
- Geocentric radius of Earth (center to smooth surface), round Earth model or oblate spheroid at the nadir of the aircraft.
ANSI/AIAA S-119

Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>RALT</td>
<td>heightOfCmWrtTerrain_ft heightOfCmWrtTerrain_m or heightWrtTerrain_ft heightWrtTerrain_m</td>
<td>Height of the aircraft Cm above the terrain</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTERRAI N</td>
<td>heightOfTerrainWrtGround_ft heightOfTerrainWrtGround_m</td>
<td>Height of the terrain at the nadir of the aircraft Cm. It is the terrain height above the smooth surface of the Earth, regardless of whether a flat, round or oblate spheroid model is used.</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table A.2 — Vehicle velocities and angular rates

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V'_{TXX} )</td>
<td>VTxx</td>
<td>totalSpeedWrtXx_ft_s totalSpeedWrtXx_m_s</td>
<td>Total Velocity with respect to and observed from Xx where Xx is the coordinate system as defined in the body of this standard.</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( V_X )</td>
<td>V3</td>
<td>groundSpeed_ft_s groundSpeed_m_s</td>
<td>Vehicle speed along the ground.</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( M_N )</td>
<td>XMACH</td>
<td>mach</td>
<td>Mach number of the aircraft</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( h'_{XX} )</td>
<td>ALTOxx</td>
<td>xxAltitudeRate_ft_s xxAltitudeRate_m_s</td>
<td>Attitude time rate of change in xx coordinate system.</td>
<td>DN</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Annex A Page 12 of 54
## Annex A. Standard Variable Names

### General Definition-Translational Velocities

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>XLOND</td>
<td>xlLongitudeRate_rad_s</td>
<td>Longitude Rate Of Change in xa coordinate system.</td>
<td>WEST</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>xlLongitudeRate_deg_s</td>
<td>Default is Ge coordinate system.</td>
<td>WEST</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>xlatitudeRate_rad_s</td>
<td>Geocentric Latitude Rate Of Change in xa coordinate system.</td>
<td>NORTH</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>xlatitudeRate_deg_s</td>
<td>Default is Ge coordinate system.</td>
<td>NORTH</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### General Definition-Angular Velocities

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>xAngularRateOfYyyMrtSzz_rad_s [3]</td>
<td>General Definition for Rotational Motion: Refer to section 6 of this standard for complete guidance on the naming of variables. General expression for angular velocities presented in the xc coordinate system. Yyy indicates the reference coordinate system on the vehicle and the Ody_y may be omitted if it is the Z coordinate system. Zzz represents the coordinate system that the vehicle is moving with respect to and which the vehicle motion is observed. The Ody_y may be omitted if it is the Z coordinate system. (see section 6.3.3 for more detail) Therefore $\omega_{s,Roll}$ is the angular rate of the body axis of the vehicle with respect to the locally level coordinate system and presented (or measured) in the Earth centered inertial (ei) coordinate system and is the default expression for $\omega_{s,Roll}$.</td>
<td>WEST</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>xAngularRateOfYyyMrtSzz_rad_s [3]</td>
<td>General Definition for Rotational Motion: Refer to section 6 of this standard for complete guidance on the naming of variables. General expression for angular velocities presented in the xc coordinate system. Yyy indicates the reference coordinate system on the vehicle and the Ody_y may be omitted if it is the body coordinate system. Zzz represents the coordinate system that the vehicle is moving with respect to and which the vehicle motion is observed. The Ody_y may be omitted if it is the locally level (LL) coordinate system. (see section 6.3.3 for more detail) Therefore $\omega_{s,Roll}$ is the angular rate of the body axis of the vehicle with respect to the locally level coordinate system and presented (or measured) in the Earth centered inertial (ei) coordinate system and is the default expression for $\omega_{s,Roll}$.</td>
<td>WEST</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>OM</td>
<td>bodyAngularRate_ref_rad [rad/s]</td>
<td>Vector of body coordinate angular rates of the ownership body coordinate system with respect to the locally level (LL) coordinate system. Comprised of the three components as defined below. Angular motion is always with respect to the LL coordinate system unless otherwise specified.</td>
<td>RWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>P</td>
<td>bodyAngularRate_ref_rad_a_Roll</td>
<td>Vehicle roll velocity, body coordinate system</td>
<td>RWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>Q</td>
<td>bodyAngularRate_ref_rad_a_Pitch</td>
<td>Vehicle pitch velocity, body coordinate system</td>
<td>ANU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>R</td>
<td>bodyAngularRate_ref_rad_a_Yaw</td>
<td>Vehicle yaw velocity, body coordinate system</td>
<td>ANU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>q_b</td>
<td>OMB</td>
<td>bodyAngularRateRef_Ei_rad_a [rad/s]</td>
<td>Vector of body coordinate angular rates with respect to the Earth centered inertial (EI) coordinate system. Comprised of the three components as defined below.</td>
<td>RWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>q_b</td>
<td>PB</td>
<td>bodyAngularRateRef_Ei_rad_a_roll</td>
<td>Vehicle roll velocity, body coordinate system</td>
<td>RWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>q_b</td>
<td>QB</td>
<td>bodyAngularRateRef_Ei_rad_a_pitch</td>
<td>Vehicle pitch velocity, body coordinate system</td>
<td>ANU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r_b</td>
<td>RB</td>
<td>bodyAngularRateRef_Ei_rad_a_yaw</td>
<td>Vehicle yaw velocity, body coordinate system</td>
<td>ANU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p_x</td>
<td>PS</td>
<td>xAngularRateRefBodyRef_Ei_rad_a_roll</td>
<td>Body coordinate system Roll rate about the X axis in the sa coordinate system, also known as the Stability Axis roll rate.</td>
<td>RWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r_x</td>
<td>RS</td>
<td>xAngularRateRefBodyRef_Ei_rad_a_yaw</td>
<td>Body coordinate system Yaw rate about the Z axis in the sa coordinate system, also known as the Stability Axis yaw rate.</td>
<td>ANU</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( e )</td>
<td>EULD</td>
<td>eulerAngles_deg_s(3)</td>
<td>Array of the roll, pitch, and yaw Euler angle rates defined below. LL (locally level) coordinate system.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \phi )</td>
<td>PHID</td>
<td>eulerAngles_rad_s_Roll</td>
<td>Euler roll rate, LL coordinate system.</td>
<td>RWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \theta )</td>
<td>THETO</td>
<td>eulerAngles_rad_s_Pitch</td>
<td>Euler pitch rate, LL coordinate system.</td>
<td>ANU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \psi )</td>
<td>PSID</td>
<td>eulerAngles_rad_s_Yaw</td>
<td>Euler yaw rate, LL coordinate system.</td>
<td>ANR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \mathbf{v} )</td>
<td>VEL</td>
<td>bodyVelocity_MtoAir_ft_s(3)</td>
<td>Vector of body coordinate velocities of the Cm with respect to the instantaneous wind comprised of the three components as defined below. This is the conventional body coordinate system airspeed vector.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( u )</td>
<td>U</td>
<td>bodyVelocity_MtoAir_ft_s_X</td>
<td>X-velocity Body coordinate system.</td>
<td>FWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( v )</td>
<td>V</td>
<td>bodyVelocity_MtoAir_ft_s_Y</td>
<td>Y-velocity Body coordinate system</td>
<td>RT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( w )</td>
<td>W</td>
<td>bodyVelocity_MtoAir_ft_s_Z</td>
<td>Z-velocity Body coordinate system.</td>
<td>DN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \mathbf{v}' )</td>
<td>VEL8[3]</td>
<td>bodyVelocity_MtoSI_ft_s(3)</td>
<td>Vector of body coordinate inertial velocities of the Cm comprised of the three components as defined below.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_b$</td>
<td>UB</td>
<td>bodyVelocityECEF_x</td>
<td>X-velocity Body coordinate system</td>
<td>FWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v_b$</td>
<td>VB</td>
<td>bodyVelocityECEF_y</td>
<td>Y-velocity Body coordinate system</td>
<td>RT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$w_b$</td>
<td>WB</td>
<td>bodyVelocityECEF_z</td>
<td>Z-velocity Body coordinate system</td>
<td>DN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v'_{b_e}$</td>
<td>VELE[B]</td>
<td>bodyVelocityGECEF_x</td>
<td>Vector of body coordinate Cm velocities with respect to the geocentric Earth (Ge) coordinate system</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>bodyVelocityGECEF_y</td>
<td>Comprised of the three components as defined below.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>bodyVelocityGECEF_z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$u_e$</td>
<td>UE</td>
<td>bodyVelocityGECEF_x</td>
<td>X-velocity Body coordinate system</td>
<td>FWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v_e$</td>
<td>VE</td>
<td>bodyVelocityGECEF_y</td>
<td>Y-velocity Body coordinate system</td>
<td>RT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$w_e$</td>
<td>WE</td>
<td>bodyVelocityGECEF_z</td>
<td>Z-velocity Body coordinate system</td>
<td>DN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v'<em>{e</em>{r_{ge}}}$</td>
<td>VELFE</td>
<td>feVelocity_ECEF_x</td>
<td>Vector of Flat Earth (FE) coordinate translational velocities of the Cm comprised of the three components as defined below.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>feVelocity_ECEF_y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>feVelocity_ECEF_z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v_n$</td>
<td>VNF</td>
<td>feVelocity_ECEF_x</td>
<td>Northward Velocity Over Flat Earth (FE) coordinate system [flat, non-rotating Earth]</td>
<td>NORTH</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Annex A Page 16 of 54
## ANSI/AIAA S-119

### Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_E )</td>
<td>VEFE</td>
<td>feVelocity_ft_s Y / feVelocity_m_s Y</td>
<td>Eastward Velocity Over Flat Earth (FE) coordinate system [flat, non-rotating Earth]</td>
<td>EAST</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( V_D )</td>
<td>VDFE</td>
<td>feVelocity_ft_s Z / feVelocity_m_s Z</td>
<td>Downward Velocity Toward Earth Cn. (FE) coordinate system [flat, non-rotating Earth]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( V_E )</td>
<td>VE</td>
<td>geVelocity_ft_s [3] / geVelocity_m_s [3]</td>
<td>Vector of Geocentric Earth (GE) coordinate translational velocities of the Cm comprised of the three components as defined below.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( V_{E_X} )</td>
<td>VXGE</td>
<td>geVelocity_ft_s X / geVelocity_m_s X</td>
<td>( X ) axis velocity over the Earth in the geocentric Earth (GE) coordinate system in ft/sec</td>
<td>(+X)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( V_{E_Y} )</td>
<td>VYGE</td>
<td>geVelocity_ft_s Y / geVelocity_m_s Y</td>
<td>( Y ) axis velocity over the Earth in the geocentric Earth (GE) coordinate system in ft/sec</td>
<td>(+Y)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( V_{E_Z} )</td>
<td>VZGE</td>
<td>geVelocity_ft_s Z / geVelocity_m_s Z</td>
<td>( Z ) axis velocity over the Earth in the geocentric Earth (GE) coordinate system in ft/sec</td>
<td>(+Z)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**OTHER EXAMPLES**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_{E_X} )</td>
<td>VXGE</td>
<td>geVelocity_km_s X</td>
<td>( X ) axis velocity of the vehicle Cm is the geocentric (ge) coordinate system in kilometers/sec</td>
<td>(+X)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>runway22Velocity_ft_s Z</td>
<td>Z axis velocity of the Cm in the user defined &quot;runway22&quot; coordinate system in ft/s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Annex A Page 17 of 54
## Annex A. Standard Variable Names

### Table A.3 — Vehicle linear and angular accelerations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_x$</td>
<td>CMBD</td>
<td>bodyAngularAccel_rad_s2[3]</td>
<td>General Definition of Translational Accelerations (ax)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\phi$</td>
<td></td>
<td>bodyAngularAccel_deg_s2[3]</td>
<td>General Definition of Angular Accelerations (\phi)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p$</td>
<td>PBD</td>
<td>bodyAngularAccel_rad_s2_Roll [3]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>bodyAngularAccel_deg_s2_Roll</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \dot{q} )</td>
<td>QSD</td>
<td>bodyAngularAccel_rad_s2_Pitch&lt;br&gt;bodyAngularAccel_deg_s2_Pitch</td>
<td>Aircraft Pitch Acceleration, Body coordinate system</td>
<td>ANU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \dot{r} )</td>
<td>RBD</td>
<td>bodyAngularAccel_rad_s2_Yaw&lt;br&gt;bodyAngularAccel_deg_s2_Yaw</td>
<td>Aircraft Yaw Acceleration, Body coordinate system</td>
<td>ANR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \dot{\theta}_B )</td>
<td>OMBDE</td>
<td>bodyAngularAccelWrtEl_rad_s2[3]&lt;br&gt;bodyAngularAccelWrtEl_deg_s2[3]</td>
<td>Vector of body coordinate angular accelerations with respect to the Earth-centered inertial coordinate system, comprised of the three body axis components as defined below.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \dot{p} )</td>
<td>PBDE</td>
<td>bodyAngularAccelWrtEl_rad_s2_Roll&lt;br&gt;bodyAngularAccelWrtEl_deg_s2_Roll</td>
<td>Aircraft Roll Acceleration, Body coordinate system</td>
<td>RWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \dot{q} )</td>
<td>QBDE</td>
<td>bodyAngularAccelWrtEl_rad_s2_Pitch&lt;br&gt;bodyAngularAccelWrtEl_deg_s2_Pitch</td>
<td>Aircraft Pitch Acceleration, Body coordinate system</td>
<td>ANU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \dot{r} )</td>
<td>RBDE</td>
<td>bodyAngularAccelWrtEl_rad_s2_Yaw&lt;br&gt;bodyAngularAccelWrtEl_deg_s2_Yaw</td>
<td>Aircraft Yaw Acceleration, Body coordinate system</td>
<td>ANR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \dot{v}_B )</td>
<td>VBD</td>
<td>bodyAccelWrtEl_ft_s2[3]&lt;br&gt;bodyAccelWrtEl_m_s2[1]</td>
<td>Vector of accelerations of the Cm of the aircraft with respect to and observed from the ( \hat{e}_i ) coordinate system (Earth-centered inertial) in the body coordinate system. This variable includes the gravitation vector. Comprised of the three components as defined below.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \ddot{u}_B )</td>
<td>USB</td>
<td>bodyAccelWrtEl_ft_s2_X&lt;br&gt;bodyAccelWrtEl_m_s2_X</td>
<td>Longitudinal acceleration (along the X-body coordinate)</td>
<td>FWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \ddot{v}_B )</td>
<td>VBD</td>
<td>bodyAccelWrtEl_ft_s2_Y&lt;br&gt;bodyAccelWrtEl_m_s2_Y</td>
<td>Right Sideward Acceleration, Body coordinate</td>
<td>RT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \ddot{w}_B )</td>
<td>WBD</td>
<td>bodyAccelWrtEl_ft_s2_Z&lt;br&gt;bodyAccelWrtEl_m_s2_Z</td>
<td>Downward Acceleration, Body coordinate</td>
<td>DNDN</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\gamma$</td>
<td>bodyAccelMrtAir_ft_s2 [3]</td>
<td>bodyAccelMrtAir_m_s2 [3]</td>
<td>Vector of body coordinate accelerations of the ( C_m ) with respect to the mean air mass and comprised of the three components as defined below. This variable includes the gravitation vector. Comprised of the three components as defined below.</td>
<td>FWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\dot{w}$</td>
<td>bodyAccelMrtAir_ft_s2_X</td>
<td>bodyAccelMrtAir_m_s2_X</td>
<td>Longitudinal acceleration (along the X-body coordinate)</td>
<td>FWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\ddot{v}$</td>
<td>bodyAccelMrtAir_ft_s2_Y</td>
<td>bodyAccelMrtAir_m_s2_Y</td>
<td>Right Sideward Acceleration, along the Y Body coordinate</td>
<td>RT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\dot{w}$</td>
<td>bodyAccelMrtAir_ft_s2_Z</td>
<td>bodyAccelMrtAir_m_s2_Z</td>
<td>Downward Acceleration, along the Z Body coordinate</td>
<td>CDN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\dot{V}_E$</td>
<td>bodyAccelMrtGEE_ft_s2 [3]</td>
<td>bodyAccelMrtGEE_m_s2 [3]</td>
<td>Vector of body coordinate ( C_m ) velocities with respect to the geocentric Earth (GE) coordinate system. Comprised of the three components as defined below. This variable includes the gravitation vector. Comprised of the three components as defined below.</td>
<td>FWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\dot{U}_E$</td>
<td>bodyAccelMrtGEE_ft_s2_X</td>
<td>bodyAccelMrtGEE_m_s2_X</td>
<td>Longitudinal acceleration (along the X-body coordinate)</td>
<td>FWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\dot{V}_E$</td>
<td>bodyAccelMrtGEE_ft_s2_Y</td>
<td>bodyAccelMrtGEE_m_s2_Y</td>
<td>Right Sideward Acceleration, along the Y Body coordinate</td>
<td>RT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\dot{W}_E$</td>
<td>bodyAccelMrtGEE_ft_s2_Z</td>
<td>bodyAccelMrtGEE_m_s2_Z</td>
<td>Downward Acceleration, along the Z Body coordinate</td>
<td>DN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>totalAccelMrtEl_ft_s2</td>
<td></td>
<td>Magnitude of the inertial acceleration vector</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>VTDox</td>
<td>totalVelocityDataWrtXx_ft_s2</td>
<td>totalVelocityRateWrtXx_m_s2</td>
<td>Rate of change of speed with respect to and observed from Xx, where Xx is the coordinate system as defined in the body of this standard. Total velocity (speed) rate of change is not the same as total acceleration (magnitude of the acceleration vector). For example, a velocity vector that only undergoes a direction change shows zero change in speed (magnitude) but still shows a positive total acceleration due to its direction change.</td>
<td>INC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>xxAccel_ft_s2 [3]</td>
<td>xxAccel_m_s2 [3]</td>
<td>General form for the vector of aircraft translational acceleration with respect to and observed from the specified [xx] coordinate system comprised of the three components as defined below.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>xxAccel_ft_s2_X</td>
<td>xxAccel_m_s2_X</td>
<td>X-axis acceleration in xx coordinate system</td>
<td>+X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>xxAccel_ft_s2_Y</td>
<td>xxAccel_m_s2_Y</td>
<td>Y-axis acceleration in xx coordinate system</td>
<td>+Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>xxAccel_ft_s2_Z</td>
<td>xxAccel_m_s2_Z</td>
<td>Z-axis acceleration in xx coordinate system</td>
<td>+Z</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bodyAccel0fImuWrtSi_ft_s2[3]</td>
<td>bodyAccel0fImuWrtSi_m_s2[3]</td>
<td>Vector of true inertial accelerations at the inertial measurement unit (imu) including the effects of the gravitation vector. This variable assumes a perfect imu Comprised of the three body coordinate system components as defined below.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AX</td>
<td>bodyAccelOfImuWrtEi_ft_a2_X</td>
<td>bodyAccelOfImuWrtEi_ft_a2_X</td>
<td>X Acceleration Of aircraft Cm (body coordinate)</td>
<td>FWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bodyAccelOfImuWrtEi_n_a2_X</td>
<td>bodyAccelOfImuWrtEi_n_a2_X</td>
<td>Includes the gravitation vector.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bodyAccelOfImuWrtEi_ft_a2_Y</td>
<td>bodyAccelOfImuWrtEi_ft_a2_Y</td>
<td>Y Acceleration Of aircraft Cm (body coordinate)</td>
<td>RT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bodyAccelOfImuWrtEi_n_a2_Y</td>
<td>bodyAccelOfImuWrtEi_n_a2_Y</td>
<td>Includes the gravitation vector.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bodyAccelOfImuWrtEi_ft_a2_S</td>
<td>bodyAccelOfImuWrtEi_ft_a2_S</td>
<td>Z Acceleration Of aircraft Cm (body coordinate)</td>
<td>DN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bodyAccelOfImuWrtEi_n_a2_S</td>
<td>bodyAccelOfImuWrtEi_n_a2_S</td>
<td>Includes the gravitation vector.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bodySenseAccelOfImuWrtEi_ft_a2[3]</td>
<td>bodySenseAccelOfImuWrtEi_ft_a2[3]</td>
<td>Vector of sensed inertial accelerations at the inertial measurement unit (imu) including the effects of the gravitation vector. This variable includes any sensor scale, bias and noise. Comprised of the three body coordinate system components as defined below</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bodySenseAccelOfImuWrtEi_n_a2[3]</td>
<td>bodySenseAccelOfImuWrtEi_n_a2[3]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bodySenseAccelOfImuWrtEi_ft_a2_X</td>
<td>bodySenseAccelOfImuWrtEi_ft_a2_X</td>
<td>X Acceleration Of aircraft Cm (body coordinate)</td>
<td>FWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bodySenseAccelOfImuWrtEi_n_a2_X</td>
<td>bodySenseAccelOfImuWrtEi_n_a2_X</td>
<td>Includes the gravitation vector.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bodySenseAccelOfImuWrtEi_ft_a2_Y</td>
<td>bodySenseAccelOfImuWrtEi_ft_a2_Y</td>
<td>Y Acceleration Of aircraft Cm (body coordinate)</td>
<td>RT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bodySenseAccelOfImuWrtEi_n_a2_Y</td>
<td>bodySenseAccelOfImuWrtEi_n_a2_Y</td>
<td>Includes the gravitation vector.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bodySenseAccelOfImuWrtEi_ft_a2_S</td>
<td>bodySenseAccelOfImuWrtEi_ft_a2_S</td>
<td>Z Acceleration Of aircraft Cm (body coordinate)</td>
<td>DN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bodySenseAccelOfImuWrtEi_n_a2_S</td>
<td>bodySenseAccelOfImuWrtEi_n_a2_S</td>
<td>Includes the gravitation vector.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bodyAccelOfPilotWrtEi_ft_a2[3]</td>
<td>bodyAccelOfPilotWrtEi_ft_a2[3]</td>
<td>Vector of inertial accelerations at the pilot reference point, in the body coordinate system, comprised of the three components as defined below</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bodyAccelOfPilotWrtEi_n_a2[3]</td>
<td>bodyAccelOfPilotWrtEi_n_a2[3]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Annex A Page 22 of 54
### Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AXP</td>
<td>bodyAccelOfPilotWrtEI_ft_s2_X</td>
<td>bodyAccelOfPilotWrtEI_m_s2_X</td>
<td>X Acceleration Of Pilot reference point (body coordinate)</td>
<td>FWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AYP</td>
<td>bodyAccelOfPilotWrtEtf_s2_Y</td>
<td>bodyAccelOfPilotWrtEti_m_s2_Y</td>
<td>Y Acceleration Of Pilot reference point (body coordinate)</td>
<td>RT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AZP</td>
<td>bodyAccelOfPilotWrtEI_ft_s2_Z</td>
<td>bodyAccelOfPilotWrtEI_m_s2_Z</td>
<td>Z Acceleration Of Pilot reference point (body coordinate)</td>
<td>DN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GVBODY</td>
<td>bodyLocalGravitation_ft_s2 [3]</td>
<td>bodyLocalGravitation_m_s2 [3]</td>
<td>Local gravitation vector in the body coordinate system</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GVBODYX</td>
<td>bodyLocalGravitation_ft_s2_X</td>
<td>bodyLocalGravitation_m_s2_X</td>
<td>X axis component</td>
<td>FWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GVBODYY</td>
<td>bodyLocalGravitation_ft_s2_Y</td>
<td>bodyLocalGravitation_m_s2_Y</td>
<td>Y axis component</td>
<td>RT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GVBODYZ</td>
<td>bodyLocalGravitation_ft_s2_Z</td>
<td>bodyLocalGravitation_m_s2_Z</td>
<td>Z axis component</td>
<td>DN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GVGE</td>
<td>gLocalNavitiation_ft_s2 [3]</td>
<td>gLocalNavitiation_m_s2 [3]</td>
<td>Local gravitation vector in the ge (geocentric Earth) coordinate system</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GVGEK</td>
<td>gLocalNavitiation_ft_s2_X</td>
<td>gLocalNavitiation_m_s2_X</td>
<td>X axis component</td>
<td>FWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GVGEY</td>
<td>gLocalNavitiation_ft_s2_Y</td>
<td>gLocalNavitiation_m_s2_Y</td>
<td>Y axis component</td>
<td>RT</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GVEZ</td>
<td>eilocalGravitation_ft_s2_Z</td>
<td>eilocalGravitation_ft_s2_Z</td>
<td>Z axis component</td>
<td>DN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GVEIX</td>
<td>eilocalGravitation_ft_s2_X</td>
<td>eilocalGravitation_ft_s2_X</td>
<td>X axis component</td>
<td>FWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GVEFY</td>
<td>eilocalGravitation_ft_s2_Y</td>
<td>eilocalGravitation_ft_s2_Y</td>
<td>Y axis component</td>
<td>RT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GVEZ</td>
<td>eilocalGravitation_ft_s2_Z</td>
<td>eilocalGravitation_ft_s2_Z</td>
<td>Z axis component</td>
<td>DN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>localGravity_ft_s2</td>
<td>localGravity_m_s2</td>
<td>Acceleration Due To Gravity (at the vehicle altitude), in the LI coordinate system.</td>
<td>ANU</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table A.4 — Vehicle air data

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>ALFA</td>
<td>angleOfAttack_deg</td>
<td>Angle Of Attack, Body coordinate</td>
<td>ANU</td>
<td>−π / 180</td>
<td>+π / 180</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>angleOfAttack_rad</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>BETA</td>
<td>angleOfSlip_deg</td>
<td>Sideslip Angle, Body coordinate</td>
<td>ANL</td>
<td>−π / 180</td>
<td>+π / 180</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>angleOfSlip_rad</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>α</td>
<td>ALFD</td>
<td>angleOfAttackRate_rad_s</td>
<td>Angle Of Attack Rate, Body coordinate</td>
<td>ANU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symbol</td>
<td>Short Name</td>
<td>Full Variable Name</td>
<td>Description</td>
<td>Positive Sign</td>
<td>Initial Value</td>
<td>Min Value</td>
<td>Max Value</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
<td>--------------------------------------------------------</td>
<td>--------------------------------------------------</td>
<td>---------------</td>
<td>---------------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>(\beta)</td>
<td>BETD</td>
<td>angleOfSideslipRate_rad_s</td>
<td>Sideslip Angle Rate</td>
<td>ANL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\sin \alpha)</td>
<td>SALPH</td>
<td>sineAngleOfAttack</td>
<td>Sine Of Angle Of Attack</td>
<td>ANU</td>
<td>-1.0</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>(\cos \alpha)</td>
<td>CALPH</td>
<td>cosineAngleOfAttack</td>
<td>Cosine Of Angle Of Attack</td>
<td>ANU</td>
<td>-1.0</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>(\sin \beta)</td>
<td>SBETA</td>
<td>sineAngleOfSideslip</td>
<td>Sine Of Sideslip Angle</td>
<td>ANL</td>
<td>-1.0</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>(\cos \beta)</td>
<td>CBETA</td>
<td>cosineAngleOfSideslip</td>
<td>Cosine Of Sideslip Angle</td>
<td>ANL</td>
<td>-1.0</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>(V_{\text{cal}})</td>
<td>VCAL</td>
<td>calibratedAirspeed_nmi_h</td>
<td>Calibrated Air Speed, knots</td>
<td>FWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(V_{\text{eq}})</td>
<td>VEQ</td>
<td>equivalentAirspeed_nmi_h</td>
<td>Equivalent Air Speed</td>
<td>FWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(V_{\text{ind}})</td>
<td>VCAL</td>
<td>indicatedAirspeed_nmi_h</td>
<td>Calibrated Air Speed, FWD</td>
<td>FWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(V)</td>
<td>VRW</td>
<td>trueAirspeed_ft_s</td>
<td>Vehicle Velocity relative to the local wind (true airspeed)</td>
<td>FWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>trueAirspeed_m_s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>trueAirspeed_nmi_h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(q)</td>
<td>QBAR</td>
<td>dynamicPressure_lbf_ft2</td>
<td>Dynamic Pressure</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>dynamicPressure_N_m2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>((q))</td>
<td>QBARC</td>
<td>impactPressure_lbf_ft2</td>
<td>Impact Pressure</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>impactPressure_N_m2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\rho)</td>
<td>RHO</td>
<td>airDensity_lbm_ft3</td>
<td>Air Density, At Altitude of the aircraft</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>airDensity_kg_m3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DENALT</td>
<td>densityAltitude_ft</td>
<td>Density altitude</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>densityAltitude_m</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>SOUND</td>
<td>speedOfSound_ft_s</td>
<td>Velocity Of Sound At Altitude of the aircraft</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>speedOfSound_m_s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TR</td>
<td>TR</td>
<td>totalTempRatio</td>
<td>Total Temperature Ratio</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>totalTempRatio</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PR</td>
<td>PR</td>
<td>totalPressureRatio</td>
<td>Total Pressure Ratio</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>totalPressureRatio</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAM</td>
<td>TAM</td>
<td>ambientTemperature_dgC</td>
<td>Ambient Temperature at altitude</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ambientTemperature_dgK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAMB</td>
<td>PAMB</td>
<td>ambientPressure_lbf_ft2</td>
<td>Ambient Pressure at altitude</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ambientPressure_lbf_m2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAMB</td>
<td>PAMB</td>
<td>ambientPressureRatio</td>
<td>Ratio Of ambient pressure at altitude to sea level ambient pressure</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ambientPressureRatio</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAM</td>
<td>TAM</td>
<td>ambientTemperatureRatio</td>
<td>Ratio Of ambient temperature at altitude to sea level ambient temp</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ambientTemperatureRatio</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TTOT</td>
<td>TTOT</td>
<td>totalTemp_dgC</td>
<td>Total Temperature at altitude</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>totalTemp_dgK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PTOT</td>
<td>PTOT</td>
<td>totalPressure_lbf_ft2</td>
<td>Total Pressure at altitude</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>totalPressure_lbf_m2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAM</td>
<td>TAM</td>
<td>ambientTemperatureAtAlt_dgK</td>
<td>Ambient temperature, at the altitude of the Cm</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ambientTemperatureAtAlt_dgK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ambientTemperatureAtAlt_dgC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>TTOT</strong></td>
<td>Total temperature at the altitude of the Cm</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>totalTemperatureAtAlt_dgK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>totalTemperatureAtAlt_dgR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>totalTemperatureAtAlt_dgC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>ALT_SET</strong></td>
<td>Cockpit Altimeter setting (Kohlsman window)</td>
<td></td>
<td>29.92</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>instrumentAltimeterSetting_inchMerc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>P_ALT</strong></td>
<td>Pressure altitude at the Cm</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>pressureAltitude_ft</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>pressureAltitude_m</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>RHO_SL</strong></td>
<td>Air density at sea level</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>seaLevelAirDensity_lbm_ft3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>TAMB_SL</strong></td>
<td>Ambient temperature at mean sea level</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>seaLevelAmbientTemp_dgK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>seaLevelAmbientTemp_dgC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>PAMB_SL</strong></td>
<td>Ambient pressure at sea level</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>seaLevelAmbientPressure_lbf_ft2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>seaLevelAmbientPressure_N_m2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Atmospheric disturbances and turbulence

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>WIND_SPE ED</strong></td>
<td>Total velocity of mean wind</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>meanAirMassSpeed_ft_s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>meanAirMassSpeed_m_s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>WIND_DIR ECTION</strong></td>
<td>Mean wind direction</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>meanAirMassDirection_deg</td>
<td>Wind blowing FROM North</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>V_w</strong></td>
<td>Total Wind Speed (mean wind plus turbulence)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>totalWindSpeed_ft_s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>totalWindSpeed_m_s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Annex A Page 27 of 54
### Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{WT_x}$</td>
<td>11TurbulenceVelocityWrtGrE ft s$^{-1}$</td>
<td>11TurbulenceVelocityWrtGrE m s$^{-1}$</td>
<td>Vector of locally level coordinate wind turbulence velocities with respect to the geocentric earth (G0) fixed coordinate system.</td>
<td>comprised of the three components as defined below.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note: This vector is designed to be transformed from the GRAM 07 small wind perturbation velocities. GRAM 07 outputs the wind perturbation vector:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$u_{small}$ (+ East)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$v_{small}$ (+ North)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$w_{small}$ (+ UP)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Therefore:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$11turbulenceVelocityWrtGrE_m_x = u_{small}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$11turbulenceVelocityWrtGrE_m_y = v_{small}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$11turbulenceVelocityWrtGrE_m_z = w_{small}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N_{small}$</td>
<td>NSMALL</td>
<td>11TurbulenceVelocityWrtGrE ft s$^{-1}$</td>
<td>X-velocity Turb. Component</td>
<td>to the North</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$E_{small}$</td>
<td>ESMALL</td>
<td>11TurbulenceVelocityWrtGrE ft s$^{-1}$</td>
<td>Y-velocity Turb. Component</td>
<td>to the East</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_{small}$</td>
<td>DSMALL</td>
<td>11TurbulenceVelocityWrtGrE ft s$^{-1}$</td>
<td>Z-velocity Turb. Component</td>
<td>to downward</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$Y_{Turb}$</td>
<td>VELBWT</td>
<td>bodyTurbulenceVelocityWrtGrE ft s$^{-1}$</td>
<td>Vector of body coordinate translational turbulence velocities comprised of the three components as defined below.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$U_{Turb}$</td>
<td>UBTURB</td>
<td>bodyTurbulenceVelocityWrtGrE ft s$^{-1}$</td>
<td>$x$-velocity Turb. Component, Body coordinate</td>
<td>EXO (tailwind)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{Turb}$</td>
<td>VBTURB</td>
<td>bodyTurbulenceVelocityWrtGrE ft s$^{-1}$</td>
<td>$y$-velocity Turb. Component, Body coordinate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Annex A Page 28 of 54
### ANSI/AIAA S-119

#### Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( W_{tur} )</td>
<td>W69TURB</td>
<td>BodyTurbulentVelocityMtrDe f t_x</td>
<td>Z-velocity, Turb. Component, Body coordinate</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( V_{W_M} )</td>
<td>VELMW</td>
<td>VelocityOfMeanAirMassHtDe [f t_x]</td>
<td>Vector of locally level coordinate mean wind translational velocities comprised of the three components as defined below. Note: This vector is designed to be transformed from the GRAM 07 wind model velocities. GRAM 07 outputs the mean wind velocity vector: ( v_{mean} ) (+ East) ( v_{mean} ) (+ North) ( v_{mean} ) (+ UP) Therefore ( \text{VelocityOfMeanAirMassHtDe}<em>{x} ) ( \text{VelocityOfMeanAirMassHtDe}</em>{y} ) ( \text{VelocityOfMeanAirMassHtDe}_{z} ).</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( N_{mean} )</td>
<td>NMEAN</td>
<td>VelocityOfMeanAirMassHtDe [f t_x]</td>
<td>X-velocity Component, locally level coordinate system</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( E_{mean} )</td>
<td>EMEAN</td>
<td>VelocityOfMeanAirMassHtDe [f t_y]</td>
<td>Y-velocity Component, locally level coordinate system</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( D_{mean} )</td>
<td>DMEAN</td>
<td>VelocityOfMeanAirMassHtDe [f t_z]</td>
<td>Z-velocity Component, locally level coordinate system</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Annex A Page 29 of 54
## Annex A: Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{\text{MB}}$</td>
<td>VELMB</td>
<td>bodyVelocityOfMeanAirMachRefDe ft [s]</td>
<td>Vector of body coordinate system mean wind translational wind velocities comprised of the three components as defined below.</td>
</tr>
<tr>
<td>$\omega_{\text{MB}}$</td>
<td>UMEANB</td>
<td>bodyVelocityOfMeanAirMachRefDe_x [s]</td>
<td>X-velocity Component, body coordinate system</td>
</tr>
<tr>
<td>$\omega_{\text{MB}}$</td>
<td>VMEANB</td>
<td>bodyVelocityOfMeanAirMachRefDe_y [s]</td>
<td>Y-velocity Component, body coordinate system</td>
</tr>
<tr>
<td>$\omega_{\text{MB}}$</td>
<td>WMEANB</td>
<td>bodyVelocityOfMeanAirMachRefDe_z [s]</td>
<td>Z-velocity Component, body coordinate system</td>
</tr>
<tr>
<td>$V_{\text{TW}}$</td>
<td>VTW</td>
<td>bodyWindVelocityWrtDe_x [s]</td>
<td>Vector of the body coordinate system of net wind velocities impinging on the aircraft at the Cm. This would include mean air mass motion and any disturbances such as turbulence and shear.</td>
</tr>
<tr>
<td>$U_{\text{TW}}$</td>
<td>UBTWX</td>
<td>bodyWindVelocityWrtDe_x [s]</td>
<td>Net wind velocity impinging on the vehicle in the X body axis.</td>
</tr>
</tbody>
</table>

Note: This vector is designed to be the GRAM 07 wind model mean velocities transformed into the body coordinate system.

Note: Winds are always with respect to and observed from an Earth-fixed reference frame. This vector is with respect to the geocentric Earth (Geofixed coordinate system).
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>V_{\text{bx}}</td>
<td>VBTWY</td>
<td>bodyWindVelocityWtcs_x, y, z</td>
<td>Net wind velocity impinging on the vehicle in the Y body axis. Net wind is the mean air mass plus any turbulence, shears, or other wind disturbances.</td>
<td>RT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V_{\text{bz}}</td>
<td>VBTWZ</td>
<td>bodyWindVelocityWtcs_x, y, z</td>
<td>Net wind velocity impinging on the vehicle in the Z body axis. Net wind is the mean air mass plus any turbulence, shears, or other wind disturbances.</td>
<td>DN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>bodyAngularRateOfTurbulenceWtcs_x, y, z</td>
<td>Vector of angular turbulence velocities comprised of the three body coordinate system components as defined below. Note: Turbulence angular rate is always with respect to and observed from the locally level (L) reference frame.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PTURB</td>
<td></td>
<td>bodyAngularRateOfTurbulenceWtcs_x, y, z</td>
<td>Body coordinate roll turbulence. The turbulence would move the aircraft right wing down.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QTURB</td>
<td></td>
<td>bodyAngularRateOfTurbulenceWtcs_x, y, z</td>
<td>Body coordinate pitch turbulence. The turbulence would move the aircraft nose up.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTURB</td>
<td></td>
<td>bodyAngularRateOfTurbulenceWtcs_x, y, z</td>
<td>Body coordinate yaw turbulence. The turbulence would move the aircraft nose left.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Annex A. Standard Variable Names

**Table A.6 — Vehicle physical characteristics**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>l</td>
<td></td>
<td>bodyMomentOfInertiaSlugf2 [3,3]</td>
<td>Matrix of the total moments of inertia of the aircraft. This is with respect to the Cm and includes everything in or attached to the aircraft (stores, passengers, crew, fuel, etc.). It is comprised of the components below.</td>
<td></td>
<td>l1x, l1y, l1z, l2x, l2y, l2z</td>
<td></td>
<td></td>
</tr>
<tr>
<td>kx</td>
<td>XIXX</td>
<td>bodyMomentOfInertiaSlugf2_X</td>
<td>Vehicle Roll Moment Of Inertia about Cm, body coordinate system</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ky</td>
<td>XIYY</td>
<td>bodyMomentOfInertiaSlugf2_Y</td>
<td>Vehicle Pitch Moment Of Inertia about Cm, body coordinate system</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kz</td>
<td>XIZZ</td>
<td>bodyMomentOfInertiaSlugf2_Z</td>
<td>Vehicle Yaw Moment Of Inertia about Cm, body coordinate system</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bx</td>
<td>XIXZ</td>
<td>bodyProductOfInertiaSlugf2_EX</td>
<td>Vehicle ZX Cross Product Of Inertia about Cm, body coordinate system</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>by</td>
<td>XIXY</td>
<td>bodyProductOfInertiaSlugf2_XY</td>
<td>Vehicle XY Cross Product Of Inertia about Cm, body coordinate system</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bz</td>
<td>XIXZ</td>
<td>bodyProductOfInertiaSlugf2_YZ</td>
<td>Vehicle YZ Cross Product Of Inertia about Cm, body coordinate system</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MrcPos</td>
<td>vrsPositionOfMrc_ft</td>
<td>vrsPositionOfMrc_m</td>
<td>Vector of the location of the moment reference center (Mrc) of the aircraft in the vehicle reference system (vrs). Comprised of the three components as defined below. This vector is used to define the fixed physical location of the moment reference center in the vehicle. Note that the vrs definition is specific to each vehicle.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XMrC</td>
<td>vrsPositionOfMrc_ft</td>
<td>vrsPositionOfMrc_m</td>
<td>X Mrc Position</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YMrC</td>
<td>vrsPositionOfMrc_ft</td>
<td>vrsPositionOfMrc_m</td>
<td>Y Mrc Position</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZMrC</td>
<td>vrsPositionOfMrc_ft</td>
<td>vrsPositionOfMrc_m</td>
<td>Z Mrc position</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeltaA</td>
<td>DCG</td>
<td>bodyPositionOfCmMrcMrc_ft</td>
<td>Vector of the location of the Cm with respect to the moment reference center (Mrc) of the aircraft in the body coordinate system. Comprised of the three components as defined below. This vector is used to define the location of the Cm which moves with the fixed moment reference center in the vehicle. Note that vrsPositionOfMrc locates the Mrc in the vehicle.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeltaX</td>
<td>DXCG</td>
<td>bodyPositionOfCmMrcMrc_ft_X</td>
<td>X Cm position</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeltaY</td>
<td>DYCG</td>
<td>bodyPositionOfCmMrcMrc_ft_Y</td>
<td>Y Cm position</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeltaZ</td>
<td>DZCG</td>
<td>bodyPositionOfCmMrcMrc_ft_Z</td>
<td>Z Cm position</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>XMASS</td>
<td>totalMass_slug/totalMass_kg</td>
<td>Total Mass Of Vehicle (including Fuel, crew, cargo, stores, passengers, etc.)</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>WEIGHT</td>
<td>grossWeight_lbf</td>
<td>Aircraft Gross Weight (mass*gravity), including all fuel, occupants, stores, etc.</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>grossWeight_N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>AREA</td>
<td>referenceWingArea_ft2</td>
<td>Reference Wing Area</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>referenceWingArea_n2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>SPAN</td>
<td>referenceWingSpan_ft</td>
<td>Reference Wing Span</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>referenceWingSpan_n</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>CHORD</td>
<td>referenceWingChord_ft</td>
<td>Mean Aerodynamic Chord (reference wing chord)</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>referenceWingChord_n</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>engineMomentOfInertia_slugft2[1,3]</td>
<td>Matrix of the moments of inertia of the Rotating engine, for an engine with the propeller, includes the propeller and drive train. This is w.r.t. the rotational axis of the engine. For multi-engine vehicles is for one engine. It is comprised of the components below.</td>
<td></td>
<td>leXX</td>
<td>leXY</td>
<td>leXZ</td>
</tr>
<tr>
<td></td>
<td></td>
<td>engineMomentOfInertia_hgm2[1,3]</td>
<td></td>
<td></td>
<td>leXX</td>
<td>leXY</td>
<td>leXZ</td>
</tr>
<tr>
<td></td>
<td>cIEXX</td>
<td>engineMomentOfInertia_slugft2_X</td>
<td>Moment of inertia about the X-axis Of Rotating Eng, for an engine with the propeller, includes the propeller</td>
<td></td>
<td>leXX</td>
<td>leXY</td>
<td>leXZ</td>
</tr>
<tr>
<td></td>
<td></td>
<td>engineMomentOfInertia_hgm2_X</td>
<td>This is w.r.t. the rotational axis of the engine</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n_e )</td>
<td>iEYY</td>
<td>engineMomentOfInertia_slugft2_Y engineMomentOfInertia_kgm2_Y</td>
<td>Moment of inertia about the Y-axis Of Rotating Eng, for an engine with the propeller, includes the propeller This is w.r.t. the rotational axis of the engine</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( n_e )</td>
<td>iEZZ</td>
<td>engineMomentOfInertia_slugft2_Z engineMomentOfInertia_kgm2_Z</td>
<td>Moment of inertia about the Z-axis Of Rotating Eng, for an engine with the propeller, includes the propeller This is w.r.t. the rotational axis of the engine</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \omega_e )</td>
<td>iEXZ</td>
<td>engineProductOfInertia_slugft2_KZ engineProductOfInertia_kgm2_KZ</td>
<td>Product of inertia about the XZ-axis Of Rotating Eng, for an engine with the propeller, includes the propeller This is w.r.t. the rotational axis of the engine</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \omega_e )</td>
<td>iEXY</td>
<td>engineProductOfInertia_slugft2_KY engineProductOfInertia_kgm2_KY</td>
<td>Product of inertia about the XY-axis Of Rotating Eng, for an engine with the propeller, includes the propeller This is w.r.t. the rotational axis of the engine</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Annex A Page 35 of 54
### Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_{EY}$</td>
<td>IYEZ</td>
<td><code>engineProductOfInertia_slugft2_YE</code>&lt;br&gt;<code>engineProductOfInertia_kgn2_YE</code></td>
<td>Product of inertia about the YZ-axis of Rotating Eng. for an engine with the propeller, includes the propeller. This is w.r.t. the rotational axis of the engine.</td>
</tr>
<tr>
<td><code>fuelInTank_lbm[number of fuel tanks]</code>&lt;br&gt;<code>fuelInTank_kg[number of fuel tanks]</code></td>
<td></td>
<td>Vector of fuel weight by tank. Each aircraft tank is normally numbered and the vector should be ordered according to fuel tank number. In the absence of tank numbering the convention of port to starboard, upper to lower, then front to rear should be used.</td>
<td></td>
</tr>
<tr>
<td><code>fuelTankCentroid_ft[number of fuel tanks,3]</code>&lt;br&gt;<code>fuelTankCentroid_m[number of fuel tanks,3]</code></td>
<td></td>
<td>Matrix used to locate the centroids of the fuel tanks. Each aircraft tank is normally numbered and the matrix should be ordered according to fuel tank number. The second component is the x, y and z moment arms from the moment reference center to the tank centroid in the structural coordinate system. In the absence of tank numbering the convention of port to starboard, upper to lower, then front to rear should be used.</td>
<td></td>
</tr>
</tbody>
</table>

### Table A.7 — Vehicle control position

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
</table>

Annex A Page 36 of 54
### Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_{\text{h}}$</td>
<td>pilotControlPos_deg_long</td>
<td>Longitudinal control position of the pilot.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pilotControlPos_rad_long</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pilotControlRate_deg_e_long</td>
<td>Rate of the pilot Longitudinal control movement.</td>
<td></td>
<td>Moving aft</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pilotControlRate_rad_s_long</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pilotControlAccel_deg_e2_long</td>
<td>Acceleration of the pilot Longitudinal control movement.</td>
<td></td>
<td>Accelerating aft</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pilotControlAccel_rad_e2_long</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pilotControlForce_lbf_long</td>
<td>Longitudinal control force of the pilot.</td>
<td></td>
<td>Alt force</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pilotControlForce_N_long</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The convention and examples above apply to all pilot and copilot controls defined below.
### Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
</table>
| $D_h$  | pilotControlPos_deg_lat  
       |             | pilotControlPos_rad_lat | Lateral control position of the pilot. | LEFT  
   0° = vertical when vehicle body is level (pitch and roll attitude = 0) |                |           |           |
| $D_h$  | pilotControlPos_deg_avgPedal  
       |             | pilotControlPos_rad_avgPedal | Net Directional control position of the pilot. Normally, left pedal = right pedal. | Left Pedal in or counter clockwise twist of a sidestick |                |           |           |
| $D_h$  | pilotControlPos_deg_ftPedal  
       |             | pilotControlPos_rad_ftPedal | Right Directional control position of the pilot. | NEGATIVE for Pedal in  
   0° = pedal full aft |                |           |           |
| $D_h$  | pilotControlPos_deg_ftPedal  
       |             | pilotControlPos_rad_ftPedal | Left Directional control position of the pilot. | Pedal in.  
   0° = pedal full aft |                |           |           |
| $D_v$  | pilotControlPos_deg_collective  
       |             | pilotControlPos_rad_collective | Pilot collective control position. | UP  
   0 = full down |                |           |           |
| $D_v$  | pilotControlPos_deg_avgThrottle  
       |             | pilotControlPos_rad_avgThrottle | Average pilot throttle control position. | FWD  
   On full aft w/o thrust reversing  
   Thrust reversing is negative |                |           |           |
| $D_v$  | pilotControlPos_deg_throttle [number of engines]  
       |             | pilotControlPos_rad_throttle [number of engines] | Individual pilot throttle control positions. Order is outboard port (left) to outboard starboard. | FWD  
   On full aft w/o thrust reversing,  
   Thrust reversing is negative |                |           |           |
### Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>copilotControlPos_deg_long</td>
<td></td>
<td>Longitudinal control position of the copilot.</td>
<td>AFT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>copilotControlPos_rad_long</td>
<td></td>
<td>R= vertical when vehicle body is level (pitch and roll attitude = 0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>copilotControlPos_deg_lat</td>
<td></td>
<td>Lateral control position of the copilot.</td>
<td>LEFT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>copilotControlPos_rad_lat</td>
<td></td>
<td>L= vertical when vehicle body is level (pitch and roll attitude = 0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>copilotControlPos_deg_throttle</td>
<td></td>
<td>Net Directional control position of the copilot. Normally, Left pedal + right pedal.</td>
<td>Left Pedal in or counter clockwise twist of a sidestick</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>copilotControlPos_rad_throttle</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>copilotControlPos_deg_pedal</td>
<td></td>
<td>Right Directional control position of the copilot.</td>
<td>NEGATIVE for Pedal in</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>copilotControlPos_rad_pedal</td>
<td></td>
<td>0= pedal full aft</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>copilotControlPos_deg_collective</td>
<td></td>
<td>Left Directional control position of the copilot.</td>
<td>Pedal in</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>copilotControlPos_rad_collective</td>
<td></td>
<td>0= pedal full aft</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>copilotControlPos_deg_throttle</td>
<td></td>
<td>Copilot collective control position.</td>
<td>UP</td>
<td>0= full down</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>copilotControlPos_rad_throttle</td>
<td></td>
<td>FWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Average copilot throttle control position.</td>
<td>FWD= full aft w/ thrust reversing; Thrust reversing = negative</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>copilotControlPos_deg_throttle [number of engines]</td>
<td>Individual copilot throttle control positions. Order is outboard port (left) to outboard starboard.</td>
<td>FWD 0= full aft w/o thrust reversing, Thrust reversing is negative</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>copilotControlPos_rad_throttle [number of engines]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>controlPos_deg_avgThrottle</td>
<td>Average pilot and copilot throttle control position.</td>
<td>FWD 0= full aft w/o thrust reversing, Thrust reversing is negative</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>controlPos_rad_avgThrottle</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>controlPos_deg_avgPropeller</td>
<td>Average pilot and copilot propeller blade pitch control position.</td>
<td>FWD Offset pitch, Thrust reversing is negative</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>controlPos_rad_avgPropeller</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>controlPos_deg_propeller [number of engines]</td>
<td>Individual propeller blade pitch control position. Order is outboard port (left) to outboard starboard.</td>
<td>FWD Offset pitch, Thrust reversing is negative</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>controlPos_rad_propeller [number of engines]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Annex A. Standard Variable Names

#### Standard naming convention for control surfaces
- "Pos" indicates a control surface position
- "Rate" indicates the derivative of the control surface position
- "Accel" indicates the derivative of the control surface rate
- "HingeMoment" indicates the hinge moment on the control surface (sign convention = + deflection results in + hinge moment)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>θ_{ub}</td>
<td></td>
<td>controlSurfacePos_deg_TEF [number of trailing edge flap control surfaces]</td>
<td>Vector of trailing edge flap positions, one for each surface deflected. Order is outboard port (left) to outboard starboard.</td>
<td>TED</td>
<td>0 defection is flap retracted</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>controlSurfacePos_rad_TEF [number of trailing edge flap control surfaces]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>controlSurfaceRate_deg_e_TEF [number of trailing edge flap control surfaces]</td>
<td>Vector of trailing edge flap deflection rates, one for each surface. Order is outboard port (left) to outboard starboard.</td>
<td>TED</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>controlSurfaceRate_rad_e_TEF [number of trailing edge flap control surfaces]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>controlSurfaceAccel_deg_a1_TEF [number of trailing edge flap control surfaces]</td>
<td>Vector of trailing edge flap deflection accelerations, one for each surface. Order is outboard port (left) to outboard starboard.</td>
<td>TED</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>controlSurfaceAccel_rad_a1_TEF [number of trailing edge flap control surfaces]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Annex A Page 41 of 54
## Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \delta_{\text{TEF}} )</td>
<td>controlSurfacePos_rad_TEF</td>
<td>controlSurfaceHingMoment_rad_TEF ( \text{ft-lbf_TEF} )</td>
<td>Vector of the hinge moments on the trailing edge flap, one for each surface. Order is outboard port (left) to outboard starboard.</td>
<td>+ = TEU moment (positive deflection results in positive moment)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \delta_{\text{TEF}} )</td>
<td>controlSurfacePos_deg_TEF</td>
<td>controlSurfaceHingMoment_ft_lbf_TEF ( \text{number of trailing edge flap control surfaces} )</td>
<td>Vector of the hinge moments on the trailing edge flap, one for each surface. Order is outboard port (left) to outboard starboard.</td>
<td>+ = TEU moment (positive deflection results in positive moment)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \delta_{\text{TEF}} )</td>
<td>controlSurfacePos_rad_avg_TEF</td>
<td>controlSurfacePos_deg_avg_TEF</td>
<td>Trailing edge flap deflection (TEF). Average for all trailing edge flap surfaces.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \delta_{\text{LEF}} )</td>
<td>controlSurfacePos_rad_diff_TEF</td>
<td>controlSurfacePos_rad_avg_TEF</td>
<td>Differential trailing edge flap (DTEF) deflection (left deflections (+=TED) - right deflections (+=TRED))</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \Delta_{\text{LEF}} )</td>
<td>controlSurfacePos_rad_LEP</td>
<td>controlSurfacePos_rad_LEP</td>
<td>Vector of leading edge flap (LEF) deflection, one for each surface deflected. Order is outboard port (left) to outboard starboard.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \Delta_{\text{LEF}} )</td>
<td>controlSurfacePos_rad_avg_LEF</td>
<td>controlSurfacePos_rad_avg_LEF</td>
<td>Leading edge flap deflection. Average for all deflected leading edge flap/slat surfaces.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The convention and examples above apply to all control surface positions defined below.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta_{\text{LEF}}$</td>
<td></td>
<td>controlSurfacePos_deg_diffLEF</td>
<td>Differential leading edge flap (LEF) deflection (left deflections-right deflections)</td>
<td>LT LED (RWD moment)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta_{\text{spoiler}}$</td>
<td></td>
<td>controlSurfacePos_deg_spoiler [number of spoiler control surfaces]</td>
<td>Vector of spoiler control surface positions, one for each surface deflected. Order is outboard port (left) to outboard starboard.</td>
<td>TEU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta_{\text{spoiler}}$</td>
<td></td>
<td>controlSurfacePos_rad_spoiler [number of spoiler control surfaces]</td>
<td>Vector of spoiler control surface positions, one for each surface deflected. Order is outboard port (left) to outboard starboard.</td>
<td>TEU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta_{\text{spoiler}}$</td>
<td></td>
<td>controlSurfacePos_avg_spoiler</td>
<td>Spoiler deflection. Average for all deflected spoilers (sum of positions/number of surfaces)</td>
<td>TEU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta_{\text{spoiler}}$</td>
<td></td>
<td>controlSurfacePos_rad_avg_spoiler</td>
<td>Vector of spoiler control surface positions, one for each surface deflected. Order is outboard port (left) to outboard starboard.</td>
<td>TEU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta_{\text{aeron}}$</td>
<td></td>
<td>controlSurfacePos_deg_diff_aeron [number of aileron control surfaces]</td>
<td>Vector of aileron control positions, one for each surface deflected. Order is outboard port (left) to outboard starboard.</td>
<td>TEU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta_{\text{aeron}}$</td>
<td></td>
<td>controlSurfacePos_rad_diff_aeron</td>
<td>Differential aileron deflection, (right deflections-left deflections)</td>
<td>Right aileron TEU (RWD moment)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta_{\text{aeronTab}}$</td>
<td></td>
<td>controlSurfacePos_deg_diff_aeronTab [number of aileron control surfaces, number of tabs]</td>
<td>Array of aileron tab control positions (i) for each surface deflected (n). Order is outboard port to outboard starboard</td>
<td>TEU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta_{\text{aeronTab}}$</td>
<td></td>
<td>controlSurfacePos_rad_diff_aeronTab</td>
<td>Array of aileron tab control positions (i) for each surface deflected (n). Order is outboard port to outboard starboard</td>
<td>TEU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta_{\text{average}}$</td>
<td></td>
<td>controlSurfacePos_avg_aeronTab</td>
<td>Average aileron tab deflection (sum of positions/number of surfaces)</td>
<td>TEU</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta_{\text{aileron}}$</td>
<td>controlSurfacePos_deg_diffAileronTab</td>
<td>controlSurfacePos_rad_diffAileronTab</td>
<td>Differential aileron tab deflection (right tab deflections-left tab deflections)</td>
<td>RT Aileron Tab (TEU)</td>
<td>TEL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta_{\text{rudder}}$</td>
<td>controlSurfacePos_deg_rudder [number of rudder control surfaces]</td>
<td>controlSurfacePos_rad_rudder [number of rudder control surfaces]</td>
<td>Vector of rudder control positions, one for each surface deflected. Order is outboard port (left) to outboard starboard</td>
<td>TEU</td>
<td>TEL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta_{\text{avg}}$</td>
<td>controlSurfacePos_deg_avgRudder</td>
<td>controlSurfacePos_rad_avgRudder</td>
<td>Average rudder deflection (sum of positions/number of surfaces)</td>
<td>TEU</td>
<td>TEL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta_{\text{diff Rudder}}$</td>
<td>controlSurfacePos_deg_diffRudder</td>
<td>controlSurfacePos_rad_diffRudder</td>
<td>Differential rudder deflection (right deflections-left deflections)</td>
<td>RT Rudder (ANR moment)</td>
<td>TEL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta_{\text{rudderTab}}$</td>
<td>controlSurfacePos_deg_rudderTab [number of rudder control surfaces, number of tabs]</td>
<td>controlSurfacePos_rad_rudderTab [number of rudder control surfaces, number of tabs]</td>
<td>Array of rudder tab control positions (i) for each surface deflected (n). Order of tabs is upper to lower.</td>
<td>TEL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta_{\text{avg RudderTab}}$</td>
<td>controlSurfacePos_deg_avgRudderTab</td>
<td>controlSurfacePos_rad_avgRudderTab</td>
<td>Average rudder tab deflection (sum of positions/number of surfaces)</td>
<td>RT Rudder Tab (ANR moment)</td>
<td>TEL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta_{\text{diff RudderTab}}$</td>
<td>controlSurfacePos_deg_diffRudderTab</td>
<td>controlSurfacePos_rad_diffRudderTab</td>
<td>Differential rudder tab deflection (right deflections-left deflections)</td>
<td>RT Rudder Tab (ANR moment)</td>
<td>TEL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta_{e_1}$</td>
<td>controlSurfacePos_deg_elevator</td>
<td>controlSurfacePos_deg_elevator (number of elevator control surfaces)</td>
<td>Vector of elevator (or stabilizer/stabilator) control positions, one for each surface deflected. Order is outboard port (left) to outboard starboard.</td>
<td>TEU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta_{e}$</td>
<td>controlSurfacePos_avgElevator</td>
<td>controlSurfacePos_avgElevator</td>
<td>Average elevator (or stabilizer/stabilator) deflection (sum of positions/number of surfaces)</td>
<td>TEU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta_{e_2}$</td>
<td>controlSurfacePos_diffElevator</td>
<td>controlSurfacePos_diffElevator</td>
<td>Differential elevator deflection (right deflections-left deflections)</td>
<td>Right control</td>
<td>TEU</td>
<td>(RMD moment)</td>
<td></td>
</tr>
<tr>
<td>$\delta_{e_{3,ij}}$</td>
<td>controlSurfacePos_deg_elevatorTab</td>
<td>controlSurfacePos_deg_elevatorTab (number of elevator tab control surfaces, number of tabs)</td>
<td>Array of elevator (or stabilizer/stabilator) tab positions (i) for each surface (n). Order is outboard port (left) to outboard starboard per control surface.</td>
<td>TEU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta_{e_{avg}}$</td>
<td>controlSurfacePos_avgElevatorTab</td>
<td>controlSurfacePos_avgElevatorTab (number of elevator tab control surfaces, number of tabs)</td>
<td>Average elevator (or stabilizer/stabilator) tab control surface positions (sum of positions/number of surfaces)</td>
<td>TEU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta_{e_{diff}}$</td>
<td>controlSurfacePos_diffElevatorTab</td>
<td>controlSurfacePos_diffElevatorTab</td>
<td>Average differential elevator (or stabilizer/stabilator) tab deflection (right deflections-left deflections)</td>
<td>Right control</td>
<td>TEU</td>
<td>(RMD moment)</td>
<td></td>
</tr>
</tbody>
</table>

Annex A Page 45 of 54
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>δc</td>
<td>controlSurfacePos_deg_canard [number of canard control surfaces]</td>
<td>Vector of canard control positions, one for each surface. Order is outboard port (left) to outboard starboard.</td>
<td>TED</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>δc</td>
<td>controlSurfacePos_rad_canard</td>
<td></td>
<td>TED</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>δc</td>
<td>controlSurfacePos_deg_avgCanard</td>
<td>Average canard position (sum of positions/number of surfaces)</td>
<td>TED</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>δc</td>
<td>controlSurfacePos_rad_avgCanard</td>
<td></td>
<td>TED</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>δC</td>
<td>controlSurfacePos_deg_diffCanard</td>
<td>Average differential canard deflection (LEFT deflections - RIGHT deflections)</td>
<td>LEFT control TED (RMD moment)</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>δC</td>
<td>controlSurfacePos_rad_diffCanard</td>
<td></td>
<td></td>
<td>TED</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>δΩ</td>
<td>controlSurfacePos_deg_canardTab [number of canard tab control surfaces, number of tabs]</td>
<td>Array of canard tab positions (i) for each surface (n). Order is outboard port (left) to outboard starboard per control surface.</td>
<td>TED</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>δΩ</td>
<td>controlSurfacePos_rad_canardTab</td>
<td></td>
<td>TED</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>δΩ</td>
<td>controlSurfacePos_deg_avgCanardTab</td>
<td>Average canard tab deflection (sum of positions/number of surfaces)</td>
<td>TED</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>δΩ</td>
<td>controlSurfacePos_rad_avgCanardTab</td>
<td></td>
<td>TED</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>δΩ</td>
<td>controlSurfacePos_deg_diffCanardTab</td>
<td>Average differential canard tab deflection (LEFT deflections - RIGHT deflections)</td>
<td>LEFT control TED (RMD moment)</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>δΩ</td>
<td>controlSurfacePos_rad_diffCanardTab</td>
<td></td>
<td>TED</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>δS</td>
<td>controlSurfacePos_deg_speedbrake</td>
<td>Speedbrake deflection</td>
<td>Extended</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>δS</td>
<td>controlSurfacePos_rad_speedbrake</td>
<td></td>
<td></td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>δG</td>
<td>landingGearPosition [number of landing gear struts]</td>
<td>Vector of landing gear positions, one for each strut. Order is outboard port (left) to outboard starboard.</td>
<td>TED</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>δG</td>
<td></td>
<td>0= up and locked; 1= full extension with no weight on wheels</td>
<td></td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Annex A Page 46 of 54
## Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>landingGearWeightOnWheels_1BF [number of landing gear struts]</td>
<td>landingGearWeightOnWheels_1BF</td>
<td>Vector of landing gear weight on wheels, one for each strut. Order is outboard port (left) to outboard starboard.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>landingGearWeightOnWheels_N [number of landing gear struts]</td>
<td>landingGearWeightOnWheels_N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>landingGearWheelSpeed_rad_s [number of landing gear struts, number of trucks, number of wheels per truck]</td>
<td>landingGearWheelSpeed_rad_s</td>
<td>Array of landing gear wheel speeds by strut, one for each strut. Order of struts is outboard port (left) strut, to outboard starboard. Order of trucks is front to rear. Order of wheels on each truck is port to starboard.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table A.8 — Vehicle aerodynamic characteristics

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cl</td>
<td>CL</td>
<td>totalCoefficientOfLift</td>
<td>Coefficient Of Lift, Total, includes effects of stores</td>
<td>UP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cd</td>
<td>CD</td>
<td>totalCoefficientOfDrag</td>
<td>Coefficient Of Drag, Total, includes effects of stores</td>
<td>AFT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>aeroBodyForceCoefficient{x}</td>
<td>Vector of total aerodynamic force coefficients in the body coordinate system, comprised of the three components as defined below.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CX</td>
<td>CX</td>
<td>aeroBodyForceCoefficient_X</td>
<td>X-body Force Coefficient due to aerodynamic loads, includes stores (Body coordinate)</td>
<td>FWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CY</td>
<td>CY</td>
<td>aeroBodyForceCoefficient_Y</td>
<td>Y-body Force Coefficient due to aerodynamic loads, includes stores (Body coordinate)</td>
<td>RT</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Annex A: Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cz</td>
<td>CZ</td>
<td>aeroBodyForceCoefficient_Z</td>
<td>Z-body Force Coefficient due to aerodynamic loads, includes stores (Body coordinate)</td>
<td>DN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>aeroBodyForce_lbf [3]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>aeroBodyForce_N [3]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAx</td>
<td>FAX</td>
<td>aeroBodyForce_lbf_X</td>
<td>Total X-body Force due to aerodynamic loads, includes stores (Body coordinate)</td>
<td>FWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>aeroBodyForce_N_X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAY</td>
<td>FAY</td>
<td>aeroBodyForce_lbf_Y</td>
<td>Total Y-body Force due to aerodynamic loads, includes stores (Body coordinate)</td>
<td>RT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>aeroBodyForce_N_Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAZ</td>
<td>FAZ</td>
<td>aeroBodyForce_lbf_Z</td>
<td>Total Z-body Force due to aerodynamic loads, includes stores (Body coordinate)</td>
<td>DN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>aeroBodyForce_N_Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>thrustBodyForce_lbf [3]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>thrustBodyForce_N [3]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FEX</td>
<td>FEX</td>
<td>thrustBodyForce_lbf_X</td>
<td>Total net engine thrust Force, X-body axis</td>
<td>FWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>thrustBodyForce_N_X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FEY</td>
<td>FEY</td>
<td>thrustBodyForce_lbf_Y</td>
<td>Total net engine thrust Force, Y-body axis</td>
<td>RT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>thrustBodyForce_N_Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FEZ</td>
<td>FEZ</td>
<td>thrustBodyForce_lbf_Z</td>
<td>Total net engine thrust Force, Z-body axis</td>
<td>DN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>thrustBodyForce_N_Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>gearBodyForce_Lbf</td>
<td>[3]</td>
<td>gearBodyForce_Lbf [3]</td>
<td>Vector of total landing gear ground reaction forces in the body coordinate system. Does NOT include aerodynamic forces on the landing gear that are included in aeroBodyForce defined above. Comprised of the three components as defined below.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FGX</td>
<td>gearBodyForce_Lbf_X</td>
<td>Total landing gear reaction force, X-body axis</td>
<td>FWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FGX</td>
<td>gearBodyForce_Lbf_N_X</td>
<td>Total landing gear reaction force, X-body axis</td>
<td>FWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FGY</td>
<td>gearBodyForce_Lbf_Y</td>
<td>Total landing gear reaction force, Y-body axis</td>
<td>RT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FGY</td>
<td>gearBodyForce_Lbf_N_Y</td>
<td>Total landing gear reaction force, Y-body axis</td>
<td>RT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FGZ</td>
<td>gearBodyForce_Lbf_Z</td>
<td>Total landing gear reaction force, Z-body axis</td>
<td>DN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FGZ</td>
<td>gearBodyForce_Lbf_N_Z</td>
<td>Total landing gear reaction force, Z-body axis</td>
<td>DN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>totalBodyForce_Lbf</td>
<td>[3]</td>
<td>Vector of total forces in the body coordinate system. Includes all forces exerted upon the aircraft. Comprised of the three components as defined below.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>totalBodyForce_Lbf_X</td>
<td>totalBodyForce_Lbf_X</td>
<td>Total Forces On a/c, X-body axis</td>
<td>PWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>totalBodyForce_Lbf_N_X</td>
<td>totalBodyForce_Lbf_N_X</td>
<td>Total Forces On a/c, X-body axis</td>
<td>PWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>totalBodyForce_Lbf_Y</td>
<td>totalBodyForce_Lbf_Y</td>
<td>Total Forces On a/c, Y-body axis</td>
<td>RT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>totalBodyForce_Lbf_N_Y</td>
<td>totalBodyForce_Lbf_N_Y</td>
<td>Total Forces On a/c, Y-body axis</td>
<td>RT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>totalBodyForce_Lbf_Z</td>
<td>totalBodyForce_Lbf_Z</td>
<td>Total Forces On a/c, Z-body axis</td>
<td>DN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>totalBodyForce_Lbf_N_Z</td>
<td>totalBodyForce_Lbf_N_Z</td>
<td>Total Forces On a/c, Z-body axis</td>
<td>DN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>aeroBodyMomentCoefficient</td>
<td>aeroBodyMomentCoefficient [3]</td>
<td>Vector of total aerodynamic moment coefficients in the body coordinate system, including stores. Comprised of the three components as defined below.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>CGL</td>
<td>Total Aerodynamic Rolling Moment Coefficient including stores. Moment about the X-body axis</td>
<td>RWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Annex A. Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C&lt;sub&gt;0&lt;/sub&gt;</td>
<td>CLM</td>
<td>aeroBodyMomentCoefficient_Pitch</td>
<td>Total Aerodynamic Pitching Moment Coefficient, including stores. Moment about the Y-body axis</td>
<td>ANU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C&lt;sub&gt;0 &lt;/sub&gt;</td>
<td>CLN</td>
<td>aeroBodyMomentCoefficient_Yaw</td>
<td>Total Aerodynamic Yawing Moment Coefficient, including stores. Moment about the Z-body axis</td>
<td>ANR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>aeroBodyMoment_ftlb [3]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>aeroBodyMoment_Ns [3]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C&lt;sub&gt;N&lt;/sub&gt;</td>
<td>TAL</td>
<td>aeroBodyMoment_ftlb_Roll</td>
<td>Total Aerodynamic Rolling moment (including attached stores), about the X-body axis</td>
<td>RWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>aeroBodyMoment_Ns_Roll</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M&lt;sub&gt;N&lt;/sub&gt;</td>
<td>TAM</td>
<td>aeroBodyMoment_ftlb_Pitch</td>
<td>Total Aerodynamic pitching moment (including attached stores), about the Y-body axis</td>
<td>ANU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>aeroBodyMoment_Ns_Pitch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N&lt;sub&gt;N&lt;/sub&gt;</td>
<td>TAN</td>
<td>aeroBodyMoment_ftlb_Yaw</td>
<td>Total Aerodynamic yawing moment (including attached stores), about the Z-body axis</td>
<td>ANR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>aeroBodyMoment_Ns_Yaw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>thrustBodyMoment_ftlb [3]</td>
<td>Vector of total net propulsion system moments in the body coordinate system (includes installation losses, inlet efficiency and propeller efficiency). Referenced to the moment reference center. Comprised of the three components as defined below.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>thrustBodyMoment_Ns [3]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L&lt;sub&gt;E&lt;/sub&gt;</td>
<td>TEL</td>
<td>thrustBodyMoment_ftlb_Roll</td>
<td>Total Engine Rolling Moment, about the X-body axis</td>
<td>RWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>thrustBodyMoment_Ns_Roll</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M&lt;sub&gt;E&lt;/sub&gt;</td>
<td>TEM</td>
<td>thrustBodyMoment_ftlb_Pitch</td>
<td>Total Engine Pitching Moment, about the Y-body axis</td>
<td>ANU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>thrustBodyMoment_Ns_Pitch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Annex A: Standard Variable Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_t$</td>
<td>TEN</td>
<td>thrustBodyMoment_ftlbf_Yaw&lt;br&gt;thrustBodyMoment_Nm_Yaw</td>
<td>Total Engine yawing Moment, about the X-body axis</td>
<td>ANR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>landingGearBodyMoment_ftlbf [3]&lt;br&gt;landingGearBodyMoment_Nm [3]</td>
<td>Vector of total landing gear ground reaction moments in the body coordinate system. Referenced to the moment reference center. Does NOT include aerodynamic moments on the landing gear that are included in aerBodyMoment defined above. Comprised of the three components as defined below.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$L_c$</td>
<td>TOL</td>
<td>landingGearBodyMoment_ftlbf_Roll&lt;br&gt;landingGearBodyMoment_Nm_Roll</td>
<td>Total Landing Gear Rolling Moment, about the X-body axis</td>
<td>RWD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_c$</td>
<td>TGM</td>
<td>landingGearBodyMoment_ftlbf_Pitch&lt;br&gt;landingGearBodyMoment_Nm_Pitch</td>
<td>Total Landing Gear Pitch Moment, about the Y-body axis</td>
<td>ANU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N_c$</td>
<td>TGN</td>
<td>landingGearBodyMoment_ftlbf_Yaw&lt;br&gt;landingGearBodyMoment_Nm_Yaw</td>
<td>Total Landing Gear Yawing Moment, about the Z-body axis</td>
<td>ANR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$L_{tot}$</td>
<td>TTL</td>
<td>totalBodyMoment_ftlbf [3]&lt;br&gt;totalBodyMoment_Nm [3]</td>
<td>Vector of total moments in the body coordinate system. Referenced to the moment reference center. Includes all moments exerted upon the aircraft. Comprised of the three components as defined below.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_{tot}$</td>
<td>TTM</td>
<td>totalBodyMoment_ftlbf_Pitch&lt;br&gt;totalBodyMoment_Nm_Pitch</td>
<td>Total Pitching Moment, about the Y-body axis</td>
<td>ANU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N_{tot}$</td>
<td>TTN</td>
<td>totalBodyMoment_ftlbf_Yaw&lt;br&gt;totalBodyMoment_Nm_Yaw</td>
<td>Total Yawing Moment, about the Z-body axis</td>
<td>ANR</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Annex A. Standard Variable Names

**Table A.9 — Simulation control parameters**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Full Variable Name</th>
<th>Description</th>
<th>Positive Sign Convention</th>
<th>Initial Value</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>simDuration_s</td>
<td></td>
<td>Time Since Start Of Operate Mode</td>
<td>NSC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>deltaTime_s [number of different integration step sizes]</td>
<td>vector of integration step sizes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>simDate</td>
<td></td>
<td>Date simulated: Date at the start of the simulation is used. (Not the date the simulation run was made)</td>
<td>Type yyyy-mm-dd</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>simTime</td>
<td></td>
<td>Simulated time of day based on 24 hours Zulu.</td>
<td>Type hh:mm:ss:ss</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>productionDate</td>
<td></td>
<td>Date the simulation run was made (Not the simulated date [simDate])</td>
<td>Type yyyy-mm-dd</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Annex A Page 52 of 54
ANNI/AIAA

Annex A. Standard Variable Names

References

GRAM 07.

For symbols:
ISO 1151-1:1988
ISO 1151-2:1985
ISO 1151-4:1994(E)
ISO 1151-5:1987

Not all symbols presented above are defined by ISO. This document attempts to make them consistent where similar.

For Time and Dates:
ISO 8601:2000
ANSI/AI AA
Annex A. Standard Variable Names

DAVE-ML Website (Informative)

The “official” DAVE-ML site is http://daveml.org. This link contains all DAVE-ML documentation and links and information on DAVE-ML tools and applications. Additional information is available at http://www.aiaa.org/
Appendix C. XML Document Type Definition file for S-119 markup: DAVEfunc.dtd

<?xml version="1.0" encoding="UTF-8"?>

-----------------------------------------------
Dynamic Aerospace Vehicle Exchange DTD
Function Data Representation

Version: 2.0 Release Candidate 3

This DTD module is identified by these PUBLIC and SYSTEM identifiers:

PUBLIC "-/AAIAA/DTD for Flight Dynamic Models - Functions 2.0//EN"
SYSTEM "http://daveml.org/DTDs/2p0RC3/DAVEfunc.dtd"

Developed by:
American Institute of Aeronautics and Astronautics (AIAA)
Modeling & Simulation Technical Committee
Simulation Modeling Standards Subcommittee

Contact information:
E. Bruce Jackson <mailto:bruce.jackson@nasa.gov>
Bruce L. Hildreth <mailto:bhildreth@jfti.com>
Persistent DAVE-ML contact <mailto:info@daveml.org>
<http://daveml.org>

Purpose:
Proposed standard for exchanging dynamic models of aerospace vehicles, including aero, engine, gear, inertia, and control models.

This preliminary version defines static models typically associated with aerodynamic subsystem models, but can be used to describe any non-linear multi-dimensional function.

Status:
In development. Direct comments to above contacts.

-----------------------------------------------

<!--

Acknowledgments:
The editors would like to acknowledge the contributions, encouragement and helpful suggestions from Dennis Linse (originally SAIC, now Vuelo Software Analysis), Jon Berndt (Jacobs Sverdrup), Brent York (Indra), Bill Cleveland (NASA Ames), Geoff Brian (Australia's DSTO), J. Dana McMinn (NASA Langley), Peter Grant (UTIAS), Giovanni A. Cignoni (University of Pisa), Daniel M. Newman (formerly Ball Aerospace, now Quantitative Aeronautics), Hilary Keating (FortBurn Pty. Ltd.), Riley Rainey (SDS International), Jeremy Furtek (Delphi Research) and Randy Brumbaugh (Indigo Innovations).

-----------------------------------------------

Page 1

NESC Request No.: 09-00598
<!-- Include the MathML2 DTD for any math markup -->

<!-- Entity %mathml2 PUBLIC "-//W3C//DTD MathML 2.0//EN"
"http://www.w3.org/Math/DTD/mathml2/mathml2.dtd"> %mathml2;

<!-- Level 0 Elements -->

Root element is DAVEfunc, composed of a file header element followed by one or more variable definitions and zero or more breakpoint definitions, gridded or ungridded table definitions, and function elements.

<!-- Level 1 Elements -->

The header element requires at least one author and a creation date; optional content includes version indicator, description, references, and modification records.

<!-- fileHeader -->

NESC Request No.: 09-00598
variableDef elements provide wiring information (i.e., 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. The variable definition can include statistical information regarding the uncertainty of the values which it might take on, when measured after any calculation is performed. Information about the reason for inclusion or change to this element can be included in an optional provenance sub-element.

A breakpointDef lists gridded table breakpoints. Since these are separate from function data they may be reused.
bpVals is a set of breakpoints (i.e., 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 2D coefficient value table.

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 may be specified separately from the actual function declaration; if so, it requires a gID identifier attribute so that it may be used by multiple functions.

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 may be specified separately from the actual function declaration; if so, it requires an internal uID identifier attribute so that it may be used by multiple functions.
Each function has optional description, optional provenance, and either a simple input/output table values or references to more complete (possible multiple) input, output, and function data elements.

This top-level element is the place-holder for verification data of various forms for the encoded model. It will include static check cases, trim shots, and dynamic check case information. The provenance sub-element is now deprecated and has been moved to individual staticShots; it is allowed here for backwards compatibility.

The author includes alternate means of identifying author using XNS or normal e-mail/address. The address sub-element is to be replaced with the more complete contactInfo sub-element.
creationDate is simply a string with a date in it. We follow ISO 8601 and use dates like "2004-01-02" to refer to January 2, 2004.

fileCreationDate is simply a string with a date in it. We follow ISO 8601 and use dates like "2004-01-02" to refer to January 2, 2004. Its use is now deprecated in favor of the simpler creationDate.

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

The description element is a textual description of an entity. The full UNICODE character set is supported by XML but may not be available in all processing applications.

The presence of the isOutput element indicates that this variable should be forced to be an output, even if it is used internally as an input elsewhere. Otherwise, the processing program may assume a signal defined with a calculation and used subsequently in the model is only an internal signal.
The presence of an `isState` element indicates that this variable is one of possibly multiple state variables in a dynamic model; this tells the processing entity that this is the output of an integrator (for continuous models) or a discretely updated state (for discrete models).

<!ELEMENT isState EMPTY>

The presence of an `isStateDeriv` element indicates that this variable is one of possibly several state derivative variables in a dynamic model; this tells the processing entity that this is the output of an integrator (for continuous models only).

<!ELEMENT isStateDeriv EMPTY>

The presence of an `isInput` element indicates that this variable is an input signal to the model.

<!ELEMENT isInput EMPTY>

The presence of an `isControl` element indicates that this signal is a simulation control parameter used to vary the operation of the model, e.g. the time step size. Such parameters should be ignored when performing linear model extraction (for example) and should not significantly modify the dynamic behavior of the model.

<!ELEMENT isControl EMPTY>

The presence of an `isDisturbance` element indicates that this signal is an external disturbance input to the model and can be ignored when performing linear model extraction (for example). Such parameters should not significantly modify the nominal dynamic behavior of the model.

<!ELEMENT isDisturbance EMPTY>
The presence of an isStdAI AA element indicates that this variable is one of the standard AIAA variable names which should be recognizable exterior to this module (e.g. AngleOfAttack_deg). This flag should assist importing tools in determining when an input or output should match a facility-provided signal name without requiring further information.

<!ELEMENT isStdAI AA EMPTY>

<!-- The calculation element is MathML 2 content markup describing how the signal is calculated. The calculation may include both constants and variables; other variables are included by using their varID string in a MathML content identifier (ci) element. -->

<!ELEMENT calculation (math)>

<!-- This element gives identifying (citation) information to an external, possibly on-line, reference document, including a user-specified author, title, classification, accession number, date and URL. -->

<!ELEMENT reference (description?)>
<!ATTLIST reference
  xmlns:xlink CDATA #FIXED 'http://www.w3.org/1999/xlink'
  xlink:type (simple) #FIXED 'simple'
  refID ID #REQUIRED
  author CDATA #REQUIRED
  title CDATA #REQUIRED
  classification CDATA #IMPLIED
  accession CDATA #IMPLIED
  date CDATA #REQUIRED
  xlink:href CDATA #IMPLIED
>

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

<!ELEMENT modificationRecord (author+,
  description?,
  extraDocRef*)>

Page 8
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.

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

When the provenance of a set of several data is identical, the first provenance should be given a provID and referenced by later provenanceRef elements.

An independentVarPts element is a simple white space- or comma-separated 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 (default is 'neither,' meaning the value of the table is held

---

NESC Request No.: 09-00598
constant at the nearest defined value). An optional interpolate attribute indicates how to perform the interpolation within the table (supporting discrete, linear, cubic or quadratic splines). There are three different discrete options: 'discrete' means nearest-neighbor, with an exact mid-point value being rounded in the positive direction; 'ceiling' means the function takes on the value associated with the next (numerically) higher independent breakpoint as soon as the original value is exceeded; and 'floor' means the function holds the value associated with each breakpoint until the next (numerically) higher breakpoint value is reached by the independent argument. The default interpolation attribute value is 'linear.'

This element is used for simple functions that do not share breakpoint or table values with other functions.

<!ELEMENT independentVarPts (#PCDATA)>
<!ATTLIST independentVarPts
    varID IDREF #REQUIRED
    name CDATA #IMPLIED
    units CDATA #IMPLIED
    sign CDATA #IMPLIED
    extrapolate (neither | min | max | both) #IMPLIED
    interpolate (discrete | floor | ceiling | linear | quadraticSpline | cubicSpline) #IMPLIED>

A dependentVarPts element is a simple comma- or white space-delimited list 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 do not share breakpoint or table values with other functions.

<!ELEMENT dependentVarPts (#PCDATA)>
<!ATTLIST dependentVarPts
    varID IDREF #REQUIRED
    name CDATA #IMPLIED
    units CDATA #IMPLIED
    sign CDATA #IMPLIED>

An independentVarRef more fully describes the input mapping of the function by pointing to a separate breakpoint definition element. An optional extrapolate attribute describes how to extrapolate the output value when the input value exceeds specified values (default is 'neither,' meaning the value of the table is held constant at the nearest defined value). An optional interpolate attribute indicates how to perform the interpolation within the
table (supporting discrete, linear, cubic or quadratic splines). There are three different discrete options: 'discrete' means nearest-neighbor, with an exact mid-point value being rounded in the positive direction; 'floor' means the function takes on the value associated with the next (numerically) higher independent breakpoint as soon as original value is exceeded; and 'ceiling' means the function holds the value associated with each breakpoint until the next (numerically) higher breakpoint value is reached by the independent argument. The default interpolation attribute value is 'linear'.

This element allows reuse of common breakpoint values for many tables but with possible differences in interpolation or extrapolation for each use.

```xml
<!ELEMENT independentVarRef EMPTY>  
<!ATTLIST independentVarRef
  varID IDREF #REQUIRED
  min CDATA #IMPLIED
  max CDATA #IMPLIED
  extrapolate (neither | min | max | both) #IMPLIED
  interpolate (discrete | floor | ceiling | linear | quadraticSpline | cubicSpline) #IMPLIED
```

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

```xml
<!ELEMENT dependentVarRef EMPTY>  
<!ATTLIST dependentVarRef
  varID IDREF #REQUIRED
```

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).

```xml
<!ELEMENT functionDefn (griddedTableRef | griddedTableDef | griddedTable |
                      ungriddedTableRef | ungriddedTableDef | ungriddedTable)>  
<!ATTLIST functionDefn
  name CDATA #IMPLIED
```

```xml
<!-- +--------------------------------------------------------------------- -->
<!-- Level 3 Elements                                                   -->
<!-- +--------------------------------------------------------------------- -->
<!ELEMENT address (#PCDATA)>                                          
```

Page 11
Used to provide contact information about an author. Use contactInfoType to differentiate what information is being conveyed and contactLocation to denote location of the address.

<!ELEMENT contactInfo (#PCDATA)>
<!ATTLIST contactInfo
contactInfoType (address | phone | fax | email | iname | web) #IMPLIED
contactLocation (professional | personal | mobile) #IMPLIED>

functionCreationDate is simply a string with a date in it. We follow ISO 8601 and use dates like "2004-01-02" to refer to January 2, 2004. Its use is now deprecated in favor of the simpler creationDate.

<!ELEMENT functionCreationDate EMPTY>
<!ATTLIST functionCreationDate
date CDATA #REQUIRED>

<!ELEMENT documentRef EMPTY>
<!ATTLIST documentRef
docID IDREF #IMPLIED
refID IDREF #REQUIRED>

<!ELEMENT modificationRef EMPTY>
<!ATTLIST modificationRef
modID IDREF #REQUIRED>

<!ELEMENT griddedTableRef EMPTY>
<!ATTLIST griddedTableRef
gtID IDREF #REQUIRED>

<!ELEMENT griddedTable (breakpointRefs?,
confidenceBound?,
dataTable )>
<!ATTLIST griddedTable
name CDATA #IMPLIED>

<!ELEMENT ungriddedTableRef EMPTY>
<!ATTLIST ungriddedTableRef
utID IDREF #REQUIRED>

<!ELEMENT ungriddedTable (confidenceBound?,
dataPoint+)>
<!ATTLIST ungriddedTable
name CDATA #IMPLIED>
Flight Simulation Model Exchange

<!ELEMENT staticShot (description?,
(provenance | provenanceRef)?,
checkInputs,
internalValues?,
checkOutputs )>

<!ATTLIST staticShot name CDATA #REQUIRED
refID IDREF #IMPLIED>

<!-- Level 4 Elements -->

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

<!ELEMENT breakpointRefs (bpRef+)>-

The confidenceBound element is used to declare the confidence interval associated with the data table. This is a place-holder and will be removed in a future version of DAVE-ML.

<!ELEMENT confidenceBound EMPTY>
<!ATTLIST confidenceBound value CDATA #REQUIRED>

The uncertainty element is used in function and parameter definitions to describe statistical variance in the possible value of that function or parameter value. Only Gaussian (normal) or uniform distributions of continuous random variable distribution functions are supported.

<!ELEMENT uncertainty (normalPDF | uniformPDF)>
<!ATTLIST uncertainty
effect (additive | multiplicative | percentage | absolute) #REQUIRED>

The dataTable element is used by gridded tables where the independent 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.

The table data point values are specified as comma- or space-separated values in conventional floating-point notation (0.93638e-06) in a single long sequence as if the table had been unwarped with the last-specified dimension changing most rapidly. Line breaks are to be ignored. Comments may be embedded in the table to promote [human] readability, with appropriate escaping characters.

A dataTable element can also be used in an uncertainty element to provide duplicate uncertainty bound values.

<!ELEMENT dataTable (#PCDATA)>

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 an ungridded table, the interpreting application is required to handle what may be unsorted data.

<!ELEMENT dataPoint (#PCDATA)>
<!ATTLIST dataPoint
  modID  IDREF  #IMPLIED>

Specifies the contents of the input vector for the given check case.

<!ELEMENT checkInputs (signal+)>

Page 14
Flight Simulation Model Exchange

<!--- -------

Provides a set of all internal variable values to assist in debugging
recalcitrant
implementations of DAVE-ML import tools.

-->

<!ELEMENT internalValues (signal+)>}

<!--- -------

specifies the contents of the output vector for the given check
case.

-->

<!ELEMENT checkOutputs (signal+)>}

<!--- Level 5 Elements
<!--- -------

The bref element provides references to a previously-defined
breakpoint set so breakpoints can be defined separately from,
and reused by, several data tables.

-->

<!ELEMENT bref EMPTY>
<!ATTLIST bref
bID IDREF #REQUIRED
>

<!--- -------

In a normally distributed random variable, a symmetrical
distribution of given standard deviation is assumed about the
nominal value (which is given elsewhere in the parent element).

The correlates with sub-element references other functions or
variables that have a linear correlation to the current
parameter or function. The correlation sub-element specifies the
correlation coefficient and references the other function or
variable whose random value helps determine the value of this
parameter.

-->

<!ELEMENT normalPDF ( 
 bounds, 
correlateswith*,
correlation* 
)>
<!ATTLIST normalPDF
numSigmans CDATA #REQUIRED
>
In a uniformly distributed random variable, the value of the parameter has equal likelihood of assuming any value within the (possibly asymmetric, implied by specifying two) bounds, which must bracket the nominal value (which is given elsewhere in the parent element).

<!ELEMENT uniformPDF (bounds+)> 

This element contains some description of the statistical limits to the values the citing parameter element might take on. This can be in the form of a scalar value, a private dataTable, or a variableRef. In the more common instance, this element will either be a scalar constant value or a simple table whose dimensions must match the parent nominal function table and whose independent variables are identical to the nominal table. It is also possible that this limit be determined by an independent variable, either previously defined or defined in-line with this element. It does not make sense to have a dataTable cited if this bounds element is associated with anything other than an identically shaped function table.

<!ELEMENT bounds (#PCDATA | dataTable | variableDef | variableRef)*)>

When present, this element indicates the parent function or variable is correlated with the referenced other function or variable in a linear sense. This alerts the application that the random number used to calculate this function's or variable's immediate value will be used to calculate another function's or variable's value.

<!ELEMENT correlatesWith EMPTY>
<!ATTLIST correlatesWith
 varID IDREF #REQUIRED>

When present, this element indicates the parent function or variable is correlated with the referenced other function or variable in a linear sense and gives the correlation coefficient for determining this function's random value based upon the correlating function(s)'s random value.

<!ELEMENT correlation EMPTY>
<!ATTLIST correlation
 varID IDREF #REQUIRED>
corrCoef  CDATA  #REQUIRED

This element is used to document the name, ID, value, tolerance, and units of measure for check-cases. When used with checkInputs or checkOutputs, the signalName sub-element must be present (since check cases are viewed from "outside" the model); when used in an internalValues element, the varID sub-element should be used to identify the signal by its model-unique internal reference. When used in a checkOutputs vector, the tol element must be present. Tolerance is specified as a maximum absolute difference between the expected and actual value.

The signalID sub-element is now deprecated in favor of the more consistent varID.

<!ELEMENT signal ( signalName, signalUnits ) |
| (varID | signalID ) |
| signalValue, 
| tol? ) >

<!ELEMENT signalName (#PCDATA)>  

<!ELEMENT signalID (#PCDATA)>  

Used to specify the input or output varID. Replaces earlier signalID element.
<!ELEMENT varID (#PCDATA)>

<!--

Used inside a checkCase element to specify the units-of-measure for an input or output variable, for verification of proper implementation of a model.

-->  

<!ELEMENT signalUnits (#PCDATA)>

<!--

Used to give the current value of an internal signal or input/output variable, for verification of proper implementation of a model.

-->  

<!ELEMENT signalValue (#PCDATA)>

<!--

This element specifies the allowable tolerance of error in an output value such that the model can be considered verified. It is assumed all uncertainty is removed in performing the model calculations. Tolerance is specified as a maximum absolute difference between the expected and actual value.

-->  

<!ELEMENT tol (#PCDATA)>

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

Version 2.0 (Release Candidate 3)
2010-05-07

AIAA Modeling and Simulation Technical Committee [https://info.aiaa.org/tac/ASG/MSTC]

E. Bruce Jackson, NASA Langley Research Center <bruce.jackson@nasa.gov>

Abstract

The Dynamic Aerospace Vehicle Exchange Markup Language (DAVE-ML) is a text-based file format intended for encoding the principal elements of a flight simulation model for an aerospace vehicle. It is based on two other open standards: the Extensible Markup Language (XML) version 1.1 and the Mathematical Markup Language (MathML) version 2.0, both products of the World Wide Web Consortium. DAVE-ML defines additional grammar (markup elements) to provide a domain-specific language capable of aerospace flight dynamics modeling, verification, and documentation.

This markup language represents the encoding format for BSR/AIAA S-119 Flight Dynamic Model Exchange Standard [AIAA10].

This is a draft version of the reference manual for DAVE-ML syntax and markup. DAVE-ML syntax is specified by the DAVEfunc.dtd Document Type Definition (DTD) file; the version number above refers to the version of the DAVEfunc.dtd.

DAVE-ML is an open standard, being developed by an informal team of American Institute of Aeronautics and Astronautics (AIAA) members. Contact the author above for more information or comments regarding refinement of DAVE-ML.
Table of Contents

1. Changes ................................................................. 4
  1.1. RCS changes ....................................................... 4
    1.1.1. RCS DTD changes ........................................... 4
    1.1.2. RCS non-DTD changes ..................................... 5
  1.2. RC2 changes ....................................................... 6
  1.3. RC1 changes ....................................................... 7
  1.4. 1.96.3 changes ................................................... 8
  1.5. 1.96.2 changes ................................................... 8
  1.6. 1.86.1 changes ................................................... 8
  1.7. 1.76.1 changes ................................................... 9
  1.8. 1.66.1 changes ................................................... 9
  1.9. 1.56.3 changes ................................................... 9
  1.10. 1.56.2 changes .................................................. 10

2. Introduction .......................................................... 11

3. Purpose .................................................................. 12

4. Background ................................................................ 13
  4.1. Existing standards ............................................... 13
  4.2. DAVE-ML proposal ............................................... 13
  4.3. DAVE-ML applications .......................................... 13

5. Supporting technologies .............................................. 15

6. Major elements .......................................................... 16
  6.1. DAVEfunc ............................................................ 16
    6.2. DAVEfunc overview .............................................. 18
      6.2.1. fileHeader overview ...................................... 21
      6.2.2. variableDef overview ..................................... 23
      6.2.3. breakpointDef overview ................................... 30
      6.2.4. griddedTableDef overview ................................. 31
      6.2.5. ungriddedTableDef overview .............................. 34
      6.2.6. function overview ......................................... 38
      6.2.7. checkData overview ....................................... 43
  6.3. Interpolation ......................................................... 46
  6.4. Statistics ................................................................ 49

6.5. Conventions ................................................................ 55
  6.5.1. Point order ....................................................... 55
  6.5.2. Moment Reference ............................................. 56
  6.5.3. Subsystem decomposition ...................................... 56
  6.5.4. Date format ....................................................... 56
  6.5.5. Sign conventions ............................................... 56
  6.5.6. Units .................................................................. 56
  6.5.7. XML Identifiers .................................................. 57
  6.5.8. Namespace ........................................................ 57

6.6. Future .................................................................. 58

7. More information .......................................................... 59

8. Element references and descriptions .................................. 60
  8.1. Alphabetical list of elements .................................... 60
  8.2. Element descriptions .............................................. 62
1. Changes to this document

This section contains a list of changes during the development of the DAVE-ML DTD.

1.1. Changes since version 2.0RC2

This section outlines changes to the DTD and this reference manual since version 2, release candidate 2, was released in October 2008.

The changes are divided into structural changes to the DTD and non-structural changes to the DTD and reference manual.

1.1.1. DTD structural changes since version 2.0RC2

- Corrected the DTD to require at least one staticShot child element of the checkData element.

- Changed the DTD logic for inclusion of provenance information for elements griddedTableDef, ungriddedTableDef, function, checkData, and staticShot from

  
  (provenance? | provenanceRef ?) to

  
  (provenance | provenanceRef )

which is the more correct way to display an optional selection from two choices. However, use of provenance or provenanceRef is now deprecated for checkData elements as described in the non-structural changes below; this change was made for consistency with the other elements that have provenance information.

- Added optional, mutually exclusive flag elements isInput, isControl, and isDisturbance to conform with the latest draft of [AIAA10]. These flags are intended to help clarify the role of all defined variables and to assist in analyses such as linear model extraction.

- Changed the Uniform Resource Identifier (URI) for the atom2 function definition to the daveml.org domain.

- Changed SYSTEM ID to reflect new daveml.org domain; changed PUBLIC ID from NASA to AIAA.

- In the DTD, the default DAVE-ML namespace definition was added to the top-level DAVEml:funce element to observe a best-practice for XML DTDs, at the suggestion of Dan Newman.

- In the DTD and reference manual, changed the bspVals elements to allow white space separation of values in addition to comma separation (which has always been the case for the dataTable element). Ditto for dependentVarPts and independentVarPts. This may cause a problem for some existing parsers.
DAVE-ML 2

No other changes have been made to the DAVE-ML grammar since 2.0RC2, but this version of the DTD and reference manual include clean-up of a number of inconsistencies found by Dennis Linse and Trey Arthur between the reference manual and element descriptions, as noted below.

1.1.2. Non-structural DTD and reference manual changes since version 2.0RC2

- Corrected second ungridded table definition example and CLRUD0 function example; reformatted page references from [xx] to (p. xx) in the PDF reference manual, thanks to suggestions by Dennis Linse.

- Amplified and elaborated on possible values for interpolate and extrapolate attributes for independent variables of function definitions.

- Added an index to the reference manual.

- Removed the sign convention list as this is now in the overlying AIAA draft standard, S-119 [AIAA10].

- Added clarification that, while uncertainty can be applied in multiple places in the model (including input, calculations, functions, and outputs), it is probably not a good practice to do so.

- Changed the non-dimensional signal units-of-measure indication from blank or ND to 'n/d', in accordance with the latest draft standard [AIAA10]; removed plus sign from front of sign convention since it doesn't always apply and is redundant to the definition.

- Added references to earlier standards papers by B. Hildreth [Hildreth94], [Hildreth98].

- Cleaned up formatting in the element descriptions found in Section 8.2 (p. 62) corrected the grammar in the BNF descriptions of the major elements within this text.

- Added two sentences to the description of the calculation element, explaining how to tie variables in the model to the MathML expression via the MathML content identifier (ci) element. Thanks to Missy Hill and Curtis Zimmerman for pointing out the lack of explanation.

- Changed email address for B. Hildreth; added persistent email at daveml.org; changed default Uniform Resource Locator (URL) to daveml.org

- Changed the description of bounds element in the DTD to remove out-of-date reference to [un]griddedTable[DefRef] since this is no longer supported. In the reference manual, removed a misleading second bounds element for the uniformPDF element case and added notes to imply the presence of PCDATA (either one or two, depending on the case). Thanks to Dan Newman for prompting this review and change.

- Did more DTD clean-up to remove redundant 'optional' specifiers on elements and attributes; made the 'internal identifier' descriptions consistent throughout the DTD and reference manual; tweaked the style of the reference pages in the reference manual; fixed poor grammar in the DTD; replaced parentheses with brackets in the BNF syntax in the reference manual for comments to avoid confusion; reformatted the reference pages in the manual for readability; and added references to W3C on-line documentation where necessary. Thanks to Dennis Linse for encouraging the correction of these long-standing inconsistencies and distractions.
DAVE-ML 2

- Put editorial comments in BNF syntaxes in braces ({}), to avoid confusion with parentheses, which are used to indicate a choice. Thanks to Dennis Linse for the catch.
- In the DTD, a sentence was added to the description of the deprecated fileCreationDate element and a similar description was added to the deprecated functionCreationDate element mentioning that both of them have been deprecated in version 2.0. The clarification was made after near-simultaneous suggestions from both Trey Arthur and Dan Newman.
- Added a paragraph about the alias attribute of the variableDef element; it's a valid attribute but was missing an explanation. Thanks to Trey Arthur for catching this long-standing omission.
- Changed the definition of atan2 so that the input arguments are not limited to ±1 and are not referred to as 'sine' and 'cosine' to better match ANSI C. Thanks to Dan Newman for catching this inconsistency.
- Corrected typographical errors in fileHeader, variableDef, griddedTableDef, ungriddedTableDef, and function element schematic overview syntaxes in sections 6.2.1-6.2.6. Thanks to Dennis Linse for catching these errors.
- Added a sentence about the tol sub-element being an absolute difference to remove ambiguity over its interpretation.
- In the reference manual syntax layout, moved the provenance and provenanceRef sub-elements out of checkData and into staticShot sub-elements since checkData, a singleton place-holder, doesn't warrant a description but each staticShot does. provenance and provenanceRef are still in the DTD for checkData but only for backwards compatibility; their use in a checkData element is deprecated and may be removed in a future version. Thanks to Dennis Linse for prompting this needed change.
- Added a note to documentRef docID attribute that it is deprecated. Thanks to Dennis Linse for catching this omission.
- Removed the redundant "optional" in the description of reference element's xlink:href attribute. Added the constant value for attributes xlink:xlink and xlink:type. Expanded the somewhat terse description of this element. Thanks to Dennis Linse for this correction and other helpful suggestions.

1.2. Changes since version 2.0RC1

- Tweaked examples and syntaxes to match the 2.0 RC 2 DTD. Cleaned up a couple of figures and incorporated several new reference citations. Added interpolation paragraph.
- Deprecated signalID used for internal signals in check-case data in favor of the more consistent varID (which meant the introduction of a formal varID element) and made the specification of signalUnits sub-element mandatory for input and output signals for consistency. Thanks to Dan Newman for helping solidify the thinking about this issue.
- Changed examples in this text to use updated AIAA variable names to match a revised (but unpublished) draft standard of September 2008. Changed many 'examples' to 'excerpts' to
emphasize the missing portions of a valid DAVE-ML file. Corrected units in check-case examples to match the draft AIAA standard:

- Removed the symmetric attribute of the uniformPDF sub-element; this attribute was redundant as symmetric distribution is implied with a single bounds sub-element, and asymmetric is implied by two bounds sub-elements. Kudos to Dan Newman and Dennis Linse for catching the inconsistent examples and for suggesting this convention.

- Corrected the DTD by reversing the definition of the floor and ceiling values of the interpolate attribute of the independentVarPts element; also corrected the correlated uncertainty example. Thanks to Dan Newman for catching both of these problems in 2.0 RC1.

- Corrected the DTD so that only one checkData element is allowed (but it can have multiple different staticShot test conditions). Thanks to Dan and Dennis for reporting this inconsistency between the reference manual and the DTD.

- Added a description sub-element to the staticShot element in response to a suggestion by Dennis Linse; and added a typical description to example listings.

- Added a section about namespaces and removed the hard link in the DTD that incorrectly set the namespace for the calculation element.

- Added new multi-purpose creationDate element to replace the single-purpose fileCreationDate and functionCreationDate elements, at the suggestion of Dennis Linse of SAIC. fileCreationDate and functionCreationDate are now deprecated.

- Corrected descriptions of ungriddedTableDef and griddedTableDef to reflect the possible use of an internal function element; previously the descriptions implied that these elements were only specified external to functions and thus the utID and gridID attributes, respectively, were required. Thanks to Dennis Linse for the correction.

- Depicted provenanceRef as an option to the provenance sub-element for griddedTableDef, ungriddedTableDef, and function elements in the narrative part of this manual (it was described correctly in the reference section). Also added both provenance and provenanceRef as optional sub-elements of the variableDef and checkData elements. Thanks to Dennis Linse for the correction.

- Added [Deprecated] to the description of griddedTable, ungriddedTable, and confidenceBound elements for consistency; these were previously deprecated but not marked clearly in each element’s ‘purpose’ section. Thanks to Dennis Linse for the suggestion.

- Updated the acknowledgment paragraph of the DTD; significantly reformatted the PDF version of this document, and added section numbers to all versions.

1.3. Changes since version 1.9b3

- Added ceiling and floor enumeration selections to interpolate attribute of independentVarPts and independentVarRef elements at the suggestions of Geoff Brian, Giovanni Cignoni, Randy Brumbaugh, and Dan Newman.
DAVE-ML 2

- Added five uncertainty examples.
- Cleaned up all FIXME and BUG notes.
- Corrected and expanded the labels on the DAVE-ML excerpt figure.

1.4. Changes since version 1.9b2

- Corrected link to the DAVE-ML exchange test [Jackson04] paper.
- Added discrete enumeration selection to interpolate attribute of independentVarPts and independentVarRef elements at the suggestion of Geoff Bryan, Australian Defence Science and Technology Organisation (DSTO).
- Added a section and a variableDef example on extending the MathML-2 function set with \texttt{atan2}.
- Removed all \texttt{xms} attributes from examples.
- Emphasized that it is a good practice to provide \texttt{variableDefs} in sorted sequence.

1.5. Changes since version 1.8b1

- Added a \texttt{quadraticSpline enumerated value} to the interpolate attributes of the independentVarPts and independentVarRef elements (in response to a request from Geoff Bryan of DSTO) and fixed a typographical error in \texttt{cubicSpline attribute string}. Added reference to Wikipedia article on spline interpolation [http://en.wikipedia.org/wiki/Spline interpolation].
- Added a \texttt{classification attribute} to the \texttt{reference} element and added a date attribute to the \texttt{modificationRecord} element, at the suggestion of Geoff Bryan of DSTO.
- Added 2D and 3D ungridded table examples and figures and corrected a typographical error in the ungridded table definition syntax thanks to Dr. Peter Grant of U. Toronto’s UTIAS and Geoff Bryan of DSTO.
- Reintroduced \texttt{<ENTITY> to include MathML-2 DTD (complete) in the body of this DTD. This entity definition quietly went away in version 1.6 due to a misunderstanding of the proper way to include external DTDs. It was reintroduced to assist with validating parsers.}
- Added a description sub-element to the \texttt{provenance} element, so the provenance entry can contain more information about change justification documents and made \texttt{provenance or provenanceRef} acceptable sub-elements to \texttt{variableDef} and \texttt{checkData} elements at the request of Geoff Bryan of DSTO.

1.6. Changes since version 1.7b1

- Renamed \texttt{docID attribute} to \texttt{refID} in the \texttt{modificationRecord} so the attribute name is consistent; the \texttt{docID attribute} is deprecated but remains in place for compatibility with older documents.

NESC Request No.: 09-00598
DAVE-ML 2

- Added correlatesWith and correlation sub-elements of uncertainty element to allow for multiple-dimensional linear correlation of uncertainty of selected functions and variables.

- Added a new element, contactInfo, to replace the single address element. This format supports multiple ways to indicate how to contact the author of a document or reference. address is deprecated but is retained for backwards compatibility. This element also replaces the email and xmlns attributes of author.

- Fixed a typographical error in ungriddedTableRef element: incorrect gTID attribute corrected to uTID.

- Allowed multiple author elements wherever only one had been allowed before.

- Added a new tag, isStdAIAA, to indicate that a variableDef refers to one of the standard AIAA variables.

- Removed [un]griddedTable[Ref|Def] sub-elements of the confidenceBound element since this leads to circular logic.

- Changed SYSTEM.ID to reflect new dave.ml.nasa.gov domain availability.

- Removed true email from examples to protect privacy of individual contributors.

- Added a new attribute, interpolate, to the independentVarPts element to indicate whether the table interpolation should be linear or cubic spline in the given dimension [modified to include quadratic in version 1.9].

- Added a new tag, isState, to indicate that a variableDef refers to a state variable in the model.

- Added a new tag, isStateDeriv, to indicate that a variableDef refers to a state derivative variable in the model.

1.7. Changes since version 1.6b1

Added checkData and associated elements. Added description sub-element to reference element.

1.8. Changes since version 1.5b3


1.9. Changes since version 1.5b2

- Added Bill Cleveland (NASA Ames Research Center's SimLab) and Brent York (NAVIAIR's Manned Flight Simulator) to the acknowledgments section, to thank them for their pioneering initial trials of DAVE-ML.
DAVE-ML 2

• Added provenanceRef element and changed all parents (specifically function, griddedTableDef and ungriddedTableDef elements) of provenance elements to enable them to use a provenanceRef reference instead of eliminate duplicate provenance elements.

• Realization dawned that there was little difference between griddedTables and griddedTableDefs but that the latter was more flexible (ditto ungriddedTables and ungriddedTableDefs). By making the gid and uid attributes 'implied' instead of 'required,' we can use the Def versions in both referenced-table and embedded-table functions. Thus the original griddedTable and ungriddedTable elements have been marked as 'deprecated.' They are still supported in this DTD for backwards compatibility but should be avoided in future use. The easiest way to modify older DAVE-ML models would be to rename all griddedTables as griddedTableDefs.

1.10. Changes since version 1.5b

• Fixed typographical errors pointed out by Bill Cleveland.

• 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]) standard does not appear to be catching on as rapidly as hoped, so a static e-mail link will have to do for now, at least until the replacement XRI technology is more widely adopted.

• 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.
2. Introduction

This document describes the format for DAVE-ML model definition files. DAVE-ML is a proposed standard format for the interchange of aerospace vehicle flight dynamics models. The intent of DAVE-ML is to significantly expedite the process of re-hosting a simulation model from one facility to another and function as an improved method to promulgate changes to a particular model between 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 [http://www.w3.org/XML/].

The exchange of aerospace vehicle flight dynamics models may derive many benefits from the application of XML in general, and DAVE-ML in particular:

- Provides a human-readable text description of the model
- Provides an unambiguous machine-readable model description, suitable for conversion into programming language or direct import into object-oriented data structures at run-time
- Allows use of the same source file for computer-aided design and real-time piloted simulation
- Based on open, non-proprietary, standards that are language- and facility-independent
- Allows inclusion of statistical properties, such as confidence bounds and uncertainty ranges, suitable for Monte Carlo or other statistical analysis of the model
- Complies with emerging AIAA simulation data standards
- Represents a self-contained, complete, archivable data package, including references to reports, wind-tunnel tests, author contact information, and data provenance
- Is self-documenting and easily convertible to on-line and hard-copy documentation

A more complete discussion on the benefits and design of DAVE-ML can be found at the DAVE-ML web site: http://daveml.org [http://daveml.org]
3. Purpose

DAVE-ML is intended to encode (for exchange and long-term archive) an entire flight vehicle dynamic simulation data package, as is traditionally done in initial delivery and updates to engineering development, flight training, and accident investigation simulations. It is intended to provide 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 vendor-specific and is often incomplete or inconsistent. Many times, the preparing facility makes incorrect assumptions about how the receiving facility's simulation environment is structured. As a result, the re-hosting of the dynamic flight model by the receiving facility can take weeks or longer as the receiving facility staff learns the contents and arrangement of the data package, the model structure, the various data formats, and variable names/units/sign conventions. The staff then spends additional time running check-cases (if any were included in the transmittal) and tracking down inevitable differences in results.

There are obvious benefits to automating most of this tedious, manual process. Often, when a pair of facilities has already exchanged one model, the transmission of another model is much faster since the receiving facility will probably have devised some scripts and processes to convert the data (both model and check-case data).

The purpose of DAVE-ML is to develop a common exchange format for these flight dynamic models. The advantage gained is to enable any simulation facility or laboratory, after having written a DAVE-ML import and/or export script, to automatically receive and/or transmit such packages (and updates to those packages) rapidly with other DAVE-ML-compliant facilities.

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

DAVE-ML forms the encoding portion of the Flight Simulation Model Exchange Standard, BSR/ AIAA S-119-2010 (currently in draft form). More information is available at the S-119 web site [AIAA10].
4. Background

The idea of a universally understood flight dynamics data package has been discussed for at least two decades within the AIAA technical committees [Hildreth94], [Hildreth98]. There have been proposals in the past to standardize 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 Flight Research Center, developed a hybrid 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 in the early 1990s. Some other simulation standards have subsequently been proposed by the AIAA and are under active consideration at this writing [AIAA10].

4.1. 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 and Simulation Technical Committee has prepared a draft standard for the exchange of simulation modeling data. This includes a methodology for accomplishing the gradual standardization of simulation model components, a mechanism for standardizing variable names within math models, and proposed Hierarchical Data Format 5 (HDF5) as the data format. This document is included as an Annex to the standard [AIAA01], [AIAA03], [AIAA10].

4.2. 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 pursuing such a goal.

The 2002 proposal included several key aspects from the draft standard, including allowing use of a standard variable-name convention and data table schema and including traceability for each data point back to a referenced document or change order.

In a subsequent paper, Jackson, Hildreth, York and Cleveland [Jackson04] reported on the results of a demonstration using DAVE-ML to exchange two aerodynamic models between simulation facilities, showing the feasibility of the idea.

4.3. Recent applications

Several successful applications of DAVE-ML have been reported. These include the adoption of DAVE-ML by the Australian DSTO for threat models [Brian05] and the U.S. Navy for their Next Generation Threat System [Hildreth08]. Import tools to allow the direct use of DAVE-ML models (without recompilation) in real-time piloted simulations have been reported by NASA Langley Research Center (LaRC) [Hill07] and at NASA Ames Research Center (ARC). Some interest has been generated within NASA’s Orion Project as well [Acevedo07]. Other applications include TSONT, a trajectory optimization tool ([Durak06]) and aircraft engine simulations ([Liu04]).
DAVE-ML 2

DAVE-ML format for models has also been supported by the GeneSim [http://genesim.sourceforge.net] Project, which is providing open-source utility programs that realize a DAVE-ML model in object-oriented source code such as C++, Java and C#.
5. Supporting technologies

DAVE-ML 2 relies on MathML, version 2.0, to define mathematical relationships. MathML-2 is an XML grammar 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-2 home page, found at http://www.w3.org/Math/.

MathML-2 provides a mostly complete set of mathematical functions, including trigonometric, exponential and switching functions. One function that is available in most programming languages and computer-aided design tools but is missing from MathML-2 is the two-argument arc tangent function which provides a continuous angle calculation by comparing the sine and cosine components of a 2D coordinate set. DAVE-ML provides a means of extending MathML-2 for a predefined set of functions (currently only the $atan2$ function is defined). Thus, a DAVE-ML-compliant processing tool should recognize this extension (which is accomplished by using the MathML-2 $\texttt{csymbol}$ element). See Section 6.2.2 (p. 23) for a discussion and Example 6 (p. 29).
6. 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. Static check-cases can also be provided for verification of proper implementation.

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

6.1. The DAVEfunc major element

The DAVEfunc element contains both data tables and equations for a particular static model. A DAVEfunc element is broken into six components: a file header, variable definitions, breakpoint definitions, table definitions, function definitions and optional check-cases. This decomposition reflects common practice in engineering development flight-simulation models in which the aerodynamic database is usually captured in multi-dimensional, linearly interpolated function tables. The inputs 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 MathML2 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 fileHeader element is included to give background and reference data for the represented model.

Variables, or more properly signals, are used to route inputs and calculations through the subsystem model into outputs. Each variable is 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 from the model, and/or the result of intermediate calculations. Variables must be defined for each input and output of any function element as well as any input or output of the subsystem represented. MathML2 [http://www.w3.org/Math] content markup can be used to define constant, intermediate, or output variables as mathematical combination of constant values, function table outputs, and other variables, but any presentation markup is not required and should be ignored by the processing application (except as required to generate documentation). Variables also represent the current value of a function (the dependentVariableDef in a function definition) so the output of functions can be used as inputs to other variables or functions.

Breakpoint definitions, captured in breakpointDef elements, consist of a list of monotonically increasing floating-point values separated by commas or white space. These sets are 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 aerodynamics model, and typically
DAVE-ML 2

represent a 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 "ungridged," meaning each data point has a specified coordinate location in n-space. The same table can be reused in several functions, such as a left- and right-sileron moment contribution.

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.

Check-case data (described by a single checkData element) can be included to provide information to automatically verify the proper implementation of the model by the recipient. Multiple check-cases can (and should) be specified using multiple staticShot test-case definitions, as well as optional internal signal values within the model to assist in debugging an instantiation of the model by the recipient.

Figure 1 (p. 18) contains excerpts from an example model, showing five of the six major parts of a DAVE-ML file.
Figure 1. Excerpts from an example DAVE-ML file

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 reuse of the breakpoint sets by other functions but keeps the table data private.

6.2. Schematic overview of DAVEfunc

Shown below are schematic overviews of the various elements currently available in DAVEfunc. Each element is described in detail in Section 8 (p. 60) later in this document. The following key is used to describe the elements and associated attributes.
DAVE-ML 2

Key:
*elementname* : mandatory_attributes, [optional_attributes]
  mandatory_single_sub-element
  optional_single_sub-element?
  (choice_1_sub-element | choice_2_sub-element)
  zero_or_more_sub-elements
  one_or_more_sub-elements
  (character data) impliesUniqueId text information

The DAVEfunc element has six possible sub-elements:

- **DAVEfunc**
  - fileHeader
  - variableDef
  - breakpointDef
  - griddedTableDef
  - function
  - checkData?

**DAVEfunc sub-elements:**

- **fileHeader**
  This mandatory element contains information about the origin and development of this model.

- **variableDef**
  Each DAVEfunc model must contain at least one signal path (such as a constant output value). Each input, output or internal signal used by the model must be specified in a separate variableDef.

  A signal can have only a single origin (an input block, a calculation, or a function output) but can be used (referenced) more than once as an input to one or more functions, signal calculations, and/or as a model output.

  In DAVE-ML 2.0, all signals are real and scalar.

  The variableDefs should appear in calculation order; that is, a variableDef should not appear before the definitions of variables upon which it is dependent. This is good practice since doing so avoids a circular reference.

  If a variable depends upon the output (dependentVar) of a function it can be assumed that dependence has been met, since function definitions appear later in the DAVEfunc element.

- **breakpointDef**
  A DAVEfunc model can contain zero, one, or more breakpoint set definitions. These definitions can be shared among several gridded function tables. Breakpoint definitions can appear in any order.

- **griddedTableDef**
  A DAVEfunc model can contain zero, one, or more gridded nonlinear function table definitions. Each table
must be used by multiple function definition if desired for efficiency. Alternatively, some or all functions in a model can specify their tables internally with an embedded griddedTableDef element.

A gridded function table contains dependent values, or data points, corresponding to the value of a function at the intersection of one or more breakpoint sets (one for each dimension of the table). The independent values (coordinates or breakpoint sets) are not stored within the gridded table definition but are referenced by the parent function. This allows a function table to be supported by more than one set of breakpoint values (such as left- and right-aileron deflections).

ungriddedTableDef

A DAVEfunc model can contain zero, one, or more ungridded nonlinear function table definitions. Unlike a rectangularly gridded table, an ungridded table specifies data points as individual sets of independent and dependent values. Each table must be used by at least one but can be used by multiple function definitions if necessary for efficiency. Alternatively, functions can retain their tables internally with a ungriddedTable element without sharing the table values with other functions.

Ungridded table values are specified as a single (unsorted) list of independent variable (input) values and associated dependent variable (output) values. While the list is not sorted, the order of the independent variable inputs is important and must match the order given in the parent function. Thus, functions that share an ungridded table definition must have the same ordering of independent variables.

The method of interpolating the ungridded data is not specified.

function

A function ties together breakpoint sets (for gridded-table nonlinear functions), function values (either internally or by reference to table definitions), and the input- and output-variable signal definitions, as shown in Figure 1 (p. 18). Functions also include provenance, or background history, of the function data such as wind tunnel test or other source information.

checkData

This optional element contains information allowing the model to be automatically verified after implementation by the receiving party.
6.2.1. The file header element

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

```xml
fileHeader : [name]
    author : [name, org, [email]
        contactInfo* : [contactInfoType, contactLocation]
    creationDate : [in ISO 8601 YYYY-MM-DD format]
    fileVersion* : [file version identifier]
    description? : [textual description of model]
    reference* : refID, author, title, date, [classification, accession, href]
    modificationRecord* : modID, date, [refID]
    author* : name, org, [email]
        contactInfo* : [contactInfoType, contactLocation]
    description?: [textual description of modification]
    extraDocRef* : refID
    provenance* :
        author* : name, org, [email]
            contactInfo* : [contactInfoType, contactLocation]
        creationDate : [in ISO 8601 YYYY-MM-DD format]
        documentRef* : [docID], refID
        modificationRef* : modID
        description? : [textual description of the background of the model]
```

**fileHeader sub-elements:**
- **author**: Name, organization, optional email address and other contact information for each author.
- **creationDate**: Creation date of this file. See Section 6.5.4 (p. 56) for the recommended format for encoding dates.
- **fileVersion**: A string that indicates the version of the document. No convention is specified for the format, but a good practice would include an automated revision number from a version control system.
- **description**: An optional but recommended text description: what does this DAVE-ML file represent?
- **reference**: An optional list of one or more references with a document-unique ID (must begin with alpha characters), author, title, date, and optional accession and URL of the
DAVE-ML 2

reference. This sub-element can include a description of
the reference.

modificationRecord

An optional list of one or more modifications with
optional reference IDs, 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 sub-element externalRefs
may be used in place of the modificationRecord's
refID attribute.

provenance

An optional list of one or more provenance elements
allows the author to describe the source and history
of the data within this model. Since the model may
be constructed from several sources, more than one
provenance may be provided, one for each source of
data. Use of a provID attribute in the fileHeader is
unnecessary since this provenance applies to the entire
model unless otherwise specified.

Example 1. An excerpt with an example of a fileHeader element

```xml
<fileHeader>
  <author name="Bruce Jackson" org="NASA Langley Research Center">
    <contactInfo contactInfoType="address" contactLocation="professional">
      MS 206 NASA, Hampton, VA 23681
    </contactInfo>
    <contactInfo contactInfoType="email" contactLocation="professional">
      Bruce.Jackson@larc.nasa.gov
    </contactInfo>
  </author>
  <creationDate date="2003-03-15"/>
  <fileVersions>1.24</fileVersions>
  <description>
  Version 2.0 aero model for HL-20 lifting body, as described in
  NASA TM-107586. This aero model was used for HL-20 approach
  and landing studies at NASA Langley Research Center during 1999-1995
  and for follow-on studies at NASA Johnson Space Center in 1994
  and NASA Ames Research Center in 2001. This DAVE-ML version was
  created in March 2000 by Bruce Jackson to demonstrate DAVE-ML.
  </description>
  <reference refID="REF01">
    author="Jackson, E. Bruce; Cruz, Christopher I. & Haggard, W. A."
    title="Real-Time Simulation Model of the HL-20 Lifting Body"
    accession="NASA TM-107586"
    date="1999-07-01"
  </reference>
</fileHeader>
```

NESC Request No.: 09-00598
Use of comments make models more readable by humans.
2 Start of the fileHeader element.
3 Creation date of file, in ISO-8601 format. See Section 6.5.4 (p. 56)
4 In this example, the revision number is automatically inserted by a version control system.
5 All documents referenced by notation throughout the file should be described in the fileHeader as reference elements.
6 All modifications made to the contents of this file should be listed in the fileHeader as modificationRecord sub-elements for easy reference by later modificationRecord elements.

6.2.2. The variable definition element

The variableDef element is used to define each constant, parameter, or variable used within or generated by the defined subsystem model. It contains attributes including the variable name (used for documentation), an internal and unique varID identifier (used for linking inputs, functions and outputs), the units of measure of the variable, and optional axis system, sign convention, alias, and symbol declarations. Optional sub-elements include a written text description and a mathematical description, in MathML-2 content markup, of the calculations needed to derive the variable from other variables or function table outputs. Optional sub-element isOutput serves to indicate an intermediate calculation that should be brought out to the rest of the simulation. Another optional sub-element isStdR1AA, indicates the variable name is defined in the AIAA simulation standards document. Another optional sub-element, uncertainty, captures the statistical properties of a (normally constant) parameter.

Other optional sub-elements are provided to identify inputs, disturbances, and simulation control parameters, as well as the ability to identify a variable as a state or state derivative for linear model purposes.

There must be a single variableDef for each and every input, output or intermediate constant or variable within the DAVEml model.
variableDef : name, varID, units, {axisSystem, sign, alias, symbol, initialValue} description? : 
   {description character data}
{provenanceRef : provID
OR
provenance : [provID]
   {author : name, org, [email]
   contactInfo* : [contactInfoType, contactLocation]
   {text describing contact information]
   creationDate : date [in YYYY-MM-DD format]
   documentRef* : [docID, refID
   modificationRef* : modID
   description?
}
| calculation? : 
|   method [defined in MetML-2 DTD]
| isInput | isControl | isDisturbance?
| isState?
| isStateDeriv?
| isOutput?
| isModelDep?
| uncertainty? : effect
   {normalPDF : mean(symbols) | uniformPDF : bounds}

variableDef attributes:

name
A UNICODE name for the variable (may be the same string as the varID).

varID
An internal identifier that is unique within the file.

units
The units-of-measure for the signal, using the AIAA standard units convention [AIAA10].

axisSystem
An optional indicator of the axis system (body, inertial, etc.) in which the signal is measured. See [AIAA10] or Section 6.5 (p. 55) below for recommended practice for nomenclature.

sign
An optional indicator of which direction is considered positive (+RWD, +UP, etc.). See [AIAA10] or the section on Section 6.5 (p. 55) below for recommended practice for abbreviations.

alias
An optional, facility-specific variable name, perhaps used in the equations of motion or control system model, that does not conform to the AIAA standard for variable names. Use of this attribute is discouraged for portability reasons.

symbol
A UNICODE Greek symbol for the signal [to be superseded with more formal MathML or TeX element in a later release].
DAVE-ML 2

initialValue An optional initial value for the parameter. This is normally specified for constant parameters only.

variableDef sub-elements:
description An optional text description of the variable.

provenance The optional provenance element allows the author to describe the source and history of the data within this variableDef. Alternatively, a provenanceRef reference can be made to a previously defined provenance.

calculation An optional container for the MathML-2 content 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].

A MathML-2 calculation can include both constants (using the content numeric cn element) and references to other variables internal to the parent DAVEmath description. The variables (which can include the output, or dependent variable of a function table) are identified using its varID attribute string in the appropriate MathML content identifier (ci) element of the expression.

Examples of MathML expressions appear later in this reference.

isInput This optional element, if present, signifies that this variable is an input to the model, such as a pilot input deflection or Mach number. Useful for linear model extraction tools. It must not be the result of a calculation or be cited as the dependent variable of a function.

isControl This optional element, if present, signifies that this variable is a simulation control parameter, such as a trim flag or simulation control parameter. Simulation control parameters should have no influence on the dynamic behavior of the model and should be ignored by a linear model extraction tool.

isDisturbance This optional element, if present, signifies that this variable represents an external disturbance input to the model; this is useful for linear model extraction tools to partition this input separately from the other model inputs.

isOutput This optional element, if present, signifies that 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 the $DAVEfunction$.

$\text{isStdAIAA}$

This optional element, if present, signifies that this variable is one of the standard AIAA simulation variable names that are defined as Annex A to [AIAA10]. Such identification should make it easier for the importing process to connect this variable (probably an input or output of the model) to the appropriate variable in/from the user's simulation framework.

$\text{isState}$

This optional element, if present, signifies that this variable serves as a state of the model.

$\text{isStateDeriv}$

This optional element, if present, signifies that this variable serves as a state derivative of the model.

$\text{uncertainty}$

This optional element, if present, describes the uncertainty of this parameter. See the section on Statistics below for more information about this element.

Example 2. An example of two $\text{variableDef}$ elements defining input signals

In this example, two input variables are defined: $\text{XRACH}$ and $\text{DFDCL}$. These two variables are inputs to a table lookup function shown in Example 11 (p. 41) below.

```xml
<variableDef name="mach" varID="XRACH" units="nd" symbol="N"> 
  <description>
    Mach number (dimensionless)
  </description>
  <isInput/> 
</variableDef>

<variableDef name="LBDCL" varID="DFDCL" units="deg" sign="TEC" symbol="\$4x384:DFDCL"/> 
  <description>
    Lower left body flap deflection, deg, positive trailing-edge-down (no deflections are always zero or positive).
  </description>
  <isInput/> 
</variableDef>
```
DAVE-ML 2

1. The name attribute is intended for humans to read, perhaps as the signal name in a wiring diagram. Note that "machNumber" is one of the standard AIAA simulation variable names.

2. The varId attribute is intended for the processing application to read. This is an internal identifier that must be unique within this model.

3. The optional isInput attribute indicates this variable should be treated as an input to the model for model hierarchy and linear model extraction (for example).

4. The optional instAIAA sub-element indicates this signal is one of the predefined standard variables that most simulation facilities define in their equations of motion code. The name attribute should correspond to the standard AIAA parameter name from Annex A of [AIAA10] or subsequent standards document.

5. The mandatory units attribute describes the units of measure of the variable. See Section 6.5.6 (p. 56) below for a recommended list of units-of-measure abbreviations.

6. The optional sign attribute describes the sign convention that applies to this variable. In this case, the lower-left body-flap is positive with trailing-edge-down deflection. See Section 6.5.5 (p. 56) below for a recommended list of sign abbreviations.

7. The optional symbol attribute allows a UNICODE character string that might be used for this variable in a symbols listing.

Example 3. A simple local variable definition example

This DAVE-ML excerpt defines CLBFLLO which is the dependent variable (output) from a table lookup function (shown later in Example 11 (p. 41). It is subsequently used in the calculation of the lower-left body flap lift coefficient (shown in Example 4 (p. 27)).

```xml
<--- ------------------- -->
<--- Local variables --->
<--- ------------------- -->
<--- PRELIMINARY BUILDUP EQUATIONS -->
<--- LOWER LEFT BODY FLAP CONTRIBUTIONS -->
<--- table output signal -->
<variableDef name="CLBFLLO" varId="CLBFLLO" units="nd"
<description>
Output of CLBFLLO function; lift force contribution of lower left body flap deflection due to alpha (constant term).
</description>
</variableDef>
```

Since this signal is not flagged as an input, control, disturbance or output, this variable is an intermediate signal local to this model.

Example 4. A more complete variableDef example with a calculation element

In this example, the local variable CLBFLLO is defined as a calculated quantity, based on several other input or local variables including the CLBFLLO function output variable defined in the previous example (p. 27) Note the description element is used to describe the equation in Fortran-ish human-readable text. The calculation element describes the same equation in MathML-2 content markup syntax; this portion should be used by parsing applications to create either source code, documentation, or run-time calculation structures.

```xml
<--- lower left body flap lift buildup -->
<variableDef name="CLBFLLO" varId="CLBFLLO" units="nd">
```
DAVE-ML 2

```
<description>
  Lift contribution of lower left body flap deflection
  Clbflill - Clbflill_o + alpha*(Clbflill_1 + alpha*(Clbflill_2
  + alpha*Clbflill_3))
</description>

<calculation>
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <apply>
    <plus/>
    <ci>Clbflill</ci>
    <apply>
      <plus/>
      <ci>Clbflill_o</ci>
      <apply>
        <times/>
        <ci>alpha</ci>
        <apply>
          <plus/>
          <ci>Clbflill_1</ci>
          <apply>
            <times/>
            <ci>alpha</ci>
            <ci>Clbflill_2</ci>
            <apply>
              <plus/>
              <ci>alpha</ci>
              <ci>Clbflill_3</ci>
            </apply>
          </apply>
        </apply>
      </apply>
    </apply>
  </apply>
</math>
</calculation>
</variablesdef>
```

1. This Fortran-ish equation, located in the description element, is provided in this example for the benefit of human readers; it should not parsed by the processing application.
2. A calculation element always embeds a MathML-2 math element; note the definition of the MathML-2 namespace.
3. Each apply tag pair surrounds a math operation (in this example, a plus operator) and the arguments to that operation (in this case, a variable Clbflill defined elsewhere is added to the results of the nested apply operation).
4. The content identifier (ci) MathML-2 element gives the varID of the previously defined variables used in this equation; this variable represents the output of the Clbflill function found in Example 11 (p. 41) that is captured in the Clbflill0 variable defined in Example 3 (p. 27). The other ci elements are not defined in this manual but are defined in the full manual.
5. Inner-most apply multiplies variables alpha and Clbflill2.
6. The comments here are useful for humans to understand how the equation is being built up; the processing application ignores all comments.

Example 5. Another example of an output variable based on a calculation element

This excerpt is an example of how an output variable (CL) might be defined from previously calculated local variables (in this case, CL0, Clbflill, etc.).
DAVE-ML 2

```xml
<variable name="CL" varId="CL" units="" sign="up" symbol="CL">  
  <description>
  Coefficient of lift
  CL = CL0 + CLBFL + CLBFL2 + CLBFL4 + CLWFL + CLWFL2 + CLWFL4 + CLG
  </description>
  <calculation>
  <math>
  <apply>
    <plus/>
    <ci>CL0</ci>
    <ci>CLBFL</ci>
    <ci>CLBFL2</ci>
    <ci>CLBFL4</ci>
    <ci>CLWFL</ci>
    <ci>CLWFL2</ci>
    <ci>CLWFL4</ci>
    <ci>CLG</ci>
  </apply>
  </math>
  </calculation>
  <isOutput/>
</variable>
```

1. Here apply simply sums the value of these variables, referenced by their varIDs.
2. This `ci` element refers to the lower left body flag lift contribution calculated in the previous example (p. 27).
3. The `isOutput` element signifies to the processing application that this variable should be made visible to models external to this DAVEML file.

**Example 6. An intermediate variable with a calculation element that uses a DAVE-ML function extension to the default MathML-2 function set**

In this excerpt, we demonstrate a means to encode a math function, `atan2`, that is not available in the default MathML-2 function set. The `atan2` function is used often in C, C++, Java, and other modeling languages and has been added to the DAVE-ML standard by use of the MathML-2 `csymbol` element, specifically provided to allow extension of MathML-2 for cases such as this.

```xml
<variable name="Wind vector roll angle" varId="PHI" units="rad">  
  <description>
  This encodes the equation PHI = atan2( tan(BETA), sin(ALPHA) ) where atan2 is the two-argument arc tangent function from the ANSI C standard math library.
  </description>
  <calculation>
  <math>
  <apply>
    <csymbol definitionURL="http://davenl.org/function_spaces.html#atan2" encoding="text">atan2</csymbol>
  </apply>
  </math>
</variable>
```

---

NESC Request No.: 09-00598
DAVE-ML 2

This excerpt shows how to calculate wind roll angle, phi, from angle-of-attack and angle-of-sideslip; it comes from the Apollo aerodynamics data book [NAA64].

The caseStudy element is provided by MathML-2 as a means to extend the function set of MathML-2. An extension for \textit{atan2} is the only function defined at present but others may be added to the set in the future. Note the specific URI that uniquely identifies this function; it is also the URL (web address) of the documentation of the interpretation of the \textit{atan2} function.

\begin{itemize}
\item BETA is the variID of a previously defined variable.
\item ALPHA is the variID of a previously defined variable.
\end{itemize}

\subsection*{6.2.3. The breakpoint set definition element}

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

\begin{verbatim}
breakpointDef : bpID, [name, units]
  description?:
  bpVals : [character data of comma- or white space-separated breakpoints]
\end{verbatim}

\texttt{breakpointDef} attributes:

- \texttt{bpID}:
  An internal reference that is unique within the file.

- \texttt{name}:
  A UNICODE name for the set (may be the same string as \texttt{bpID}).

- \texttt{units}:
  The units-of-measure for the breakpoint values. See Section 6.5.6 (p. 56) below.

\texttt{breakpointDef} sub-elements:

- \texttt{description}:
  An optional text description of the breakpoint set.

- \texttt{bpVals}:
  A comma- or white space-separated, monotonically increasing list of floating-point values.
Example 7: Two `breakpointDef` examples in a DAVE-ML model excerpt

As an example, two `breakpoint` sets are defined which are used in the function element given below (Example 11 (p. 41)). Breakpoint sets `XMACH1_PTS` and `DBFL_PTS` contain values for Mach and lower body flap deflection, respectively, which are used to look up function values in several gridded function tables. One example is given below in Example 8 (p. 53).

```
<breakpointDef name="mach" bpID="XMACH1_PTS" units="nd">  
  <description>
    Mach number breakpoints for all aero data tables
  </description>
  <bpVals>
    0.1, 0.6, 0.9, 0.95, 1.1, 1.2, 1.6, 2.0, 2.5, 3.0, 3.5, 4.0  
  </bpVals>
</breakpointDef>

<breakpointDef name="Lower body flap" bpID="DBFL_PTS" units="deg">  
  <description>Lower body flap deflections breakpoints for tables</description>
  <bpVals>6, 15, 30, 45, 60</bpVals>
</breakpointDef>
```

1. This `breakpointDef` element describes a Mach breakpoint set uniquely identified as `XMACH1_PTS` with no associated units of measure.
2. The breakpoint values are given as a comma- or white space-separated list and must be in monotonically increasing numerical order.
3. This breakpoint set defines the breakpoints for lower body flap deflection.

6.2.4. The gridded table definition element

The `griddedTableDef` element defines a multi-dimensional table of values corresponding with the value of an arbitrary function at each intersection of a set of specified independent input values. 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 data table definition are a comma- or white space-separated set of floating-point values. This list of values represents a multi-dimensional array whose size is inferred from the length of each breakpoint vector. For example, a 2D 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 (8 x 10) comma- or white space-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. See Section 6.5.1 (p. 55) below.

An optional `uncertainty` element may be provided that represents the statistical variation in the values presented. See Section 6.4 (p. 49) for more information about this element.
DAVE-ML 2

{description of table in character data}
RATHER
provenanceRef? : provid

or
provenance? : [provid]
    author? : name, org, [email]
    contactInfo* : [contactInfoType, contactLocation]
        [text describing contact information]
    creationDate? : [YYYY-MM-DD format]
    documentRef* : [docID], refID
    modificationRef? : modID
    description?
    breakpointRefs : bpRef*, bpID
    uncertainty? : [effect]
        (normalPDF | normalSIGMA | uniformPDF)
    dataTable
        [character data of comma- or white space-separated table values]

gridTableDef attributes:
gtID
    An internal reference that is unique within the file.

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

units
    The units-of-measure of the table's output signal. See Section 6.3.6 (p. 56) below.

gridTableDef sub-elements:
description
    The optional description element allows the author to describe the data contained within this gridTable.

provenance
    The optional provenance element allows the author to describe the source and history of the data within this gridTable. Alternatively, a provenanceRef reference can be made to a previously defined provenance.

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; see the text above.

uncertainty
    This optional element, if present, describes the uncertainty of this parameter. See Section 6.4 (p. 49) for more information about this element.

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- or white space-separated
Flight Simulation Model Exchange

Example 8. An excerpt showing an example of a griddedTableDef element

This nonlinear function table is used by a subsequent function in Example 11 (p. 41) to specify an output value based on two input values: body flap deflection and Mach number. This table is defined outside of a function element because this particular function table is used by two functions: one for the left-lower body flap and one for the lower-right body flap; thus, their actual independent (input) variable values might be different.

```xml
<gridTableDef name="CL6FLO" gid="CL6FLO_table">
  <description>
    Lower body flap contribution to lift coefficient; polynomial constant term
  </description>
  <provenance>
    <author name="Bruce Jackson" org="NASA Langley Research Center"
      email="b.jackson@larc.nasa.gov"/>
    <creationDate date="2003-01-31"/>
    <documents docID="KEF01"/>
  </provenance>
  <breakpointDefs>
    <bpRef bpID="NSFL_FTS"/>
    <bpRef bpID="XONCHL_FTS"/>
  </breakpointDefs>
  <dataTables>
    <!-- last breakpoint (MACH) changes most rapidly -->
    <grid> 0.0 -->
      0.00000E+00 , 0.00000E+00 , 0.00000E+00 , 0.00000E+00 ,
      0.00000E+00 , 0.00000E-00 , 0.00000E-00 , 0.00000E-00 ,
      0.00000E+00 , 0.00000E+00 , 0.00000E+00 , 0.00000E+00 ,
      0.99999E-01 , 0.99999E-01 , 0.99999E-01 , 0.99999E-01 ,
      0.99999E-01 , 0.99999E-01 , 0.99999E-01 , 0.99999E-01 ,
      1.00000E+00 , 1.00000E+00 , 1.00000E+00 , 1.00000E+00 ,
      1.00000E+00 , 1.00000E+00 , 1.00000E+00 , 1.00000E+00 ,
      1.00000E+00 , 1.00000E+00 , 1.00000E+00 , 1.00000E+00 ,
      1.00000E+00 , 1.00000E+00 , 1.00000E+00 , 1.00000E+00 ,
    </grid>
    <!-- DBFL -->
    <grid> 15.0 -->
      -0.86429E-02 , -0.10453E-01 , -0.11183E-01 , -0.12212E-01 , -0.12502E-01 ,
      -0.62590E-02 , -0.53679E-02 , -0.70575E-02 , -0.11300E-01 , -0.62999E-02 ,
      -0.51920E-02 , -0.38813E-02 , -0.37363E-02 ,
    </grid>
    <!-- DBFL -->
    <grid> 30.0 -->
      0.23251E+01 , 0.24055E+01 , 0.28808E+01 , 0.31206E+01 , 0.34609E+01 ,
      0.31232E+01 , 0.28996E+01 , 0.19509E+01 , 0.15498E+01 , 0.12735E+01 ,
      0.10644E-01 , 0.94493E-02 , 0.83719E-02 ,
    </grid>
    <!-- DBFL -->
    <grid> 45.0 -->
      . . . .
    </grid>
  </dataTables>
</gridTableDef>
```

1. Comments are good practice for human readers
2. name is used for documentation purposes; gid is intended for automatic wiring (autocode) tools.
3. Descriptions make for good practice whenever possible. Here we explain the contents of the function represented by the data points.
8.2.5. The ungridded table definition element

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.

A 'non-orthogonal' data set, as opposed to a gridded or 'orthogonal' data set, means that the independent values are not laid out in an orthogonal grid. This form must be used if the dependent coordinates in any table dimension cannot be expressed by a single monotonically increasing vector.

See the excerpts below for two instances of ungridded data.

An optional uncertainty element may be provided that represents the statistical variation in the values presented. See the section on Statistics below for more information about this element.

ungriddedTableDef : {utID, name, units}
    description? : [
        (description character data)
    ]
    EITHER
    provenance? : provID
    OR
    provenance? : [provID]
        author : [name, org, [email]]
        contactInfo* : [contactInfoType, contactLocation]
        [text describing contact information]
        creationDate* : date [in YYYY-MM-DD format]
        documentRef* : [docID, resID]
        modificationDate* : modID
        description?
        uncertainty? : effect
            (normalPDF | eusSignature) | (uniformPDF | bounds)
        dataPoint* : [
            [coordinate/value sets as character data]
    ]

ungriddedTableDef attributes:

utID
    An internal reference that is unique within the file

name
    An optional UNICODE name for the table (may be the same string as utID).

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

ungriddedTableDef sub-elements:
DAVE-ML 2

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. Alternatively, a provenanceRef reference can be made to a previously defined provenance.

uncertainty

This optional element, if present, describes the uncertainty of this parameter. See the section on Statistics below for more information about this element.

dataPoint

One or more sets of coordinate and output numeric values of the function at various locations within its 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 parent function.

Example 9. An excerpt showing an ungriddedTableDef element, encoding the data depicted in Figure 2 (p. 36).

This 2D function table is an example provided by Dr. Peter Grant of the University of Toronto. Such a table definition would be used in a subsequent function to describe how an output variable would be defined based on two independent input variables. The function table does not indicate which input and output variables are represented; this information is supplied by the function element later so that a single function table can be reused by multiple functions.

```xml
<ungriddedTableDef name="CLHASIC as function of flap angle and angle-of-attack" uid="CLHASICFlag_Table" units="nd">
  <description>
    CLHASIC as a function of flap angle and angle-of-attack. Note the alpha in this table is with respect to the wing design plane (in degrees). Flap is in degrees as well.
  </description>
  <provenance>
    <author name="Peter Grant" org="UTIAS"/>
    <creationDate date="2006-11-01"/>
    <documentRef refId="P008"/>
  </provenance>
  <dataPoint>
    <flap>1.0</flap> <alpha>-6.00</alpha> <angleOfAttack>-6.44</angleOfAttack> <!-- flap, alpha, alpha -->
  </dataPoint>
  <dataPoint>
    <flap>1.0</flap> <alpha>10.00</alpha> <angleOfAttack>-6.95</angleOfAttack> <!-- flap, alpha, alpha -->
  </dataPoint>
  <dataPoint>
    <flap>3.0</flap> <alpha>12.00</alpha> <angleOfAttack>1.13</angleOfAttack> <!-- flap, alpha, alpha -->
  </dataPoint>
  <dataPoint>
    <flap>3.0</flap> <alpha>14.00</alpha> <angleOfAttack>1.24</angleOfAttack> <!-- flap, alpha, alpha -->
  </dataPoint>
  <dataPoint>
    <flap>3.0</flap> <alpha>15.00</alpha> <angleOfAttack>1.31</angleOfAttack> <!-- flap, alpha, alpha -->
  </dataPoint>
  <dataPoint>
    <flap>5.0</flap> <alpha>17.00</alpha> <angleOfAttack>1.41</angleOfAttack> <!-- flap, alpha, alpha -->
  </dataPoint>
  <dataPoint>
    <flap>5.0</flap> <alpha>-5.00</alpha> <angleOfAttack>-0.58</angleOfAttack> <!-- flap, alpha, alpha -->
  </dataPoint>
  <dataPoint>
    <flap>5.0</flap> <alpha>6.00</alpha> <angleOfAttack>-0.63</angleOfAttack> <!-- flap, alpha, alpha -->
  </dataPoint>
  <dataPoint>
    <flap>5.0</flap> <alpha>5.00</alpha> <angleOfAttack>-0.60</angleOfAttack> <!-- flap, alpha, alpha -->
  </dataPoint>
</ungriddedTableDef>
```
DAVE-ML 2

Example courtesy of Dr. Peter Grant, U. Toronto
Comments are a good idea for human readers
For a 2D table such as this one, data points give two columns of independent breakpoint values and a third column of function values at those breakpoints.
The mod1D attribute implies this point was edited during modification 'A' of this model, as described in the file header information.

![Figure 2. The 2D lift function given in Example 9 (p. 35)](image)

Example 10. An excerpt from a sample aerodynamics model giving an example of a 3D ungriddedTableDef element, encoding the data shown in Figure 3 (p. 38).

In this example, the dependent coordinates all vary in each dimension.
DAVE-ML 2

Figure 3. The 3D function given in the previous example

6.2.6. The function definition element

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

```
function : name
description?:
   [text description of the function]
EITHER
   provenanceRef? : provID
OR
   provenance?: [provID]
   author*: [name, org, [email]]
   contactInfo*: [contactInfoType, contactLocation]
   [text describing contact information]
   creationDate* : date [in YYYY-MM-DD format]
   documentRef*: [docID, version], relID
   modificationRef*: modID
   description?

EITHER

   IndependentVar*: varID, [name, units, sign, extrapolate, interpolate]
   [input values as character data]
   dependentVar*: varID, [name, units, sign]
   [output values as character data]

OR

```
DAVE-ML 2

```
independentVarRef : varID, (min, max, extrapolate, interpolate)
dependentVarRef : varID
functionDefn : [name]
    CHOICE OF
      | gridTableRef : gTableID
      OR
      gridTableRef : [name, gTableID, units]
        description?    [text description of table]
        [provenance]    [provenanceRef]?  [as described earlier]
        breakPointRef?
        [uncertainty]    [affect]
          (normalPDF : numSamples) | (uniformPDF : bounds+)
        dataTable
          [gridded data table as character data]
      OR
      ungridTableDef : uTableID
      OR
      ungriddedTableDef : [name, uTableID, units]
        description?
        [text description of table]
        [provenance]    [provenanceRef]?  [as described earlier]
        uncertainty?    [affect]
          (normalPDF : numSamples) | (uniformPDF : bounds+)
        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. Alternatively, a provenanceRef reference can be made to a previously defined provenance.
- **independentVarPts**: If the author chooses, she can express a linearly interpolated functions by specifying the independent (breakpoint) value sets as one or more independentVarPts which are comma- or space-separated, monotonically increasing floating-point coordinate values corresponding to the dependentVarPts. In the case of multiple dimensions, more than one independentVarPts must be specified, one for each dimension. The mandatory varID attribute is used to connect each independentVarPts set with an input variable.
An optional interpolate attribute specifies the preference for using linear, quadratic, or cubic relaxed splines for calculating dependent values when the independent arguments are in between specified values. When not specified, the expectation would be to use a linear spline interpolation between points. The performance of interpolation of various orders is left up to the processing application. See Section 6.3 (p. 46).

**dependentVarPts**

This element goes along with the previous element to specify a function table. Only one dependentVarPts may be specified. If the function is multi-dimensional, the convention is the last breakpoint dimension changes most rapidly in this comma- or white space-separated list of floating-point output values. The mandatory varID attribute is used to connect this table's output to an output variable.

**independentVarRef**

One or more of these elements refers to separately defined variableDefs. The order of specification is important and must match the order in which breakpoint sets are specified or the order of coordinates in ungridded table coordinate/value sets.

An optional interpolate attribute specifies the preference for using discrete, linear, quadratic, or cubic splines for calculating dependent values when the independent arguments are in between specified values. When not specified, the default expectation would be a linear spline interpolation between points. The performance of interpolation of various orders is left up to the implementer. See Section 6.3 (p. 46).

**dependentVarRef**

A single dependentVarRef must be specified to connect the output of this function to a particular variableDef.

**functionDefn**

This element identifies either a separately specified data table definition 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.

**ungriddedTableRef**

If not using a simple function table, the author may use this element to point to separately specified ungriddedTableDef element.
Example 11. An excerpt giving the example of a function which refers to a previously defined gridTableDef

This example ties the input variables DBFLL and XMACH into output variable CLBFLLO through a function called CLBFLLO_fn, which is represented by the linear interpolation of the grid table previously defined by the CLBFLLO_table gridTableDef (see the gridTableDef example above).

```xml
<function name="CLBFLLO">  
<description>
  lower left body flap lookup function for lift, polynomial constant term.
</description>
<independentVarRef varID="DBFLL" min="0.0" max="60.0" extrapolate="neither"/>  
<independentVarRef varID="XMACH" min="0.0" max="4.0" extrapolate="author"/>
</function>
</functionDef>
</gridTableDef grid="CLBFLLO_table"/>
</function>

1. The independent variables must be given in the order of least-rapidly changing to most-rapidly changing values in the previously defined function table. The processing application needs to pay attention to the extrapolate attribute, which specifies how to treat a variable whose value exceeds the stated limits on input. See Section 6.3 (p. 46).
2. The dependent variable (identified as CLBFLLO) is the output variable for this function. CLBFLLO must have been declared previously with a variableDef element.
3. This is a reference to the previously declared gridTableDef.

Example 12. A function with an internal table

In this example, the function CLRUD0 returns, in the variable CLRUD0, the value of function CLRUD0_fn represented by grid table CLRUD0_table. The inputs to the function are abs_rud and XMACH which are used to normalize breakpoint sets DRUD_PTS and XMACH1_PTS respectively. The input variables are limited between 0.0 to 30.0 and 0.3 to 4.0, respectively.

In this case, the use of the CLRUD0 string for both the function name attribute and as the varID for the dependent (output) variable reference does not interfere (although they are confusing); name is not in the XML namespace. The name attribute is only used for documentation (such as a label for a box representing this function).

```xml
<function name="CLRUD0">  
<description>
</description>
</function>
</functionDef>
</gridTableDef grid="CLRUD0_table"/>
</function>
```

1. The gridTableDef is used to identify the table entries used in the function definition.
2. * Grid table entries are only used once, as their table definitions are internal to the function definition.

---

NESC Request No.: 09-00598
DAVE-ML 2

Rudder contribution to lift coefficient, polynomial multiplier for constant term.
</description>
<provenance>
  <author name="Bruce Jackson" org="NASA Langley Research Center">
    <email>bjackson@nasa.gov</email>
    <creationDate date="2003-01-31"/>
  </author>
  <documentRef docID="ISEF01"/>
</provenance>
<independentVarRef varID="nose_rud" min="-6.0" max="10.0" extrapolate="neither"/>
<independentVarRef varID="CMACH" min="0.3" max="4.0" extrapolate="neither"/>
<dependentVarRef varID="CLMUDO"/>

<functionDef name="CLMUDO_def"/>
<griddedTableDef name="CLMUDO_table">
  <breakpointDefs/>
  <breakpointDefs/>
</griddedTableDef>
<table>
  <!-- last breakpoint changes most rapidly -->
</table>
</functionDef>
</function>

1. This comment helps humans understand the reason for an embedded table.
2. The name attribute is used for documentation purposes, not for internal variable linkage.
3. The provenance element is required by the AIAA standard [AIAA10].
4. The varID attribute is used to link the output of this function with a previously defined variable as given in Example 3 (p. 27).
5. This example has an embedded gridded table.

Example 3. A simple 1D function

At the other end of the spectrum, a simple 1D nonlinear function can be defined with no reuse, as shown below; however, multiple-dimension functions are supported by adding additional independentVarPts definitions.

<function name="CI">
  <independentVarPts varID="alpha_deg"> -4.6, 0., 4.0, 8.0, 12.0, 16.0 </independentVarPts>
  <dependentVarPts varID="CI"> 9.6, 9.2, 9.4, 9.6, 10.0, 10.2 </dependentVarPts>
</function>

NESC Request No.: 09-00598
6.2.7. The `checkData` element

The `checkData` element contains one or more input/output vector pairs (and optionally a dump of internal values) for the encoded model to assist in verification and debugging of the implementation.

```xml
<checkData>
  <staticShot name="Nominal" refID="NORTH">  
    <description>An example static check of a simple DAVE-ML model.</description>
    <checkInputs>
      <signal>
        <signalName>trackSpeed</signalName>
        <signalUnits>ft/s</signalUnits>
        <signalValue>360.000</signalValue>
      </signal>
      <signal>
        <signalName>angleOfAttack</signalName>
        <signalUnits>deg</signalUnits>
        <signalValue>-5.000</signalValue>
      </signal>
    </checkInputs>
    <checkOutputs>
      <signal>
        <signalName>alpha</signalName>
        <signalUnits>deg</signalUnits>
        <signalValue>-5.000</signalValue>
      </signal>
    </checkOutputs>
  </staticShot>
</checkData>
```

Example 14. Check-case data set excerpt

A DAVE-ML file excerpt specifying a check-case data set example for a simple model.
DAVE-ML 2

</signal>
  
  (similar input signals omitted)

</signal>

<signal>  
  <signalName>Delta elevator</signalName>
  <signalUnits>deg</signalUnits>
  <signalValue>0.000</signalValue>
</signal>

</checkInputs>

<checkOutputs>  
  <signal>  
    <signalName>CX</signalName>
    <signalUnits>deg/s</signalUnits>
    <signalValue>-0.000</signalValue>
    <tol>0.000000000000000</tol>
  </signal>

  (similar output signals omitted)

</checkOutputs>

</staticShot>

<staticShot name="Positive pitch rate">  
  <checkInputs>  
    (similar input and output signal information omitted)
  </checkOutputs>

</staticShot>

<staticShot name="Positive elevator">  
  <checkInputs>  
    (similar input and output signal information omitted)
  </checkOutputs>

</checkOutputs>

</checkData>

1. This first check-case refers to a note given in the file header; this is useful to document the source of the check-case values.
2. The checkInputs element defines the input variable values, by variable name, as well as units (so they can be verified).
3. Multiple signal elements are usually given; taken together, these scalar signals represent the check-case input "vector."
4. The checkOutputs element defines output variable values that should result from the specified input values.
5. Note the included tolerance value, indicating the absolute value tolerance within which the output values must match the check-case data values.
6. Multiple check-cases may be specified; this one differs from the previous check-case due to an increase in the pitching-rate input.

Example 15. A second checkData example with internal values

This example shows another check-case; this one includes intermediate values as an aide to debugging a new implementation.

<checkData>
  
  (provenance | provenanceRef )
  
  <staticShot name="Skewed inputs">
DAVE-ML 2

The internalValues element, if present, usually is a list of all model-defined internal variable values. This is normally not required for a model exchange, but such information is very useful to aid in debugging the implementation by the recipient.
6.3. Function interpolation/extrapolation

It is possible to specify the method of interpolation to be used for nonlinear function tables by use of the interpolate attribute of the independentVarPts and independentVarRef elements. This attribute, combined with the extrapolate flag, provides several different ways of realizing the intermediate values of the function when not at one of the specified intersections of independent values.

Possible values for the interpolate attribute are:

- discrete: Output uses value associated with nearest breakpoint
- floor: Output uses value associated with next (numerically higher) breakpoint
- ceiling: Output uses value associated with last (numerically lower) breakpoint
- linear (default): Output is linearly interpolated between breakpoints
- quadraticSpline: Output follows a quadratic spline fit through values associated with two nearby breakpoints
- cubicSpline: Output follows a cubic spline fit through values associated with three nearby breakpoints

Possible values for the extrapolate attribute are:

- neither (default): Output is held constant at value associated with closest end of breakpoints if the input value is outside the limits of the associated breakpoint set
- min: Output follows extrapolated values of function if the input is below the minimum breakpoint value
- max: Output follows extrapolated values of function if the input is above maximum breakpoint value
- both: Output follows extrapolated values of function if the input is outside the limits of the associated breakpoint set

Implementation of the specific interpolation algorithm is left up to the implementer. One reference is the Wikipedia entry on interpolation [wiki01].

The following implementation notes are suggested:

- An infinite set of quadratic interpolations are possible; it is suggested to use the one that minimizes either the deviation from a linear interpolation or the slope error at any edge.
- For cubic interpolation, the natural cubic spline (which has a second derivative of zero at each end) is recommended when the extrapolate attribute is none. When the extrapolate
attribute is both, a clamped cubic spline that matches the extrapolated slope of the last two data points is suggested.

- For the discrete interpolation values (discrete, ceiling, or floor), the value of the extrapolate attribute is meaningless.

For discrete interpolation,

- discrete implies the change between output values occurs midway between independent breakpoint values, as shown in the top plot of Figure 4 (p. 48).

- ceiling means the output takes on the value of the next-higher dependent variable breakpoint as soon as each independent breakpoint value is passed (assuming the input value is increasing) as shown in the middle plot of Figure 4 (p. 48).

- floor means the output retains the value of the last dependent variable breakpoint until the next independent breakpoint value is reached (assuming the input value is increasing) as shown in the bottom plot of Figure 4 (p. 48).

The default value for interpolate is linear. The default value for extrapolate is neither.

Figures 4 (p. 48) and 5 (p. 49) below give nine different examples for a 1D table whose independent values are [1, 3, 4, 6, 7.5] with dependent values of [2, 6, 5, 7, 1.5].
Figure 4. Example of the three discrete enumeration values of `interpolate` attribute of the `independentVarDts` and `independentVarRef` elements for a 1D function table.
6.4. Statistical information encoding

Statistical measures of variation of certain parameters and functions can be embedded in a DAVE-ML model in several ways. This information is captured in an uncertainty element, which can be referenced by variableDef, griddedTableDef and ungriddedTableDef elements. For maximum modeling flexibility, it is possible to add uncertainty to the independent value arguments to a function or calculation, to the output of a function itself, as well as to any output signal. Applying uncertainty at more than one location in a calculation change is probably not a good practice, however.

Details on providing the random values for uncertainties is left to the implementer.
Uncertainty in the value of a parameter or function is given in one of two ways, depending on the appropriate probability distribution function (PDF): as a Gaussian or normal distribution (bell curve) or as a uniform (evenly spread) distribution. One of these distributions is selected by including either a normalPDF or a uniformPDF element within the uncertainty element.

Linear correlation between the randomness of two or more variables or functions can be specified. Although the correlation between parameters does not have a dependency direction (i.e., the statistical uncertainty of one parameter is specified in terms of the other parameter, therefore the calculation order does not matter), correlation is customarily specified as a dependency of one random variable on the value of another random variable. correlatesWith identifies variables or functions whose uncertainty 'depends' on the current value of this variable or parameter; the correlation sub-element specifies the correlation coefficient and identifies the (previously calculated) random variable or function on which the correlation depends.

These correlation sub-elements only apply to normal (Gaussian) probability distribution functions.

Each of these distribution description elements contain additional information, as described below.

```
uncertainty : effect=["additive","multiplicative","percentage","absolute"]
  EITHER
    normalPDF : numSigmas=["1"|"2"|"3"]
      bounds [ scalar value representing the one, two or three sigma bound ];
        (correlatesWith = varID ; correlation = varCorr )
  OR
    uniformPDF
      bounds [ one or two scalar values for abs. or min/max bounds ]
```

**uncertainty attributes:**

**effect**

Indicates, by choice of four enumerated values, how the uncertainty is modeled: as an additive, multiplicative, or percentage variation from the nominal value, or a specific number (absolute).

**uncertainty sub-elements:**

**normalPDF**

If present, the uncertainty in the parameter value has a probability distribution that is Gaussian (bell-shaped). A single parameter representing the additive (+ some value), percentage (+ some %) of variation from the nominal value in terms of 1, 2, 3, or more standard deviations (sigmas) is specified. Note: multiplicative and absolute bounds do not make much sense.

**uniformPDF**

If present, the uncertainty in the parameter or function value has a uniform likelihood of taking on any value between symmetric or asymmetric boundaries, which
DAVE-ML 2

are specified in terms of additive (either $\pm x$ or $x/y$), multiplicative, percentage, or absolute variations. If absolute, the specified range of values must bracket the nominal value. For this element, the bounds subelement may contain one or two values, in which case the boundaries are symmetric or asymmetric.

Example 16. A variable with absolute uncertainty bounds

This example shows how to specify that a constant parameter can take on a specified range of values with uniform probability distribution. The nominal value of the minimum drag coefficient is specified to be 0.005, but when performing parametric variations, it is allowed to take on values between 0.001 and 0.01.

```
<DAVEfunc>
  <fileHeader>
    ...
    ...
  </fileHeader>
  <variableDef name="CD zero" varID="CDo" units="m/s" initialValue="0.005">
    <description>
      Minimum coefficient of drag with an asymmetric uniform uncertainty band
    </description>
    <isOutput/>
    <uncertainty effect="absolute">
      <uniformDIF>
        <bounds>0.001</bounds>
        <bounds>0.010</bounds>
      </uniformDIF>
    </uncertainty>
  </variableDef>
</DAVEfunc>
```

1. We declare the parameter CDo as having a nominal value of 0.005.
2. When parametric variations are applied, the value of CDo can vary uniformly between 0.001 and 0.010.

Example 17. 10% uncertainty applied to output variable with a uniform distribution

This example shows how to specify that a variable has a 10% uniformly distributed uncertainty band. In this example, the output variable comes from a nonlinear 1D function and the uncertainty is applied to the output of the table.

```
<DAVEfunc>
  <fileHeader>
    ...
    ...
  </fileHeader>
  <variableDef name="angleOfAttack" varID="Alpha_deg" units="deg">
    <description/>
  </variableDef>
  <variableDef name="Cn_u" varID="Cn_u" units="m/s">
    <description/>
  </variableDef>
</DAVEfunc>
```
DAVE-ML 2

Coefficient of pitching moment with 10 percent symmetric uniform uncertainty band</description>
<isOutput/>
<uncertaintyEffect="percentage"> 0
  <uniformPDF>
    <bounds>0.00</bounds>
  </uniformPDF>
</uncertainty>
</variableDef>
</breakpointDef>
<breakpointDef bpID="ALP">  2
</breakpointDef>
</breakpointDef>
<variableDef/>
</griddedTableDef>
</breakpointRef>
</dataTable>
</griddedTableDef>
</functionDef>
</function>
</DAVEfunc>

We declare the output variable Cm_u as having the uncertainty of ±10% uniform distribution.

This function gives the nominal values of Cm_u as a 1D function of angle-of-attack (alpha).

Example 18. Asymmetric additive uncertainty applied to output variable with uniform distribution

This example shows how to specify that a variable has an asymmetric, uniformly distributed, additive uncertainty band. In this example, the output variable comes from a nonlinear 1D function and the uncertainty is applied to the output of the table.

<DAVEfunc>
  <fileHeader>
    
  </fileHeader>
  <variableDef name="angleOfAttack" varID="Alpha_deg" units="deg">
    <isOptional/>
  </variableDef>
  <variableDef name="Cm_u" varID="Cm_u" units="mN">
    <description>
      Coefficient of pitching moment with an asymmetric uniform uncertainty band
    </description>
    <isOutput/>
    <uncertaintyEffect="additive"> 0
      <uniformPDF>
        <bounds>-0.56</bounds>
        <bounds>-0.56</bounds>
      </uniformPDF>
    </uncertainty>
  </variableDef>
</DAVEfunc>
DAVE-ML 2

```xml
<uniformPDF/>
</uncertainty>
</variableDef>
</breakpointRef>
</function name="Nominal Cm_u">
<description>
Nominal pitching moment values prior to application of uncertainty
</description>
<independentVarRef varID="Alpha_deg"/>
<dependentVarRef varID="Cm_u"/>
<functionDefn>
<griddedTableDef>
<breakpointRef>
<bpRef bpID="ALP"/>
</breakpointRef>
</dataTable>
5.2, 4.3, 3.1, 1.9, 1.0, 1.3, 1.0, -0.0, -0.1
</dataTable>
</griddedTableDef>
</functionDefn>
</function>
</DAVEfunc>

1. We declare the output variable Cm_u varies by as much as ±0.5 to +0.0 about the nominal value. This delta value is in the same units as the nominal value (i.e., it is not a multiplier or percentage but an additive delta to the nominal value which comes from the 1D Cm_u function table description).

2. This function gives the nominal values of Cm_u as a 1D function of angle-of-attack (alpha).

Example 19. A 1D point-by-point, Gaussian distribution function

In this example, a Gaussian (normal) distribution function is applied to each point in a 1D function table, with the 3-sigma value expressed as a multiplier of the nominal value.

```xml
<DAVEfunc>
<fileHeader>
...
</fileHeader>
<variableDef name="angleOfAttack" varID="Alpha_deg" units="deg">
<isInALF/>
</variableDef>
<variableDef name="Cm_u" varID="Cm_u" units="nd">
<description>
Coefficient of pitching moment with 10 percent symmetric uniform uncertainty band
</description>
<isUncertain/>
</variableDef>
<breakpointRef bpID="ALP">
<bpVals>5, 10, 15, 20, 25, 30, 35</bpVals>
</breakpointRef>
<function name="Uncertain Cm_u">
<independentVarRef varID="Alpha_deg"/>
<dependentVarRef varID="Cm_u"/>
<functionDefn>
<griddedTableDef>
<breakpointRef>
</breakpointRef>
</functionDefn>
</function>
</DAVEfunc>
```
Flight Simulation Model Exchange

DAVE-ML 2

This declares the statistical uncertainty bounds of the C_{m_u} dependent variable will be expressed as a multiplication of the nominal value.

This declares that the probability distribution is a normal distribution and the bounds represent 3-sigma (99.7%) confidence bounds.

This table lists three-sigma bounds of each point of the C_{m_u} function as a table. The table must have the same dimensions and independent variable arguments as the nominal function; it is in effect an overlay to the nominal function table, but the values represent the bounds as multiples of the nominal function value.

This table defines the nominal values of the function; these values will be used if the random variable associated with the uncertainty of this function is zero or undefined by the application.

Example 20. Two nonlinear functions with correlated uncertainty

In this example, uncertainty in pitching-moment coefficient varies in direct correlation with lift coefficient uncertainty.

```
<!--DAVEfunc-->
<fileReader> ...

<variableRef name="angleOfAttack" varID="Alpha_deg" units="#deg"/
</variableDef>

<variableRef name="CL_u" varID="CL_u" units="#nl"/
<description> Coefficient of lift with a symmetric Gaussian uncertainty of 2%: correlated with C_{m_u} uncertainty. </description>
<uncertainty effect="multiplicative"> </
</variableDef>

<variableRef name="Cm_u" varID="Cm_u" units="#nl"/
<description> Coefficient of pitching moment with a symmetric Gaussian uncertainty distribution of 30%; correlated directly with lift uncertainty. </description>
<output/>
<uncertainty effect="percentage"/>
</variableDef>
```

NESC Request No.: 09-00598
Lift coefficient has a nominal value that varies with angle-of-attack according to a nonlinear 1D table (given in the "Nominal CL" table defined in this example). When performing parametric variations, \( CL_u \) can take on a multiplicative variation of up to 20% (3-sigma) with a Gaussian distribution.

This element indicates that the variation of lift coefficient correlates directly with the variation in pitching moment coefficient.

Pitching-moment coefficient has a nominal value that varies as a function of angle-of-attack, according to a nonlinear 1D table (given in the "Nominal Cm" table defined in this example). When performing parametric variations, \( Cm_u \) can take on a 30% variation (3-sigma) with a Gaussian distribution.

This element indicates that the variation of pitching moment correlates directly with the variation in lift coefficient.

6.5. Additional DAVE-ML conventions

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

6.5.1. Ordering of points

In listing data points in multi-dimensional table definitions, the sequence should be such that the last dimension changes most rapidly. In other words, a table that is a function of \( f(a,b,c) \) should...
list the values of \( f(1.1,1) \), then \( f(1.1,2) \), etc. This may be different than, say, a Fortran DATA, Matlab® script or C++ initialization statement; the responsibility for mapping the data in the correct sequence in the model realization is left up to the implementer.

Figure 6 (p. 56) below shows how a 3D table is represented as an unraveled list of points.

6.5.2. Locus of action of moments

It is recommended that all forces 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, piloting) provide total forces and moments about this reference point and leave the transfer of moments to the center of mass to the equations of motion.

6.5.3. Decomposition of flight dynamics subsystems

It is recommended that a vehicle’s flight dynamics 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 familiar to the authors.

6.5.4. Date format in DAVE-ML

The recommended way of representing dates in DAVE-ML documentation, especially date attributes and creation date elements, is numerically in the order YYYY-MM-DD. Thus, July 15, 2003 is given as 2003-07-15. This formatting convention conforms to the ISO-8601 standard for representing dates. [ISO8601]

6.5.5. Common sign convention notation

A convention for indicating positive sign conventions for measurements is included in Annex A of the draft AIAA Flight Dynamics Model Exchange Standard [AIAA10]. These acronyms, if applicable, should be used whenever possible to enhance communications.

6.5.6. Units-of-measure abbreviation

Each variable definition includes a mandatory units attribute. This attribute gives the units-of-measure for the signal represented by the variable and either ‘d’ (for non-dimensional) or blank
if the signal is a dimensionless quantity or flag. In addition, the use of 'frac' to indicate a fraction (0-1) or 'per' to indicate a percentage (0-100 or more) is encouraged.

Informally, this attribute can take on any reasonable abbreviation for a set of units that might be understandable by the intended audience, in either set of units (English or ISO). For greater re-usability, it is recommended that the set of measurements listed in the AIAA Flight Dynamics Model Exchange Standard [AIAA10] (of which this document is a part) be used. The Standard recommends how to encode powers of units (f s 2 for ft/sec^2, for example).

6.5.7. Internal identifiers

Identifiers are used throughout DAVE-ML to connect signals, functions, modification records and reference documents with each other, e.g., utID, gtID, bpID, refID, etc. These identifiers are character strings that must comply with the XML specification for Names [W3C-XNCML]. They must start with a character or underscore, may not start with "xml" or "XML," and may not contain colons, among other restrictions. In addition, the identifiers must be unique within a single DAVE-ML file. See the XML [W3C-XNCML] (p. 132) for more information regarding valid XML Names.

6.5.8. DAVE-ML Namespace

The XML standard allows for namespace domains, in which element names belong to either the empty (null) namespace or to a namespace that belongs to a particular XML grammar. Namespaces are identified by a uniform resource identifier (URI) that is fashioned, similar to a URL, in order that uniqueness is guaranteed.

The DAVE-ML namespace should be defined in the top-level element as follows:

```xml
```

This will allow DAVE-ML models to be embedded in other XML documents without confusion. Note that this general URI is not a URL; HTTP queries at that address may not lead to any useful information.

The reference element can include two elements that belong to the World-Wide Web Consortium (W3C)'s XLink protocol [W3C-XLINK]; they are defined with an xlink prefix which actually refers to the namespace uniquely defined with the http://www.w3.org/1999/xlink URI. If external links to documents will be included in a DAVE-ML document, the top-level element (currently reference) must include a namespace declaration (which looks like, but technically is not, an attribute):

```xml
<reference xlink:xlink="http://www.w3.org/1999/xlink"/>
```

Similarly, the MathML-2 elements are normally defined in a MathML namespace, so any calculation defined using MathML-2 notation should be conducted inside the MathML namespace:

```xml
<math xmlns="http://www.w3.org/1998/Math/MathML"/>
```
6.6. Planned major elements

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

- Support for vector and matrix variables and math operations.
- Subsystem models, to support hierarchical decomposition and problem abstraction.
- State variables, both discrete and continuous, to support dynamic models.
- Dynamic (time-history) data file format, to allow for validation check-cases for dynamic models.
7. Further information

Further information, background, and the latest DTD and example models of some aircraft data packages can be found at the DAVE-ML web site: http://daveml.org
8. Element references and descriptions

This section lists the XML tags, or elements, that make up the DAVE-ML grammar. They are listed alphabetically in two sub-sections; the first is a short description of each element; the second sub-section gives details on the element, attributes, and sub-elements.

8.1. Alphabetical list of elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>address</td>
<td>Street address or other contact information of an author [Deprecated.]</td>
</tr>
<tr>
<td>author</td>
<td>Gives name and contact information for originating party of the associated data</td>
</tr>
<tr>
<td>bounds</td>
<td>Describes limits or standard deviations of statistical uncertainties</td>
</tr>
<tr>
<td>bpRef</td>
<td>Reference to a breakpoint definition</td>
</tr>
<tr>
<td>bpVals</td>
<td>String of white space- or comma-separated values of breakpoints</td>
</tr>
<tr>
<td>breakpointDef</td>
<td>Defines breakpoint sets to be used in model</td>
</tr>
<tr>
<td>breakpointRefs</td>
<td>A list of breakpoint reference (bpRefs)</td>
</tr>
<tr>
<td>calculation</td>
<td>Used to delimit a MathML v2 calculation</td>
</tr>
<tr>
<td>checkData</td>
<td>Gives verification data for encoded model</td>
</tr>
<tr>
<td>checkInputs</td>
<td>Lists input values for check case</td>
</tr>
<tr>
<td>checkOutputs</td>
<td>Lists output values for check case</td>
</tr>
<tr>
<td>confidenceBound</td>
<td>Defines the confidence in a function [Deprecated]</td>
</tr>
<tr>
<td>contactInfo</td>
<td>Provides multiple contact information associated with an author or agency</td>
</tr>
<tr>
<td>correlatesWith</td>
<td>Identifies other functions or variables whose uncertainty</td>
</tr>
<tr>
<td>correlation</td>
<td>correlates with our random value</td>
</tr>
<tr>
<td>creationDate</td>
<td>Gives date of creation of entity</td>
</tr>
<tr>
<td>dataType</td>
<td>Defines each point of an ungridded table</td>
</tr>
<tr>
<td>dataTable</td>
<td>Lists the values of a table of function or uncertainty data</td>
</tr>
<tr>
<td>DAVEfunc</td>
<td>Root level element</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>dependentVarPts</td>
<td>Defines output breakpoint values</td>
</tr>
<tr>
<td>dependentVarRef</td>
<td>Identifies the signal to be associated with the output of a function</td>
</tr>
<tr>
<td>description</td>
<td>Verbal description of an entity</td>
</tr>
<tr>
<td>documentRef</td>
<td>Reference to an external document</td>
</tr>
<tr>
<td>extraDocRef</td>
<td>Allows multiple documents to be associated with a single modification event</td>
</tr>
<tr>
<td>fileCreationDate</td>
<td>Gives date of creation of entity [Deprecated]</td>
</tr>
<tr>
<td>fileHeader</td>
<td>States source and purpose of file</td>
</tr>
<tr>
<td>fileVersion</td>
<td>Indicates the version of the document</td>
</tr>
<tr>
<td>function</td>
<td>Defines a function by combining independent variables, breakpoints, and tables</td>
</tr>
<tr>
<td>functionCreationDate</td>
<td>Date of creation of a function table [Deprecated]</td>
</tr>
<tr>
<td>functionDefn</td>
<td>Defines a function by associating a table with other information</td>
</tr>
<tr>
<td>griddedTable</td>
<td>Definition of a gridded table; associates breakpoint data with table data [Deprecated]</td>
</tr>
<tr>
<td>griddedTableDef</td>
<td>Defines an orthogonally gridded table of data points</td>
</tr>
<tr>
<td>griddedTableRef</td>
<td>Reference to a gridded table definition</td>
</tr>
<tr>
<td>independentVarPts</td>
<td>Simple definition of independent breakpoints</td>
</tr>
<tr>
<td>independentVarRef</td>
<td>References a predefined signal as an input to a function</td>
</tr>
<tr>
<td>internalValues</td>
<td>A dump of internal model values for debugging check-cases</td>
</tr>
<tr>
<td>isControl</td>
<td>Flag to identify a model control parameter</td>
</tr>
<tr>
<td>isDisturbance</td>
<td>Flag to identify a model disturbance input</td>
</tr>
<tr>
<td>isInput</td>
<td>Flag to identify a model input variable</td>
</tr>
<tr>
<td>isOutput</td>
<td>Flag to identify non-obvious output signals from model</td>
</tr>
<tr>
<td>isState</td>
<td>Flag to identify a state variable within a dynamic model</td>
</tr>
<tr>
<td>isStateDeriv</td>
<td>Flag to identify a state derivative within a dynamic model</td>
</tr>
<tr>
<td>isStdAIAA</td>
<td>Flag to identify standard AIAA simulation variable</td>
</tr>
</tbody>
</table>
modificationRecord To associate a reference single letter with a modification event
modificationRef Reference to associated modification information
normalPDF Defines a normal (Gaussian) probability density function
provenance Describes origin or history of the associated information
provenanceRef References a previously defined data provenance description
reference Describes an external document
signal Documents an internal DAVE-ML signal (value, units, etc.)
signalID Gives the internal identifier of a varDef [Deprecated]
signalName Gives the external name of an input or output signal
signalUnits Gives the unit-of-measure of an input or output variable
signalValue Gives the value of a check-case signal variable
staticShot Used to check the validity of the model once instantiated by the receiving facility or tool.
tol Specifies the tolerance of value matching for model verification
uncertainty Describes statistical uncertainty bounds and any correlations for a parameter or function table.
ungriddedTable Definition of an ungridded set of function data [Deprecated]
ungriddedTableDef Defines a table of data, each with independent coordinates
ungriddedTableRef Reference to an ungridded table
uniformPDF Defines a uniform (constant) probability density function
variableDef Defines signals used in DAVE-ML model
variableRef Reference to a variable definition
varID Gives the internal identifier of a varDef

8.2. Element descriptions

This section lists each element in detail, giving the name, content model, attributes, possible parent elements, allowable children elements, and any future plans for the element (such as deprecation).
address

address — Street address or other contact information of an author [Deprecated.]

Content model
address :
(#PCDATA)

Attributes
NONE

Possible parents
author

Allowable children
NONE

Future plans for this element

This element has been subsumed by the contactInfo element below.
author

author — Gives name and contact information for originating party of the associated data

Content model

author : name, org, [xns], [email]
(address* | contactInfo*)

Attributes

name The name of the author or last modifier of the associated element’s data.
org The author’s organization.
xns (optional) (deprecated) The eXtensible Name Service identifier for the author.
email (optional) (deprecated) The e-mail address for the primary author.

Description

author includes alternate means of identifying author using XNS or normal e-mail/address. The address sub-element is to be replaced with the more complete contactInfo sub-element.

Possible parents

fileHeader
modificationRecord
provenance

Allowable children

address
contactInfo

Future plans for this element

Both the xns and email attributes are deprecated and will be removed. XNS was a proposed Internet technology (eXtensible Name Service) to reduce spam that didn't catch on. It is replaced with the 'name' sub-element as a single means to identify an individual or corporation in lieu of typical (and quickly dated) e-mail, phone, or address information. The address element itself is deprecated and should be replaced with the contactInfo element.
DAVE-ML 2

bounds

bounds — Describes limits or standard deviations of statistical uncertainties

Content model

bounds:

{#PCDATA | dataTable | variableDef | variableRef}*

Attributes

NONE

Description

This element contains some description of the statistical limits to the values the citing parameter element might take on. This can be in the form of a scalar value, a private dataTable, or a variableRef. In the more common instance, this element will either be a scalar constant value or a simple table whose dimensions must match the parent nominal function table and whose independent variables are identical to the nominal table. It is also possible that this limit be determined by an independent variable, either previously defined or defined in-line with this element. It does not make sense to have a dataTable cited if this bounds element is associated with anything other than an identically shaped function table.

Possible parents

normalPDF
uniformPDF

Allowable children

dataTable
variableDef
variableRef
**bpRef**

bpRef — Reference to a breakpoint definition

**Content model**

```
bpRef : bpID
EMPTY
```

**Attributes**

bpID — The internal identifier for a breakpoint set definition.

**Description**

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

**Possible parents**

breakpointRefs

**Allowable children**

NONE
bpVals

   bpVals — String of white space- or comma-separated values of breakpoints

Content model

   bpVals :
   (#PCDATA)

Attributes

   NONE

Description

   bpVals is a set of breakpoints (i.e., 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 2D coefficient value table.

Possible parents

   breakpointDef

Allowable children

   NONE
breakpointDef

breakpointDef — Defines breakpoint sets to be used in model

Content model

breakpointDef : [name], bpID, [units]
    description?
    bpVals

Attributes

name (optional) The name of the breakpoint set.
bpID The internal identifier for the breakpoint set.
units (optional) The units of measure for the breakpoint set.

Description

A breakpointDef lists gridded table breakpoints. Since these are separate from function data they may be reused.

Possible parents

DAVEfunc

Allowable children

description
bpVals
breakpointRefs

breakpointRefs — A list of breakpoint reference (bpRefs)

Content model

```
breakpointRefs :
  bpRefs
```

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
```
DAVE-ML 2

calculation

calculation — Used to delimit a MathML v2 calculation

Content model

calculation :
  math (p. 15)

Attributes

NONE

Description

The calculation element is MathML 2 content markup describing how the signal is calculated. The calculation may include both constants and variables; other variables are included by using their varID string in a MathML content identifier (ci) element.

Possible parents

variableDef

Allowable children

  math (p. 15)
checkData

checkData — Gives verification data for encoded model

Content model

checkData :  
(provenance | provenanceRef)?
staticShot;

Attributes

NONE

Description

This top-level element is the place-holder for verification data of various forms for the encoded model. It will include static check cases, trim shots, and dynamic check case information. The provenance sub-element is now deprecated and has been moved to individual staticShots; it is allowed here for backwards compatibility.

Possible parents

DAVEfunc

Allowable children

provenance
provenanceRef
staticShot
checkInputs
   checkInputs — Lists input values for check case

Content model
   checkInputs :
      signal+

Attributes
   NONE

Description
   Specifies the contents of the input vector for the given check case.

Possible parents
   staticShot

Allowable children
   signal
```python
cHECKOUTPUTS

checkOutputs — Lists output values for check case

Content model

checkOutputs :
  signal:

Attributes

NONE

Description

 Specifies the contents of the output vector for the given check case.

Possible parents

staticShot

Allowable children

signal
```
confidenceBound

confidencerBound — Defines the confidence in a function [Deprecated]

Content model

confidenceBound : value

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 place-holder and will be removed in a future version of DAVE-ML.

Possible parents

griddedTable

ungriddedTable

Allowable children

NONE

Future plans for this element

Deprecated. Used only in deprecated [un]griddedTable elements. Use uncertainty element instead.
contentInfo

contentInfo — Provides multiple contact information associated with an author or agency

Content model

collection : [contentInfoType], [contactLocation]
(#PCDATA)

Attributes

\n
contactInfoType (optional) Indicates type of information being conveyed (enumerated).
  • address
  • phone
  • fax
  • email
  • iname
  • web

contactLocation (optional) Indicates which location is identified. Default is professional (enumerated).
  • professional
  • personal
  • mobile

Description

Used to provide contact information about an author. Use contactInfoType to differentiate what information is being conveyed and contactLocation to denote location of the address.

Possible parents

author

Allowable children

NONE
correlatesWith

correlatesWith — Identifies other functions or variables whose uncertainty correlates with our random value

Content model

correlatesWith : varID

Attributes

varID Identifies the variable or function output that will depend on this function’s or variable’s randomness.

Description

When present, this element indicates the parent function or variable is correlated with the referenced other function or variable in a linear sense. This alerts the application that the random number used to calculate this function’s or variable’s immediate value will be used to calculate another function’s or variable’s value.

Possible parents

normalPDF

Allowable children

NONE
correlation

correlation — Indicates the linear correlation of this function's or variable's randomness with a previously computed random variable.

Content model

correlation : varID, corrcoef

Attributes

varID Identifies the variable or function output that helps determine the value of this random variable or function.

corrCoef Indicates the amount of correlation between this variable and the referenced variable. The value should be between -1 and +1; 0 indicates no correlation, +1 indicates perfect correlation, and -1 indicates inverse (negative) correlation.

Description

When present, this element indicates the parent function or variable is correlated with the referenced other function or variable in a linear sense and gives the correlation coefficient for determining this function's random value based upon the correlating function(s)'s random value.

Possible parents

normalPDF

Allowable children

NONE
creationDate

creationDate — Gives date of creation of entity

Content model

creationDate : date

Attributes

date The date of creation of the entity, in ISO 8601 (YYYY-MM-DD) format.

Description

creationDate is simply a string with a date in it. We follow ISO 8601 and use dates like "2004-01-02" to refer to January 2, 2004.

Possible parents

fileHeader

provenance

Allowable children

NONE
**dataPoint**

*dataPoint* — Defines each point of an ungridded table

**Content model**

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

**Attributes**

- **modID** (optional) The internal 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 an ungridded table, the interpreting application is required to handle what may be unsorted data.

**Possible parents**

- ungriddedTableDef
- ungriddedTable

**Allowable children**

NONE
dataTable

dataTable — Lists the values of a table of function or uncertainty data

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. The table data point values are specified as comma- or white space-separated values in conventional floating-point notation (0.986582E-06) in a single long sequence as if the table had been unraveled with the last-specified dimension changing most rapidly. Line breaks are to be ignored. Comments may be embedded in the table to promote [human] readability, with appropriate escaping characters. A dataTable element can also be used in an uncertainty element to provide duplicate uncertainty bound values.

Possible parents

griddedTableDef
griddedTable
bounds

Allowable children

NONE

80
DAVE-ML 2

DAVEfunc

DAVEfunc — Root level element

Content model

DAVEfunc : xmlns
fileHeader
variableDef+
breakpointDef+
griddedTableDef*
ungriddedTableDef*
function*
checkData?

Attributes

xmlns This attribute specifies that the default namespace for un-prefixed elements is this DTD.

Description

Root element is DAVEfunc, composed of a file header element followed by one or more variable
definitions and zero or more breakpoint definitions, gridded or ungridded table definitions, and
function elements.

Possible parents

NONE - ROOT ELEMENT

Allowable children

fileHeader
variableDef
breakpointDef
griddedTableDef
ungriddedTableDef
function
checkData
**Flight Simulation Model Exchange**

**Content model**

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

**Attributes**

- **varID**: The internal identifier 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 comma- or white space-delimited list 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 do not share breakpoint or table values with other functions.

**Possible parents**

- `function`

**Allowable children**

NONE
DAVE-ML 2

dependentVarRef

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

Content model

dependentVarRef : varID

Attributes

  varID     The internal 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
DAVE-ML 2

description

description — Verbal description of an entity

Content model

description :  
(#PCDATA)

Attributes

NONE

Description

The description element is a textual description of an entity. The full UNICODE character set is supported by XML but may not be available in all processing applications.

Possible parents

fileHeader
variableDef
breakpointDef
griddedTableDef
ungriddedTableDef
function
reference
modificationRecord
provenance
staticShot

Allowable children

NONE
documentRef

    documentRef — Reference to an external document

Content model

documentRef : (docID), refID

EMPTY

Attributes

docID    (optional) (deprecated) The internal identifier of a reference definition element.

refID    The internal identifier of a reference definition element.

Possible parents

provenance

Allowable children

NONE

Future plans for this element

The ‘docID’ attribute is deprecated; it has been renamed ‘refID’ to match its use in the ‘reference’ element. This attribute will be removed in a future version of DAVE-ML.
extraDocRef

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

Content model

extraDocRef : refID

Attributes

refID The internal identifier of a reference definition element.

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
fileCreationDate

fileCreationDate -- Gives date of creation of entity [Deprecated]

Content model

fileCreationDate : date

Attributes

date The date of the file, in ISO 8601 (YYYY-MM-DD) format.

Description

fileCreationDate is simply a string with a date in it. We follow ISO 8601 and use dates like "2004-01-02" to refer to January 2, 2004. Its use is now deprecated in favor of the simpler creationDate.

Possible parents

fileHeader

Allowable children

NONE

Future plans for this element

The fileCreationDate and functionCreationDate have been replaced with a new creationDate multipurpose element.
fileHeader

fileHeader — States source and purpose of file

Content model

def fileHeader : [name]
  author+ (creationDate | fileCreationDate)
  fileVersion?
  description?
  reference*
  modificationRecord*
  provenance*

Attributes

  name   (optional) The name of the file.

Description

The header element requires at least one author and a creation date; optional content includes version indicator, description, references, and modification records.

Possible parents

dAVEfunc

Allowable children

  author
  creationDate
  fileCreationDate
  fileVersion
  description
  reference
  modificationRecord
  provenance
FileVersion

FileVersion — Indicates the version of the document

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
function

function — Defines a function by combining independent variables, breakpoints, and tables

Content model

function : name
  description?  
    (provenance | provenanceRef)?
      
      {  
        independentVarPts=
        dependentVarPt
      }
    or
      {  
        independentVarRef=
        dependentVarRef
        functionDefn
      }

Attributes

  name  The name of this function.

Description

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

Possible parents

  DAVEnfunc

Allowable children

  description
  provenance
  provenanceRef
  independentVarPts
  dependentVarPt
  independentVarRef
  dependentVarRef
  functionDefn
functionCreationDate

functionCreationDate — Date of creation of a function table [Deprecated]

Content model

functionCreationDate : date

EMPTY

Attributes

date The creation date of the function, in ISO 8601 (YYYY-MM-DD) format.

Description

functionCreationDate is simply a string with a date in it. We follow ISO 8601 and use dates like "2004-01-02" to refer to January 2, 2004. Its use is now deprecated in favor of the simpler creationDate.

Possible parents

provenance

Allowable children

NONE

Future plans for this element

The fileCreationDate and functionCreationDate have been replaced with a new creationDate multipurpose element.
functionDefn

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

Content model

functionDefn : {name}
  (griddedTableRef | griddedTableDef | griddedTable | ungriddedTableRef
  | ungriddedTableDef | 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
griddedTableDef
griddedTable
ungriddedTableRef
ungriddedTableDef
ungriddedTable
griddedTable

griddedTable — Definition of a gridded table; associates breakpoint data with table data
[Deprecated]

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

Future plans for this element

Deprecated. Use griddedTableDef instead.
**griddedTableDef**

griddedTableDef — Defines an orthogonally gridded table of data points

**Content model**

```
griddedTableDef : [name], [gtID], [units]
description?  
(pregdence | provenanceRef)?
breakpointRefs?
uncertainty?
dataTable
```

**Attributes**

- **name**  (optional) The name of the gridded table.
- **gtID**  (optional) An internal identifier for the table. Required if table is to be reused by another function or is defined outside of a function.
- **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 may be specified separately from the actual function declaration; if so, it requires a gtID identifier attribute so that it may be used by multiple functions.

**Possible parents**

- DAVEfunc
  - functionDefn

**Allowable children**

- description
- provenance
- provenanceRef
- breakpointRefs
- uncertainty
- dataTable
griddedTableRef

griddedTableRef — Reference to a gridded table definition

Content model

griddedTableRef : gtID

Attributes

gtID The internal identifier of a gridded table definition.

Possible parents

functionDefn

Allowable children

NONE
**independentVarPts**

*independentVarPts* — Simple definition of independent breakpoints

**Content model**

```
independentVarPts : varID, [name], [units], [sign], [extrapolate], [interpolate]
```

**Attributes**

- **varID**
  - The internal identifier 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 (default is neither) *(enumerated).*
    - neither
    - min
    - max
    - both

- **interpolate** *(optional)*
  - Interpolation flags for independent variable (default is linear) *(enumerated).*
    - discrete
    - floor
    - ceiling
    - linear
    - quadraticSpline
    - cubicSpline

**Description**

An independentVarPts element is a simple white space- or comma-separated 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 (default is 'neither,' meaning the value of the table is held
constant at the nearest defined value). An optional interpolate attribute indicates how to perform
the interpolation within the table (supporting discrete, linear, cubic or quadratic splines). There
are three different discrete options: 'discrete' means nearest-neighbor, with an exact mid-point
value being rounded in the positive direction; 'ceiling' means the function takes on the value
associated with the next (numerically) higher independent breakpoint as soon as the original
value is exceeded; and 'floor' means the function holds the value associated with each breakpoint
until the next (numerically) higher breakpoint value is reached by the independent argument. The
default interpolation attribute value is 'linear.' This element is used for simple functions that do
not share breakpoint or table values with other functions.

Possible parents

function

Allowable children

NONE
**independentVarRef**

- **NESC Request No.:** 09-00598

**Content model**

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

**Attributes**

- `varID` - The internal 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 (default is neither) (enumerated).
  - neither
  - min
  - max
  - both

- `interpolate` - (optional) Interpolation flags for independent variable (default is linear) (enumerated).
  - discrete
  - floor
  - ceiling
  - linear
  - quadraticSpline
  - cubicSpline

**Description**

An independentVarRef more fully describes the input mapping of the function by pointing to a separate breakpoint definition element. An optional extrapolate attribute describes how to extrapolate the output value when the input value exceeds specified values (default is 'neither,'
meaning the value of the table is held constant at the nearest defined value). An optional
interpolate attribute indicates how to perform the interpolation within the table (supporting
discrete, linear, cubic or quadratic splines). There are three different discrete options: 'discrete'
means nearest-neighbor, with an exact mid-point value being rounded in the positive direction;
'floor' means the function takes on the value associated with the next (numerically) higher
independent breakpoint as soon as original value is exceeded; and 'ceiling' means the function
holds the value associated with each breakpoint until the next (numerically) higher breakpoint
value is reached by the independent argument. The default interpolation attribute value is 'linear.'
This element allows reuse of common breakpoint values for many tables but with possible
differences in interpolation or extrapolation for each use.

Possible parents

function

Allowable children

NONE
internalValues

internalValues — A dump of internal model values for debugging check-cases

Content model

internalValues :

Attributes

NONE

Description

Provides a set of all internal variable values to assist in debugging recalcitrant implementations of DAVE-ML import tools.

Possible parents

staticShot

Allowable children

signal
DAVE-ML 2

isControl

isControl — Flag to identify a model control parameter

Content model

   isControl : EMPTY

Attributes

NONE

Description

The presence of an isControl element indicates that this signal is a simulation control parameter used to vary the operation of the model, e.g., the time step size. Such parameters should be ignored when performing linear model extraction (for example) and should not significantly modify the dynamic behavior of the model.

Possible parents

variableDef

Allowable children

NONE
isDisturbance

isDisturbance — Flag to identify a model disturbance input

Content model

```
isDisturbance := EMPTY
```

Attributes

NONE

Description

The presence of an isDisturbance element indicates that this signal is an external disturbance input to the model and can be ignored when performing linear model extraction (for example). Such parameters should not significantly modify the nominal dynamic behavior of the model.

Possible parents

```
variableDef
```

Allowable children

NONE
isInput

isInput — Flag to identify a model input variable

Content model

   isInput ,
   EMPTY

Attributes

   NONE

Description

   The presence of an isInput element indicates that this variable is an input signal to the model.

Possible parents

   variableDef

Allowable children

   NONE
DAVE-ML 2

isOutput

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

Content model

isOutput : EMPTY

Attributes

NONE

Description

The presence of the isOutput element indicates that this variable should be forced to be an output, even if it is used internally as an input elsewhere. Otherwise, the processing program may assume a signal defined with a calculation and used subsequently in the model is only an internal signal.

Possible parents

variableDef

Allowable children

NONE
isState

isState — Flag to identify a state variable within a dynamic model

Content model

isState : EMPTY

Attributes

NONE

Description

The presence of an isState element indicates that this variable is one of possibly multiple state variables in a dynamic model; this tells the processing entity that this is the output of an integrator (for continuous models) or a discretely updated state (for discrete models).

Possible parents

variableDef

Allowable children

NONE
**isStateDeriv**

*isStateDeriv* — Flag to identify a state derivative within a dynamic model

**Content model**

```xml
isStateDeriv : EMTH
```

**Attributes**

NONE

**Description**

The presence of an *isStateDeriv* element indicates that this variable is one of possibly several state derivative variables in a dynamic model; this tells the processing entity that this is the output of an integrator (for continuous models only).

**Possible parents**

`variableDef`

**Allowable children**

NONE
isStdAIAA

isStdAIAA — Flag to identify standard AIAA simulation variable

Content model

isStdAIAA:

EMPTY

Attributes

NONE

Description

The presence of an isStdAIAA element indicates that this variable is one of the standard AIAA variable names which should be recognizable exterior to this module (e.g. AngleOfAttack_deg). This flag should assist importing tools in determining when an input or output should match a facility-provided signal name without requiring further information.

Possible parents

variableDef

Allowable children

NONE
modificationRecord

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

Content model

modificationRecord : modID, date, [refID]

author+

description?

extraDocRef*

Attributes

modID A single letter used to identify all modified data associated with this modification record.

date The date of the modification, in ISO 8601 (YYYY-MM-DD) format.

refID (optional) A reference to a predefined document that describes the reason for this modification.

Description

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

Possible parents

fileHeader

Allowable children

author
description
extraDocRef
modRef

modRef — Reference to associated modification information

Content model

modRef; modID

Attributes

modID The internal identifier of a modification definition.

Possible parents

provenance

Allowable children

NONE
normalPDF

normalPDF — Defines a normal (Gaussian) probability density function

Content model

```xml
normalPDF : numSigma
  bounds
  correlatesWith
  correlation
```

Attributes

- `numSigma` Indicates how many standard deviations is represented by the uncertainty values given later. Integer value > 0.

Description

In a normally distributed random variable, a symmetrical distribution of given standard deviation is assumed about the nominal value (which is given elsewhere in the parent element). The correlatesWith sub-element references other functions or variables that have a linear correlation to the current parameter or function. The correlation sub-element specifies the correlation coefficient and references the other function or variable whose random value helps determine the value of this parameter.

Possible parents

- `uncertainty`

Allowable children

- `bounds`
  - correlatesWith
  - correlation
provenance

provenance — Describes origin or history of the associated information

Content model

provenance :  [provId]
  author:
    (creationDate | functionCreationDate)
    documentRef:
    modificationRef:
    description?

Attributes

provID    (optional) This attribute allows the provenance information defined here to be cited elsewhere.

Description

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

Possible parents

fileHeader
variableDef
girdedTableDef
ungirdedTableDef
function
checkData
staticShot

Allowable children

author
creationDate
functionCreationDate
documentRef
modificationRef
description
provenanceRef

provenanceRef — References a previously defined data provenance description

Content model

```xml
  provenanceRef : provID
```

Attributes

```xml
  provID     The internal identifier for a previously defined provenance.
```

Description

When the provenance of a set of several data is identical, the first provenance element should be given a provID and referenced by later provenanceRef elements.

Possible parents

```xml
  variableDef
gridDEDTableDef
ungirdDEDTableDef
function
checkData
staticShot
```

Allowable children

```xml
NONE
```
reference

reference — Describes an external document

Content model

reference, xlink:xlink, xlink:type, refID, author, title, classification, accession, date, xlink:href

description

Attributes

xmlns:xlink The value of this attribute must be "http://www.w3.org/1999/xlink". If omitted, it should be provided by the parser since it is specified in the DTD.
xlink:type The value of this attribute must be "simple". If omitted, it should be provided by the parser since it is specified in the DTD.
refID An internal identifier for this reference definition.
author The name of the author of the reference.
title The title of the referenced document.
classification (optional) The security classification of the document.
accession (optional) The accession number (ISBN or organization report number) of the document.
date The date of the document, in ISO 8601 (YYYY-MM-DD) format.
xlink:href (optional) A URL to an on-line copy of the referenced document.

Description

This element gives identifying (citation) information to an external, possibly on-line, reference document, including a user-specified author, title, classification, accession number, date and URL.

Possible parents

fileHeader

Allowable children

description
signal

signal — Documents an internal DAVE-ML signal (value, units, etc.)

Content model

\[
\text{signal} ::
\begin{align*}
    \{ & \text{signalName} \\
    \text{signalUnits} \} \\
\text{OR} \{ & \text{varID} | \text{signalID} \} \\
& \text{signalValue} \\
\text{tol}\}
\end{align*}
\]

Attributes

NONE

Description

This element is used to document the name, ID, value, tolerance, and units of measure for check-cases. When used with checkInputs or checkOutputs, the signalName sub-element must be present (since check cases are viewed from "outside" the model); when used in an internalValues element, the varID sub-element should be used to identify the signal by its model-unique internal reference. When used in a checkOutputs vector, the tol element must be present. Tolerance is specified as a maximum absolute difference between the expected and actual value. The signalID sub-element is now deprecated in favor of the more consistent varID.

Possible parents

checkInputs
internalValues
checkOutputs

Allowable children

signalName
signalUnits
varID
signalID
signalValue
tol
signalID

signalID — Gives the internal identifier of a varDef [Deprecated]

Content model

signalID :

Attributes

NONE

Description

Used to specify the input or output varID. Now deprecated; reuse of varID is best practice.

Possible parents

signal

Allowable children

NONE

Future plans for this element

The signalID element has been deprecated with 2.0 in favor of the more consistent varID element and will be unsupported in a future release of this specification.
signalName

  signalName — Gives the external name of an input or output signal

Content model

  signalName : #PCDATA

Attributes

  NONE

Description

  Used inside a checkCase element to specify the input or output variable name

Possible parents

  signal

Allowable children

  NONE
signalUnits

    signalUnits — Gives the unit-of-measure of an input or output variable

Content model

    signalUnits : #PCDATA

Attributes

    NONE

Description

    Used inside a checkCase element to specify the units-of-measure for an input or output variable, for verification of proper implementation of a model.

Possible parents

    signal

Allowable children

    NONE
**signalValue**

- **signalValue** — Gives the value of a check-case signal/variable

**Content model**

```
<signalValue : (
  #PCDATA)
```

**Attributes**

- **NONE**

**Description**

- Used to give the current value of an internal signal or input/output variable, for verification of proper implementation of a model.

**Possible parents**

- **signal**

**Allowable children**

- **NONE**
staticShot

staticShot — Used to check the validity of the model once instantiated by the receiving facility or tool.

Content model

staticShot . name, [refID]
  description? [provenance | provenanceRef]?
  checkInputs
  internalValues?
  checkOutputs

Attributes

name

refID (optional) Points to a reference given in the file header.

Description

Contains a description of the inputs and outputs, and possibly internal values, of a DAVE-ML model in a particular instant of time.

Possible parents

checkData

Allowable children

description
provenance
provenanceRef
checkInputs
internalValues
checkOutputs
tol

tol — Specifies the tolerance of value matching for model verification

Content model
tol : retal : (transient)

Attributes

NONE

Description

This element specifies the allowable tolerance of error in an output value such that the model can be considered verified. It is assumed all uncertainty is removed in performing the model calculations. Tolerance is specified as a maximum absolute difference between the expected and actual value.

Possible parents

signal

Allowable children

NONE
uncertainty

uncertainty — Describes statistical uncertainty bounds and any correlations for a parameter or function table.

Content model

uncertainty : effect
(normalPDF | uniformPDF)

Attributes

effect Indicates how uncertainty bounds are interpreted. (enumerated):
  - additive
  - multiplicative
  - percentage
  - absolute

Description

The uncertainty element is used in function and parameter definitions to describe statistical variance in the possible value of that function or parameter value. Only Gaussian (normal) or uniform distributions of continuous random variable distribution functions are supported.

Possible parents

variableDef
griddedTableDef
ungriddedTableDef

Allowable children

normalPDF
uniformPDF
ungriddedTable

ungriddedTable — Definition of an ungridded set of function data [Deprecated]

Content model

ungriddedTable : {name}
    confidenceBound?
    dataPoint+

Attributes

name (optional) The name of the ungridded table being defined.

Possible parents

functionDefn

Allowable children

    confidenceBound
    dataPoint

Future plans for this element

Deprecated. Use ungriddedTableDef instead.
ungriddedTableDef

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

Content model

    ungriddedTableDef : [name], [utID], [units]
    description?
    (provenance | provenanceRef)?
    uncertainty?
    dataPoint+

Attributes

    name (optional) The name of the ungridded table.
    utID (optional) An internal identifier for the table. Required if table is to be reused by
    another function or is defined outside of a function.
    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 may be specified separately from the actual function declaration; if so, it requires an
internal utID identifier attribute so that it may be used by multiple functions.

Possible parents

    DAVEfunc
    functionDefn

Allowable children

    description
    provenance
    provenanceRef
    uncertainty
    dataPoint
ungriddedTableRef

ungriddedTableRef — Reference to an ungridded table

Content model

ungriddedTableRef : utID

Attributes

utID The internal identifier of a ungridded table definition.

Possible parents

functionDefn

Allowable children

NONE
uniformPDF

uniformPDF — Defines a uniform (constant) probability density function

Content model

uniformPDF :
  bounds+

Attributes
NONE

Description
In a uniformly distributed random variable, the value of the parameter has equal likelihood of assuming any value within the (possibly asymmetric, implied by specifying two) bounds, which must bracket the nominal value (which is given elsewhere in the parent element).

Possible parents
uncertainty

Allowable children
bounds
variableDef

variableDef — Defines signals used in DAVE-ML model

Content model

variableDef : name, varID, units, (axisSystem), (sign), (alias), (symbol),
(initialValue)
description?
(provenance | provenanceDef)?
calculation?
(input | isControl | isDisturbance)?
isState?
isDataDeriv?
isOutput?
isRealIIA?
uncertainty?

Attributes

name
varID
units
axisSystem
sign
alias
symbol
(initialValue

The name of the signal being defined.
An internal identifier for the signal.
The units of the signal.
(optional) The axis in which the signal is measured.
(optional) The sign convention for the signal, if any.
(optional) Possible alias name (facility specific) for the signal.
(optional) UNICODE symbol for the signal.
(optional) An initial and possibly constant numeric value for the signal.

Description

variableDef elements provide wiring information (i.e., 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. The variable definition can include statistical information regarding the uncertainty of the values which it might take on, when measured after any calculation is performed. Information about the reason for inclusion or change to this element can be included in an optional provenance subelement.

Possible parents

DAVEfunc
bounds
DAVE-ML 2

Allowable children

description
provenance
provenanceRef
calculation
isInput
isControl
isDisturbance
isState
isStateDeriv
isOutput
isStdAIAA
uncertainty
variableRef

variableRef — Reference to a variable definition

Content model

variableRef : varID

EMPTY

Attributes

varID The internal identifier of a previous variable definition.

Possible parents

bounds

Allowable children

NONE
DAVE-ML 2

varID

varID — Gives the internal identifier of a varDef

Content model

varID : (#PCDATA)

Attributes

NONE

Description

Used to specify the input or output varID. Replaces earlier signalID element.

Possible parents

signal

Allowable children

NONE
DAVE-ML 2

The editors would like to acknowledge the contributions, encouragement and helpful suggestions for steering the evolution of DAVE-ML from the following individuals: Trey Arthur (NASA Langley Research Center), Jon Berndt (Jacobs Sverdrup), Geoff Brian (Australian DSTO), Randy Brumbaugh (Indigo Innovations), Giovanni A. Cignoni (University of Pisa), Bill Cleveland (NASA Ames Research Center), Jeremy Furtel (Delphi Research), Peter Grant (UITAS), Missy Hill (UNISYS), Hilary Keating (Fortburn Pty. Ltd.), Dennis Linse (SAIC), J. Dana McMinn (NASA Langley Research Center), Daniel M. Newman (formerly Ball Aerospace, now Quantitative Aeronautics), Riley Rainey (SDS International), Brent York (formerly NAVAIR, now Indra Systems, Inc.), and Curtis Zimmerman (NASA Marshall Space Flight Center).
References


DAVE-ML 2


Index

A
‘accession’ attribute
of ‘reference’ element, 113
‘address’ element (deprecated)
as child of ‘author’ element, 64
definition, 63
deprecation, 9
AIAA
Draft Standard S-119, 12
Modeling and Simulation Technical Committee, 13
Recommended Practice R-004-1992, 13
Standards, 12
web site, 12
‘alias’ attribute
of ‘variableDef’ element, 24, 126
Apollo, 30
Arthur, Jarvis A., III, 5, 6, 6, 130
atan2, 15
‘author’ element
as child of ‘fileHeader’ element, 21, 88
as child of ‘modificationRecord’ element, 108
as child of ‘provenance’ element, 111
definition, 64
of ‘reference’ element, 113
‘axisSystem’ attribute
of ‘variableDef’ element, 24, 126

B
Berndt, Jon S., 130
‘bounds’ element
as child of ‘normalPDF’ element, 110
as child of ‘uniformPDF’ element, 125
definition, 65
‘bplD’ attribute
of ‘bpRef’ element, 66
of ‘breakpointDef’ element, 30, 68
‘bpRef’ element
as child of ‘breakpointRefs’ element, 69
definition, 66
‘bpVals’ element
as child of ‘breakpointDef’ element, 30, 68
definition, 67
‘breakpointDef’ element
as child of ‘DAVEfunc’ element, 19, 81
definition, 68
example of, 31
introduction, 16
overview, 50
test, 20
breakpointRefs' element
as child of 'griddedTable' element, 93
as child of 'griddedTableDef' element, 32, 94
definition, 69
Brian, Geoffrey, 7, 8, 8, 8, 8, 8, 37, 130
Brumbaugh, Randy, 7, 130

C
'calculation' element
as child of 'variableDef' element, 25, 127
definition, 70
'checkData' element
as child of 'DAVEfunc' element, 20, 81
definition, 71
example of
with internal values, 44
without internal values, 43
introduction, 17
overview, 43
'checkInputs' element
as child of 'staticShot' element, 119
definition, 72
'checkOutputs' element
as child of 'staticShot' element, 119
definition, 73
Cignoni, Giovanni A., 7, 130
'classification' attribute
of 'reference' element, 113
Cleveland, William, 9, 10, 13, 130
'confidenceBound' element (deprecated)
as child of 'griddedTable' element, 93
as child of 'ungriddedTable' element, 122
definition, 74
'contactInfo' element
as child of 'author' element, 64
definition, 75
discussion, 9
'contactInfoType' attribute
of 'contactInfo' element, 75
'contactLocation' attribute
of 'contactInfo' element, 75
'corrCoef' attribute
of 'correlation' element, 77
'correlatesWith' element
as child of 'normalPDF' element, 110
DAVE-ML 2

definition, 76
'correlation' element
as child of 'normalPDF' element, 110
definition, 77
'creationDate' element
as child of 'fileHeader' element, 21, 88
as child of 'provenance' element, 111
definition, 78

D
'dataPoint' element
as child of 'ungriddedTable' element, 122
as child of 'ungriddedTableDef' element, 35, 123
definition, 79
'dataTable' element
as child of 'bounds' element, 65
as child of 'griddedTable' element, 93
as child of 'griddedTableDef' element, 32, 94
definition, 80
'date' attribute
of 'creationDate' element, 78
of 'fileCreationDate' element, 87
of 'functionCreationDate' element, 91
of 'modificationRecord' element, 108
of 'reference' element, 113
date format, 56
'DAVEmef' element
definition, 81
discussion, 16
overview, 16, 18
root element, 16
DAVE-ML
additional resources, 11
extending, 15
web site, 11
'dependentVarPts' element
as child of 'function' element, 40, 90
definition, 82
'dependentVarRef' element
as child of 'function' element, 40, 90
definition, 83
'description' element, 30
as child of 'breakpointDef' element, 68
as child of 'fileHeader' element, 21, 88
as child of 'function' element, 59, 90
as child of 'griddedTableDef' element, 32, 94
as child of 'modificationRecord' element, 108
as child of 'provenance' element, 111
as child of 'reference' element, 113
as child of 'staticShot' element, 119
  as child of 'ungriddedTableDef' element, 35, 123
  as child of 'variableDef' element, 25, 127
  definition, 84
'docID' attribute (deprecated)
  of 'documentRef' element, 85
'documentRef' element
  as child of 'provenance' element, 111
  definition, 85
DSTO, 8, 13, 37

E
'effect' attribute
  of 'uncertainty' element, 50, 121
'email' attribute (deprecated)
  of 'author' element, 64
'extraDocRef' element
  as child of 'modificationRecord' element, 108
  definition, 86
'extrapolate' attribute
  of 'independentVarPts' element, 96
  of 'independentVarRef' element, 98
  overview, 46

F
'FileCreationDate' element (deprecated)
  as child of 'fileHeader' element, 88
  definition, 87
'fileHeader' element
  as child of 'DAVEfunc' element, 19, 81
  definition, 88
  example of, 22
  introduction, 16
  overview, 21
'FileVersion' element
  as child of 'fileHeader' element, 21, 88
  definition, 89
'function' element
  as child of 'DAVEfunc' element, 20, 81
  definition, 90
  example of, 41
  example of simple, 42
  example of with internal table, 41
  introduction, 17
  overview, 38
  with an internal table, 20
'functionCreationDate' element (deprecated)
  as child of 'provenance' element, 111
  definition, 91
DAVE-ML 2

`functionDefn` element
  as child of `function` element, 40, 90
  definition, 92
function table
  dimensionality, 17
  reuse, 17
Furtak, Jeremy, 130
future plans, 58

G
GeneSim, 14
Grant, Peter, 130
`griddedTable` element (deprecated)
  as child of `functionDefn` element, 92
  definition, 93
deprecation, 10
`griddedTableDefn` element
  as child of `DAVEfunc` element, 20, 81
  as child of `functionDefn` element, 92
  definition, 94
example of, 33
introduction, 17
overview, 20, 31
reuse, 20
`griddedTableRef` element
  as child of `function` element, 40
  as child of `functionDefn` element, 92
  definition, 95
gidID' attribute
  of `griddedTableDefn` element, 32, 94
  of `griddedTableRef` element, 95

H
HDF5, 13
Hill, Melissa, 5, 130

I
ID strings, 57
`IndependentVarPts` element
  as child of `function` element, 39, 90
  definition, 96
`IndependentVarRef` element
  as child of `function` element, 40, 90
  definition, 98
`initialValue` attribute
  of `variableDef` element, 25, 126
`internalValues` element
  as child of `staticShot` element, 119
  definition, 100
DAVE-ML 2

'interpolate' attribute
   of 'independentVarPnts' element, 96
   of 'independentVarRef' element, 98
overview, 46
interpolation
   of ungridded data, 20
'tsControl' element
   as child of 'variableDef' element, 25, 127
definition, 101
'tsDisturbance' element
   as child of 'variableDef' element, 25, 127
definition, 102
'tsInput' element
   as child of 'variableDef' element, 25, 127
definition, 103
'tsOutput' element
   as child of 'variableDef' element, 25, 127
definition, 104
'tsState' element
   as child of 'variableDef' element, 26, 127
definition, 105
'tsStateDeriv' element
   as child of 'variableDef' element, 26, 127
definition, 106
'tsStdMAA' element
   as child of 'variableDef' element, 26, 127
definition, 107

K
Keating, Hilary, 130

L
Linse, Dennis, 5, 5, 5, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7, 130

M
'math' element
   as child of 'calculation' element, 70
MathML-2, 15
'max' attribute
   of 'independentVarRef' element, 98
McMinn, J. Dana, 130
'min' attribute
   of 'independentVarRef' element, 98
'modDay' attribute
   of 'dataPoint' element, 79
   of 'modificationRecord' element, 108
   of 'modificationRef' element, 109
'modificationRecord' element
   as child of 'fileHeader' element, 22, 88


DAME-ML 2

definition, 108
'modificationRef' element
as child of 'provenance' element, 111
definition, 109
moment reference center, 56

N
'N' attribute
of 'author' element, 64
of 'breakpointDef' element, 30, 68
of 'dependentVarPts' element, 82
of 'fileHeader' element, 88
of 'function' element, 39, 90
of 'functionDefn' element, 92
of 'griddedTable' element, 93
of 'griddedTableDef' element, 32, 94
of 'independentVarPts' element, 96
of 'staticShot' element, 119
of 'ungridedTable' element, 122
of 'ungridedTableDef' element, 34, 123
of 'variableDef' element, 24, 126
namespace, XML, 57
NASA
Ames Research Center, 13
Dryden Flight Research Center, 13
Langley Research Center, 13
Project Orion, 13
NASP, 13
Newman, Daniel M., 4, 5, 6, 6, 7, 7, 130

normalPDF element
as child of 'uncertainty' element, 50, 121
definition, 110
'numSigma' attribute
of 'normalPDF' element, 110

O
'org' attribute
of 'author' element, 64

P
points, ordering of, 56
'provenance' element
as child of 'checkData' element, 71
as child of 'fileHeader' element, 22, 88
as child of 'function' element, 39, 90
as child of 'griddedTableDef' element, 32, 94
as child of 'staticShot' element, 119
as child of 'ungridedTableDef' element, 35, 123
as child of 'variableDef' element, 25, 127
DAVE-ML 2

definition, 111
`provenanceRef` element
  as child of 'checkData' element, 71
  as child of 'function' element, 90
  as child of 'griddedTableDef' element, 94
  as child of 'staticShot' element, 119
  as child of 'ungriddedTableDef' element, 123
  as child of 'variableDef' element, 127
  definition, 112
  `provID` attribute
    of 'provenance' element, 111
    of 'provenanceRef' element, 112

R
Rainey, Riley, 130
'referenced' element
  as child of 'fileHeader' element, 21, 88
  definition, 113
  `refID` attribute
    of 'documentRef' element, 85
    of 'extraDocRef' element, 86
    of 'modificationRecord' element, 108
    of 'reference' element, 113
    of 'staticShot' element, 119

S
Schilling, Larry, 13
'sign' attribute
  of 'dependentVarPts' element, 82
  of 'independentVarPts' element, 96
  of 'variableDef' element, 24, 126
'signal' element
  as child of 'checkInputs' element, 72
  as child of 'checkOutputs' element, 73
  as child of 'internalValues' element, 100
  definition, 114
'signalID' element
  as child of 'signal' element, 114
  definition, 115
deprecation, 6
'signalName' element
  as child of 'signal' element, 114
  definition, 116
'signals', 16
'signalUnits' element
  as child of 'signal' element, 114
  definition, 117
'signalValue' element
  as child of 'signal' element, 114

NESC Request No.: 09-00598
DAVE-ML 2

definition, 118
sign convention, 56
'Statistic' element
  as child of 'checkData' element, 43, 71
definition, 119
introduction, 17
statistics
coding for, 49
subsystems, 56
'symbol' attribute
  of 'variableDef' element, 24, 126

T
'title' attribute
  of 'reference' element, 113
'tol' element
  as child of 'signal' element, 114
definition, 120
TSOLT, 13

U
'uncertainty' element
  as child of 'griddedTableDef' element, 32, 94
  as child of 'ungriddedTableDef' element, 35, 123
  as child of 'variableDef' element, 26, 127
definition, 121
discussion, 49
example of
  additive, 52
  normal distribution, 53
  percentage, 51
  with absolute bounds, 51
  with correlation, 54
'ungriddedTable' element (deprecated)
  as child of 'functionDef' element, 92
definition, 122
deprecation, 10

'ungriddedTableDef' element
  as child of 'DAVEfunc' element, 20, 81
  as child of 'functionDef' element, 92
definition, 123
example of, 35, 56
introduction, 17
overview, 20, 24
reuse, 20

'ungriddedTableRef' element
  as child of 'function' element, 40
  as child of 'functionDef' element, 92
definition, 124
DAVE-ML 2

'uniformPDF' element
  as child of 'uncertainty' element, 50, 121
definition, 125
'units' attribute
  of 'breakpointDef' element, 30, 68
  of 'dependentVarPts' element, 82
  of 'griddedTableDef' element, 32, 94
  of 'independentVarPts' element, 96
  of 'ungriddedTableDef' element, 34, 123
  of 'variableDef' element, 24, 126
'units-of-measure' attribute
  of 'ungriddedTableDef' element, 34, 123
  of 'ungriddedTableRef' element, 124

V
'value' attribute
  of 'confidenceBound' element, 74
'variableDef' element
  as child of 'bounds' element, 65
  as child of 'DAVEfunc' element, 19, 81
definition, 126
  example of, 26
  example of a local, 27
  example of calculated, 27, 28
  example with extension to MathML-2, 29
  introduction, 16
  order of appearance, 19
  overview, 23
'variableDefinitions' element
  'variableRef' element
  as child of 'bounds' element, 65
definition, 128
'varID' attribute
  as child of 'signal' element, 114
definition, 129
  of 'correlatesWith' element, 76
  of 'correlation' element, 77
  of 'dependentVarPts' element, 82
  of 'dependentVarRef' element, 83
  of 'independentVarPts' element, 96
  of 'independentVarRef' element, 98
  of 'variableDef' element, 24, 126
  of 'variableRef' element, 128

X
<xlink:href' attribute
  of 'reference' element, 113
<xlink:type' attribute

142
DAVE-ML 2

- of 'reference' element, 113
- XML, 11
- XML namespace, 57
- xmlns' attribute
  - of 'DAVEfun' element, 81
- xmlns:link' attribute
  - of 'reference' element, 113
- xmlns attribute (deprecated)
  - deprecation, 9
- of 'author' element, 64

Y
York, Brent, 9, 13, 130

Z
Zimmerman, Curtis, 5, 130
Flight Simulation Model Exchange
Appendices

Murri, Daniel G.; Jackson, E. Bruce

NASA Langley Research Center
Hampton, VA 23681-2199

National Aeronautics and Space Administration
Washington, DC 20546-0001

Unclassified - Unlimited
Subject Category 59 - Mathematical and Computer Sciences
Availability: NASA CASI (443) 757-5802

The NASA Engineering and Safety Center Review Board sponsored an assessment of the draft Standard, Flight Dynamics Model Exchange Standard, BSR/ANSI-S-119-201x (S-119) that was conducted by simulation and guidance, navigation, and control engineers from several NASA Centers. The assessment team reviewed the conventions and formats spelled out in the draft Standard and the actual implementation of two example aerodynamic models (a subsonic F-16 and the HL-20 lifting body) encoded in the Extensible Markup Language grammar. During the implementation, the team kept records of lessons learned and provided feedback to the American Institute of Aeronautics and Astronautics Modeling and Simulation Technical Committee representative. This document contains the appendices to the main report.

<table>
<thead>
<tr>
<th>NASA Engineering and Safety Center; Extensible Markup Language; American Institute of Aeronautics and Astronautics; Modeling and Simulation Technical Committee</th>
</tr>
</thead>
<tbody>
<tr>
<td>STI Help Desk (email: <a href="mailto:help@sti.nasa.gov">help@sti.nasa.gov</a>)</td>
</tr>
<tr>
<td>(443) 757-5802</td>
</tr>
</tbody>
</table>