USGS
science for a changing world

The Virtual Data Center Tagged-Format Tool—Introduction and Executive Summary

By John R. Evans[1], Melinda Squibb[2], Christopher D. Stephens[1], W.U. Savage[1], Hamid Haddadi[3], Charles A. Kircher[4], and Mahmoud M. Hachem[5]

Open-File Report 2008-1327

U.S. Department of the Interior
U.S. Geological Survey

[1]U.S. Geological Survey, Menlo Park, Calif.
[2]COSMOS Virtual Data Center, University of California, Santa Barbara (now at: NEESit, NEES Cyberinfrastructure Center, University of California, San Diego, La Jolla, Calif.).
[3]California Geological Survey, Office of Strong Motion Studies, Sacramento, Calif.
[4]Kircher and Associates, Palo Alto, Calif.
[5]Wiss, Janney, Elstner Assoc., Inc., Emeryville, Calif.

U.S. Department of the Interior
DIRK KEMPTHORNE, Secretary

U.S. Geological Survey
Mark D. Myers, Director

U.S. Geological Survey, Reston, Virginia: 2008

For product and ordering information:
World Wide Web: http://www.usgs.gov/pubprod
Telephone: 1-888-ASK-USGS

For more information on the USGS—the Federal source for science about the Earth, its natural and living resources, natural hazards, and the environment:
World Wide Web: http://www.usgs.gov
Telephone: 1-888-ASK-USGS

Suggested citation:
Evans, J.R., Squibb, Melinda, Stephens, C.D., Savage, W.U., Haddadi, Hamid, Kircher, C.A., and Hachem, M.M., 2008, The Virtual Data Center Tagged-Format Tool; introduction and executive summary: U.S. Geological Survey Open-File Report 2008-1327, 171 p. [http://pubs.usgs.gov/of/2008/1327/].

The Virtual Data Center Tagged-Format Tool: Introduction and Executive Summary

By John R. Evans, Melinda Squibb, Christopher D. Stephens, W.U. Savage, Hamid Haddadi, Charles A. Kircher, and Mahmoud M. Hachem

Executive Summary

This Report introduces and summarizes the new Virtual Data Center (VDC) Tagged Format (VTF) Tool, which was developed by a diverse group of seismologists, earthquake engineers, and information technology professionals for internal use by the COSMOS VDC and other interested parties for the exchange, archiving, and analysis of earthquake strong-ground-motion data.

Background

For many decades, specialized formats have been defined and used to encode and store digitized acceleration time-series data from strong-motion sensors, as well as information derived from these data (for example, the "response spectra" widely used by structural engineers for design and retrofit of structures subject to seismic hazards). While many strong-motion formats currently in use derive in some way from the California Institute of Technology Blue Book format (circa 1970), there is little accepted standardization. Additionally, as compared to other seismic disciplines, the strong-motion and engineering communities require much more extensive supporting and derivative data to adequately qualify a record; thus, the use of more modern formats, such as SEED and SAC, is precluded.

In many ways current strong-motion formats reflect an early period of digital computation and storage when such resources were limited by today's standards. These early facilities also relied on the column-formatted "flat files" used by Fortran and lacked modern information-technology (IT) notions, such as "structured" storage, "object-oriented" and "platform-independent" programming, markup languages like xml, and relational databases.

The COSMOS v1.20 fixed-field format successfully amalgamates the various derivatives of the "Blue Book" fixed-field format and other compatible formats and could be used as a standard mechanism for exchanging data among various institutions and individuals worldwide; indeed, it is being used in the exchange of data between several institutions within California (the California Integrated Seismic Network), for example.

There is an opportunity and a need for mechanisms of strong-motion formatting, data exchange, and processing that exploit modern resources and techniques. There is also an opportunity to create a viable international exchange and format-conversion medium—strong-motion seismology and earthquake engineering depend upon international exchange of data and results. For example, the data provided to the world by Taiwan's Central Weather Bureau for the Chi-Chi earthquake mainshock more than doubled the worldwide strong-motion database and addressed issues that were previously

untenable. In particular, the global scope of VDC activities necessitated a medium of conversion between the numerous extant, historical, and likely future formats for strong-motion data and products. This VTF tool is intended to fill this need. By analogy to telephony switching centers, the VTF tool will be used as a crossbar switch, interconnected and compatible with all sources and users of strong-motion data. The VTF, and specifically its xml mirror, currently serve as a primary data-storage, data-exchange, and operational tool within the VDC IT structure.

The limitations inherent to fixed-field formats and legacy IT techniques motivated our indepth review of strong-motion formatting, data exchange, operations, and management, resulting in this proposed VTF along with its exactly equivalent xml representation and provision of key user tools for both the VDC and other interested parties.

Approach

The VTF tool is designed with the following key goals and constraints:

1. To be readily and unambiguously readable throughout by both humans and computers;
2. To be expressly international in its viability, including support of non-English scripts and the use of English—the current international standard for science and engineering—as its basis;
3. To be inherently both "backward compatible" with older versions of the VTF and prospectively extensible with minimal effort by the VDC and all potential users (that is, no tag will ever be dropped from subsequent versions, so that software that can read a later version of the VTF will also be able to read legacy VTF files);
4. To be inherently self-documenting and autonomous (that is, not requiring reference to *any* external information for the meaning of all parts of a record to be clear to future users even when *all* supporting information, such as this document, is lost; and
5. To be easily translated to and from both fixed-field-formats and formally structured formats, including xml, Excel™, relational databases, and other media now in use or likely to evolve.

To a lesser degree, the VTF tool also is designed to minimize redundancy and duplication within each file—that is, to express a given piece of information in exactly one location. Because of widely reported difficulty with parsing multiple-record files, we also limit VTF records to a single component of data per file. Finally, the notion of separate text, real, and integer header sections is supplanted by a universal form of data-type identifier, allowing either alphabetical or logical arrangements of the information for ease of use and for making evident the relationships between tags. The notions of data types remains, but only as guidance incorporated directly in the name of each variable (in the formal IT sense, *all* VTF data are text, including the text representations of numbers).

To accomplish these goals, we have chosen a simple tagged format exemplified by ***Descriptive.Tag_type* = value units;**. Within this tagged format we also implement a modern "structured" or "class" design that is easily understood by a reader but readily translated by computers to and from other structured forms. Structuring the tag names also makes relationships between them more evident. Thus, we have made every tag name, its value, and its units of measure explicitly descriptive (and have provided for extensive comments). This design is the essence of dual human- and computer-readability, as well as of extensibility and autonomy.

An example of one minor set of entries for a VTF file is:

```
DataSeries.Sa(1).Period_dbl = 0.3 s;                 || Period
DataSeries.Sa(1).StructureDamping_dbl = 0.05;        || Damping, 0-1
DataSeries.Sa(1).Value_dbl = 32.0 cm/s/s;            || Sa value
```

This example defines the response spectral acceleration (32 cm/s^2) at a period of 0.3 s for 5 percent structure damping (0.05 fractional damping). The comments at the right or in the numerous ***Comments*** tags are anything the users believe will be helpful to themselves or others. The index ***(1)*** illustrates that arrays of tags—arrays of any length—can be created.

Three departures from Blue-Book-style formats are:

1. There is no longer either a fixed number or any concatenation of real, integer, or text variables (in order to allow greater efficiency and extensibility as well as more convenient or logical organization);
2. There are no coded entries, that is, no look-up tables requiring the survival of those tables for the entry to be meaningful. Instead, all table values are clearly understandable words, phrases, or values (±units) whose meaning will be the same in thirty years as they are today without reference to the tables from which they come; and
3. We have corrected the few typographical errors and omissions in the v1.20 fixed-field format while the opportunity for complete, detailed descriptions also is greatly expanded.

Where look-up tables are used, in addition to the listed self-evident values most such tables are extensible. That is, VTF users may define new entries if they do not find a suitable value already listed, so long as it is terse and clear to all users. Similarly, users may invoke ***Private*** tags when no extant tag suffices. While it is highly desirable to use standardized values and tags, and we have attempted to be comprehensive, these facilities accommodate items we have not anticipated and provide a means for suggesting additions to the VTF, which will be reviewed and acted upon.

Utility

Applications of the VTF include providing a generalized storage and translation medium for use by the VDC to simplify and expedite translations between all sources and users of strong-motion data internationally. The VTF tool is available for download and use by any interested party and is complemented by a series of portable software applications designed to make implementation straightforward.

Because we have kept the essence of the COSMOS v1.20 fixed-field format in the VTF tool, all of the Blue Book data dictionary is reproduced here, though sometimes rearranged or generalized. Thus, it is straightforward to translate unambiguously between VTF and the v1.20 fixed-field format and, by extension, all other formats.

The support tools currently available from the VDC are self-updating Java™ routines for access, display, and input/output; these subroutines can become the front and back ends of existing processing systems as desired. Initially, we are supporting Fortran, C, MatLab™, and Excel™, which are used widely by these communities. We also ask the strong-motion engineering and seismology community for consensus on other software worthy of direct support. By design, the translation routines are easily extended to other formats, thus are the basis of a universal translator at the VDC.

Appendix

Attached at the following page is the current governing document for the VTF. Although this document is detailed and in some ways difficult to read, it is the sole formal definition of the format and is kept up-to-date and distributed only by the VDC and its assigns. The most current version can be downloaded at http://db.cosmos-eq.org/FormatConversion.html if they postdate this Open-file Report.

The COSMOS Virtual Data Center Tagged-Format Tool: Formal Definition [Version VTF.1.0]

(16 October 2008)

John R. Evans[1], Melinda Squibb[2], Christopher D. Stephens[1], W.U. Savage[1],

Hamid Haddadi[3], Charles A. Kircher[4], and Mahmoud M. Hachem[5]

[1]U.S. Geological Survey, Menlo Park, Calif.

[2]SOPAC, University of California, San Diego, La Jolla, Calif.

[3]California Geological Survey, Office of Strong Motion Studies, Sacramento, Calif.

[4]Kircher and Associates, Palo Alto, Calif.

[5]Wiss, Janney, Elstner Assoc. Inc., Emeryville, Calif.

(There is a Table of Contents at the front and an Index of Tags at the end of this document. This is the sole, formal, governing document for this format, and as such is authoritative but difficult reading. The Executive Summary and other supporting items are supplied only for the user's convenience.)

Table of Contents

Introductory Material

Overview

The COSMOS v1.20 fixed-field format was an effort to unite in a single forum the various derivatives of the CalTech "Blue Book" formats being used to store and distribute strong-motion data. Issues of readability, generality, comprehensiveness, structure, and autonomy motivated revisiting the format issue in depth, resulting in this proposed COSMOS VDC Tagged Format (VTF) and its equivalent xml mirror as additional tools for the VDC (Virtual Data Center) and others who may find these tools useful. Efforts are underway to revise the fixed-field format, as well.

Primarily, the VTF tool is designed to be readily and unambiguously readable throughout by both humans and computers, to be expressly international in its viability, to be inherently both backward compatible and extensible with little effort (thus, easily maintained), to be inherently self documenting and autonomous (that is, not requiring reference to any external information for the meaning of all parts to be clear in the future even when all supporting information, including this document, are lost), and to be translated easily to and from structured formats, including xml, Excel™, databases, and other common media. To a lesser degree, this format tool also is designed to minimize redundancy and duplication in each file (to express a given piece of information in one location only, feasible because of its dual human and computer readability). The notion on text, real, and integer header sections is supplanted by a universal form of header information.

The tagged format's applications include providing a generalized storage and translation medium for the COSMOS VDC, a medium that can absorb and translate data from all sources internationally. Thus, the format is intended to be a superset of all formats now used in strong-motion work and to anticipate likely future needs. The tagged format also is available for use by any interested party, along with a series of tools designed to make that use straightforward.

By "backward compatible" we mean that parsing (reading) software for VTF Version VTF.1.1 and beyond will be able to read earlier VTF versions without difficulty, and this functionality will persist as a natural part of the format definition.

To accomplish the goals enumerated above, we have chosen a simple "tagged" format (and within this tagged format a modern "structured" or "class" design that will be both user friendly and still readily translatable by computers to and from other formats).

We have made every tag name (and its value) explicitly descriptive and have added extensive comments in this governing document about each tag's purpose and rules of use. Indeed, most of these tag names should look familiar to those who know the Blue Book formats and, where not so, we have commented about what equivalent variables our tags replace.

This design is the essence of dual human- and computer-readability. Everything expressed in VTF is made easy to read, understand, and bring into analysis systems.

Two departures from the Blue Book formats are, first, that there is no longer a fixed number of real or integer variables; this allows both greater efficiency and extensibility. Second, there are no longer any "coded" entries—no references to tables of entry definitions that must survive into posterity. Where there are tables of values that can be entered, all the values in those tables are clearly understandable words, phrases, or values whose meanings will remain the same and will not require reference to the tables from which they come.

In all instances we have tried to keep the essence of the v1.20 fixed-field format alive and well in the VTF tool. Although rearranged and recast into a more parseable form, all of the Blue Book data dictionary is reproduced here in some form. It is, therefore, straightforward to translate unambiguously between VTF and any of these Blue-Book formats. The VDC will be providing translation matrices and Java translation facilities, as well as subroutines for parsing and for writing VTF; these subroutines can become the front and back ends of existing ***Processing*** systems if desired. We will provide these front- and back-end routines in Fortran, C, MatLab™, and Excel™, and will ask the COSMOS community for consensus on other widely used software. The general translation routine will be in (automatically selfupdating) Java™ and driven by Excel™ tables describing the various formats. Thus, the translation routine is easily extended to additional formats and we anticipate supporting all widely used formats. Similarly, VTF can be used as the core of a "universal translator" connecting any of the formats described in those Excel™ tables. This translator and the Java™ interface to VTF will be key tools for implementing, refining, and extending COSMOS VDC functionality and user friendliness.

Finally, we have added items not available in v1.20 of the fixed-field format, such as various new types of instruments and manufacturers from Japan and Germany, Laplace, and z-plane descriptions of instrument and filter responses and to correct a number of trivial typographical errors.

The version of this document released by the COSMOS VDC is the sole formal definition of the COSMOS VDC Tagged Format Tool (VTF). Therefore, no change to the format or its use-rules from any other source is permitted. This document is long and detailed; a more easily understood executive summary and access tools for the format are available from the COSMOS VDC.

Formal Definition of the Cosmos Virtual Data Center Tagged Format

1. **Exactly one** time series or spectrum is allowed per VTF file, except in the case of response spectra, for which multiple damping values may be represented in a single file, all derived together from a single time series. Multiple channels are not to be concatenated.
2. In each VTF file:
   a. Two mandatory header lines begin the file, giving the VDC Tagged Format version name and text-encoding scheme (for example, "US-ASCII"),
   b. Next follow a variable number of other "header" lines,
   c. Optionally, a URL (Web address) pointing to an alternate source for this/these ***DataSeries*** (time series or spectrum),
   d. Optionally, a data-series checksum, and
   e. A ***DataSeries*** listing matrix containing the primary data as described by the following three items, f. through h.:
   f. A data-initiator line, "***DataSeries.DataSeriesValues_txt*** = {",
   g. A ***DataSeries*** listing matrix, each row (that is, line) containing, if given explicitly, an abscissa value (time, frequency, or period) followed by: (i) one time-series or amplitude- or power-spectral value (real or complex, hence one or two columns); or (ii) response spectral value(s) (real) for one or more damping values, such that columns 2 and beyond of the listing matrix form vectors of one response spectral period with varying damping;
   h. Lastly, a data-terminator line, "};", closing the ***DataSeries*** listing matrix,
3. All tag-value pairs shall be of the following format:

Formal Syntax

```
<TagName>_<VariableType> = <value>[ <units>];[[<white space>]|| <comments>]
     or
[<white space>]|| <comments>
```

where **TagName** is defined by

```
<TagName> = <ClassName>[(m)][.<SubClassName>[(n)][.<SubSubClassName>[(o)]...]]<MemberName>
```

As is common in software documentation, "**<>**" delineates a description of what belongs there while "**[]**" delineates optional or occasional elements, possibly nested. Note that **<TagName>_<VariableType>** (as well as any text values taken from the tables below) **are case sensitive**. Tag values are usually in English; tag names are always in English.

Similarly, neither tag names nor text values may contain any of the following characters: tabs, ", ' (double and single quotes), and ` (grave accent). Such characters can be misunderstood by xml and databases.

Furthermore, while tag names shall not contain white space characters, there shall be white space on both sides of the equal sign (" **=** ") and between the **<value>** and any **<units>**. White space may be blanks or tabs. Finally, note the presence of the semicolon after either the **<units>** or a unitless **<value>**—that is, the semicolon terminates the "***tag* = value [units]**" sequence and must be present.

White space also is allowed almost anywhere else, but not within tag names nor within numerical values (you can put white space inside text that you create yourself, but you must not inject it into text values taken from tables—use those exactly as they appear in the tables).

Numerical values may appear in integer, floating point, or scientific notation, but must be in base ten. (To avoid decimal-binary-decimal translations, thus preserving accuracy, numerical values are treated as text by most VTF utilities until such time that they must be operated upon as numbers, such as for plotting the ***DataSeries***.)

In most cases tags can be given in any order, however, it is recommended that either an alphabetic order or the functionally organized order shown here generally be respected so that tags can be located easily. The ***Index of Tags*** near the end of this document is a comprehensive alphabetical list that may aid users.

(There are three exceptions to the flexible-order rule, one at the top and two at the bottom of the file. (1) The first tag in table 3A must appear first in the file, (2) the tags surrounding the ***DataSeries*** are in a specific order, and (3) the ***DataSeries*** listing matrix, when the abscissa is implicit, must be in increasing order of the abscissa's value. We encourage the same practice even when the abscissa is given explicitly as the first (or first two) columns of the ***DataSeries*** listing matrix.

As implied above, there are instances in which the abscissa of a time series or spectrum must be given explicitly as the first one (real) or two (complex) columns of the ***DataSeries*** listing matrix. However, there are many common situations in which is it sufficient to define the abscissa implicitly, as with regularly sampled time series and Discrete Fourier Transform (DFT) spectra. Many other spectral forms will require an explicit spectrum. The rule is that if an abscissa can be correctly defined by a starting point and an increment, it may be (but does not have to be) given explicitly. All other cases must provide an explicit abscissa. Instances of requisite explicit abscissas commonly include response spectra and power spectra computed from a DFT that is spaced evenly in frequency but where the power spectrum is provided as a function of period.

Authority

The only authoritative definition of the COSMOS VDC Tagged Format (VTF) is this document, as maintained by the COSMOS Virtual Data Center or its assign.

Nevertheless, the VTF has many "user-extensible lists" (tables of values) for which a user may assign any reasonable value, as long as it starts with the string **"User's description: "** (to help us make format-verification work well in the most general case). Such user-created values in extensible lists are likely to help us expand this authoritative VDC document. Similarly, use of ***Private*** tag sets, defined in table 3Q, also may inform our modifications of the VTF.

Private tag sets and user-created text values should be limited to necessity and kept reasonable and likely to be understandable to any user at any future time.

Language

The official language of all tags (*including* ***Private*** tags) and of most text values is English. However, we recognize that certain text values, such as the descriptive name of a geographic location (***GeoLocation.Name.Description***) and addresses (***GeoLocation.Name.Address***), sometimes must be in their local languages to be meaningful. Furthermore, translation of comments and other errata into English should not be allowed to become a barrier to the dissemination of data.

A Note on Fourier Spectra

For a DFT, if only the non-negative frequencies for a real time series are to be given in this VTF file, then the power in the negative-frequency bins generally must be added to their corresponding positive-frequency bins so that the positive-frequency bins represent the total power at each frequency (generating a true one-sided spectrum). For real time series, this correction commonly can be accomplished by scaling up the non-Nyquist positive-frequency bins by the square root of two, however, caution is appropriate.

(Many FFT implementations of the DFT yield their output in some order or another, such that it contains negative-frequency bins, one 0-Hz ("DC") bin, positive-frequency bins, and one Nyquist bin. Typically then, half the power of non-Nyquist finite-frequency bins is to be found in the negative-frequency bins. Thus all non-Nyquist positive-frequency bins are likely to be too small by a factor of the square root of two. In contrast, most of these FFT routines put all the power of the 0-Hz and Nyquist bins each into a single bin, so that these two bins do not need correction.)

Since many FFT routines differ in their output formats and rules, it is essential that the user test the behavior of their FFT routine with real data to determine whether this correction for power lost to negative-frequency bins is required as well as to determine the particular behavior of the 0–Hz and Nyquist bins.

To test an FFT routine, take the FFT of one or more pure sine waves fitting exactly into the input window and using no weighting in that window (that is, a boxcar window). "Fitting exactly" means that the time-series point just after the end of the input window and, therefore, not present in the data provided to the FFT, must exactly equal the first point in the input window and be in the same phase. That is, that the window duration is one sample shorter than a full integer

number of cycles. This arrangement is called "DFT symmetry"—because the DFT assumes a precisely periodic input, repeating the first point just after the last and so forth, DFT symmetry yields an effectively infinite-duration pure sine wave which will put all its power into exactly one frequency bin or exactly two complementary-frequency bins, positive and negative—there will be no side lobes. The total power contained in either the one bin or the positive-negative pair of bins will equal the power computed in the time domain. If a pair of bins is required to match the total power in the time domain, then the correction to non-Nyquist positive-frequency bins is required when providing only the non-negative bins in the VTF file. Similarly, it is possible to verify whether all or half the power of the 0-Hz or Nyquist bins is in one bin.

Tag Arrays

The VTF allows ***Subscript***ed arrays of variables (that is, arrays of tags). This facility allows for things like enumerations of filter poles and zeros, several earthquakes appearing in a single record (for example, rapid-fire aftershocks or swarm earthquakes), and historical information about an instrument.

These "tag arrays" are implemented by the "***(m)***" part that is shown at the ends of class or subclass names as indicated in this document—in other words, ***Subscript***s are not allowed for a tag or portion of a tag unless shown in this document. ***Subscript***s must be positive integers (1, 2, 3, …) with no missing values, so they are in the MatLab™ or Fortran style. ***Subscript***s are not in the C style of non-negative ***Subscript***ing (0, 1, 2, …).

For example if a user wants to declare several possible versions of an ***Event*** location those tags would be ***Event.Hypocenter(n)***. In contrast, locations for multiple ***Event***s appearing within a single record would be rendered as ***Event(m)***. However, it would not make any sense to write ***Event.Hypocenter.Agency(n)***, which would imply one ***Event*** location with several authors, so this ***Subscript***ing is not permitted.

In another example, a user can document multiple instances of names for a spot on the Earth or in a structure (a ***GeoLocation***)—a history of changing names or differing names used by different ***Agencies*** for the same instrument. A user might, for example, want to document some name multiplicity with ***GeoLocation.Name(1).ShortName***, ***GeoLocation.Name(2).ShortName***, and so forth, in cases where the ***Sensor*** never moved or moved only slightly in the past and is functionally at the same location. ***GeoLocation(n)*** makes no sense since in the above example as it would imply a ***Sensor*** is in several places at once; this usage is disallowed.

Classes and subclasses end with periods while class members end with an underscore ("_"). The "***(m)***" is *just before that period or underscore character* and just after the class or member name, as in "***Sa(2).***" or "***ResponseSpectrumDamping(3)***".

To limit ambiguity, users are urged to keep these ***Subscript***s time-, frequency- or period-sequential where those concepts are meaningful. Note the prevalence of explicit time stamps in this format, which also facilitate documenting historical associations.

Our ***Subscript***ing guidance is either "***[(n)]***", meaning "optional ***Subscript*** here", or "***(n)***" meaning "required ***Subscript*** here". As mentioned, ***Subscript***s may not appear anywhere else. If a tag allows ***Subscript***s but none are given, it defaults to an implicit ***Subscript*** of "***(1)***"—note that only the last of repeated un***Subscript***ed tags will be saved.

Primary Classes

Both the physical grouping of tags and their **<[Sub]ClassName>** and **<MemberName>** are attempts to associate variables in a logical manner. For example, variables having to do with the Data Acquisition Units (***DAU***s, which are recorders and data loggers) are named by the ***DAU*** class; tags associated with the geographic locations ("station names", loosely speaking) are ***GeoLocation.Name***; and tags associated with the source event are ***Event***. The following table includes a complete list of the primary ("top") set of Classes and their definitions.

Table of Primary Tag Classes and Definitions of Terms

ClassName	Meaning
ThisFile	A small set of special tags which apply to this VTF file, such as those naming the version of the format and the VDC's unique identifier of this record.
Array	Any set of ***Sensor***s plus ***DAU***s at one or more ***GeoLocations*** that is managed as or logically forms a single data-acquisition system—for example, a structural ***Array***, a geotechnical ***Array***, or a single free-field instrument at a single ***GeoLocation***. (An ***Array*** with a single-***GeoLocation*** is the usual case in the free field, but it is still an ***Array*** in the same sense that a scalar is a one-element vector). Note that arrangements many of us think of as and call "an array" (such as the NESMP "array"), are not ***Array***s as narrowly defined here. In the VTF, such agglomerations are more properly called an ***Agency***.†
GeoLocation	The exact geographic location in three dimensions of the point of measurement (strictly speaking, of the ***Sensor***'s proof mass). *Note that this is a very narrow definition* and defines the location of a single ***Sensor*** or package of several axes. Even some classical simple "stations" with ***Sensor***s spaced a few meters apart might be considered by some users to occupy several distinct ***GeoLocations***, for example, in cases of precise GPS location and large ground-motion spatial variance. We strongly discourage the practice of giving the same set of coordinates for all members of an ***Array***, such as the ***Array*** monitoring a dam site. This practice obviates many applications of the data.†
GeoLocationRupture	Tags relating ***GeoLocation***s to fault-rupture surfaces.
Event	The signal of interest (for example, the natural or artificial source of ground motion, a calibration signal, and so forth).
EventGeoLocation	Tags relating ***Event***s to the ***GeoLocation***s of ***Sensor***s.

DAU	Data Acquisition Unit—the recorder or data logger. Typically a preamplifier, analog-to-digital converter, and some recording and/or telemetry medium, when taken in combination. Formerly, often a film recorder.
Sensor	The transducer that converts the Earth or structural motion to an electronic (or sometimes optical) signal. Generally, the accelerometer.
DataSeries	The single, delimited, time-, frequency-, or period-series or group of response spectra, as referenced by or included in this VTF file—that is, the primary data of this file. Also, any other values closely associated with, or generated from, these primary data (the start time of the time series, its RMS or maximum value, its ***Sa*** or Housner Intensity, the data format and number of points, and so on).
RawSeries	Values closely associated with, or generated from, the time- or frequency-series from which this processed ***DataSeries*** originated—the immediate predecessor to this VTF file.
Processing	Parameters describing the record-***Processing*** steps applied to the ***RawSeries*** or original time series to obtain the ***DataSeries*** in this file.
Private	***Private*** variables for use *only when* there is not yet any appropriate VTF tag. Search the ***Index of Tags*** carefully before defining a ***Private*** tag.

†The reader may notice that we rarely use the word "station", instead nearly always using either the word ***Array*** or the word ***GeoLocation*** as they are narrowly defined above. We do so because the word "station" means many different things to different practitioners, in some cases signifying an entire ***Array***. Similarly, we limit our use of the word "***Site***" to mean the site conditions at a particular ***GeoLocation***; "site" has other special meanings to large portions of the COSMOS community, as in "work site", which is broader then our definition (notwithstanding that many ***GeoLocation.Site*** data are generalized for an entire work site, often for reasons of cost, and not as specific to a particular ***GeoLocation*** as one might desire.)

In tags identifying entities responsible for various parts of this VTF file (such as the owner/operator of the instrument in question) we use the word ***"Agency"***. By selecting this word, we do not mean to limit in any way the type of entity (organization or individual) that may be named in such variables. The term ***Agency*** is nothing more than a reasonably general **<MemberName>** with a meaning tolerably close to its intended use. We expect that many an "***Agency***" will be an individual person (such as amateurs of the Public Seismic Network), a secondary-school classroom, a Government agency, a consortium such as IRIS, or a college or university (this name is analogous to a GVDC "Business Associate").

In all cases, we intend that preference be given to identifying ***Agencies*** by FDSN network names, some of which are shown in the second column of table 3E; additional names that may be used are given in the first column of that table. If no appropriate ***Agency*** name is available from these lists, the user may insert some terse, well known, identifiable abbreviation for an organization's name or insert the individual's name. That is, the ***Agency*** list is "user extensible"—it is unlikely that the we have been comprehensive and we do not wish to exclude any ***Agency*** from the VTF.

Tag-Value Example

To illustrate this format, the following lines represent typical spectral acceleration values at the two periods, 0.3 and 1.0 s, for 5 percent damping:

```
DataSeries.Sa(1).Period_dbl = 0.3 s;            || Period
DataSeries.Sa(1).StructureDamping_dbl = 0.05;   || Damping fraction
DataSeries.Sa(1).Value_dbl = 32.0 cm/s/s;       || Sa value
DataSeries.Sa(2).Period_dbl = 1.0 s;            || Period
DataSeries.Sa(2).StructureDamping_dbl = 0.05;   || Damping fraction
DataSeries.Sa(2).Value_dbl = 50.0 cm/s/s;       || Sa value
```

A Note on Comments

There are two kinds of comments in the VTF: (1) comments associated with a particular group of tags (any ***Comments***, ***Description***, ***Annotations***, or ***Citation*** tags) and (2) comments that can be put almost anywhere, but which are given limited treatment by the parser (for example, "ll This value needs updating.").

For *all important comments, please give preference* to the group of tags ending with ***Comments***, particularly those that you wish to associate rigorously with a particular subject matter. Although "ll" comments are brought forward by the parser, they are associated only loosely with their original subject (by position in the VTF file). For example, in the xml mirror of the VTF "ll" comments are added to the item at which they are given as a "***Note***" xml type. That is, the parser simply attaches all "ll" comments to the nearest tag above the "ll" comment). Thus, "ll" comments are primarily for human consumption and not computer manipulation and should be avoided.

Nevertheless, "ll" comments are used extensively in this governing document and other supporting materials to elucidate details of usage—these comments generally should not be included in any working file.

In practice, "ll" comments might be used to provide *temporary* notes by the user to the user (for example, "ll still need to fix this") or to make minor comments that do not need to be associated rigorously with any particular tag. However, "ll" *comments should not be used* to make comments that are important in any significant way to the data.

Note that this is a change from the similar looking comments of the Blue Book formats. Important comments should be put into the variable ***ThisFile.Annotations.TextValue*** if not elsewhere associated with specific portions of the header in one of the other ***Comments*** (or ***Description***, ***Annotations***, or ***Citation*** tags) variables.

(The xml class ***Note*** is not visible to the user. It is used by parsers to store "ll" comments in proximal association with the tags to which they are appended. All such comments are attached to the nearest previously encountered tag.)

Additional Matters of Format

Blank lines and lines consisting entirely of comments may appear anywhere after the first line excepting within the ***DataSeries.DataSeriesValues_txt*** = {…}; sequence of lines. Studies have shown that simply inserting blank lines in appropriate locations, such as between major header sections and between logically associated blocks of information, greatly improves the human readability of files. We commend such uses of white space to the reader and make extensive use of it in this governing document.

There is no firm limit on line length since the COSMOS community is no longer bound by the rigors of punched cards. Nevertheless, for the sake of keeping files easy to read on modern workstation screens, *users are urged to keep line lengths below 100 characters where feasible*. This length easily fits onto screens possessed of moderate resolution at reasonable character size without imposing unreasonable limits on the sizes of tags or their values. (This document uses about a 100-character line length, for example.)

The primary exception to this voluntary constraint is for ***Comments***, ***Description***, ***Annotations***, and ***Citation*** tags (the preferred sorts of comments), which users may wish to make long in many cases. A word-based wrap-around display rule in your editor of choice should be sufficient to ensure readability of such text, as long as users are reasonably careful to write with adequate white space inside the comment text.

In particular, ***please replace line break characters*** with the four-character sequence "***\n***" to ease the reader's job. (The "***\n***" is the C-language symbol for "newline".)

In any case, lines *shall not be blank padded* on the right. (All lines will be read in as strings by the parsers, so blank padding simply wastes file space.)

File Naming

We suggest, but do not require, a format such as the following for naming VTF files:

```
<Date and time>[_<Network Code>]_<GeoLocation Name, possibly full SNCL>[_<Processing
instance, stage, volume, etc.>][_Ch<Channel>[In<Index>]][_<Etc.>]_<[A|V|D|…]>.COSM
```

The COSMOS VDC will use such filenames in data it translates into VTF for a user.

For example, "**20021103_221245_XX_PS09_Op1_Vo2_Ch2_V.COSM**" is a record from the Denali fault earthquake of 03 November 2002 ("20021103_221245") for Alyeska ("_XX" == "other network") station [***GeoLocation***] "PS09", evaluated with "***Processing*** option/instance" one ("_Op1"), fully processed ("Vo2" == Volume 2), for Channel 2 ("_Ch2"), and is a velocity trace ("_V"). The "COSM" (or "cosm") means "a COSMos vdc tagged format".

The ***<Date and time>*** stamp should be in largest-to-smallest order and generally should be to the nearest second; we suggest the format YYYYMMDD_hhmmss, as in "20051117_173802", which is a variant of the ISO time format we use here.

Note that this ***<Date and time>*** should be the best available time, preferably the time of the first sample. If that instant is not known, use the trigger time of that record. If even that is not known, use the origin time of the ***Event***. (This ordered list of options is for the sake of consistency between records and to take account of legacy data, which often have very poorly known timing.)

The **<Network Code>** is from either ***Array.Agency*** or the "NN" field in ***GeoLocation.Name.SNCL*** or ***GeoLocation.Name.ExpandedSNCL***.

The **<GeoLocation Name, possibly full SNCL>** is whichever of ***GeoLocation.Name.SNCL***, ***GeoLocation.Name.ExpandedSNCL***, or ***GeoLocation.Name.ShortName*** is used to identify the ***GeoLocation*** of this ***Sensor***, with a SNCL name preferred if several are given. Obviously, use the ***GeoLocation.Name*** that is in force at the time of the ***Event*** and as used by the ***Agency*** providing this record.

The portion **"[_Ch<Channel>[In<Index>]]"** comes from the tags ***Sensor.ArrayChannel*** or ***Sensor.DAUchannel*** (giving preference to the former) and from ***GeoLocation.Name.Index***.

In any case, the combination "**_[<Network Code>]_<** ***GeoLocation*** **Name> … [_Ch<Channel>[In<Index>]]**" should be as detailed as is required to identify uniquely the signals from a specific proof mass (both within that particular ***Array*** and among all your international colleagues). That is why a full SNCL name is preferred in place of the entire sequence of strings—the FDSN designed SNCL network names to be unique worldwide.

The portion "**_<[A|V|D|…]>**" should always be present and is a string of from one to three letters indicating the type of trace, "_A" for acceleration, "_V" for velocity, "_D" for displacement, and so forth as shown in column two of table 3B(b).

The optional "***[_<Processing instance, stage, volume, etc.>]***" identifies the stage of ***Processing***, for example "_Vo2" for the equivalent of a Blue Book "Volume 2" record. This portion of the VTF filename should correspond to the meaning of one or more of the following five tags from table 3N as shown here:

Tag and value	Filename fragment
Processing.BlueBookVolume_int = [0-3];	"_Vo[0|1|2|3]"
Processing.Stage_txt = ["Raw" | "Preliminary" | "Final"];	"_St[R|P|F]"
Processing.HumanReview_txt = ["None" | "Reviewed" | "Updated"];	"_Hn[N|R|M]"
Processing.Instance_int = <non-negative value>;	"_Op<Value>"
Processing.SpecialProcessing_txt = "<Value>";	"_SP<Value>"

These substrings should be concatenated as needed, as in the example above of "_Op1_Vo2".

Finally, some users may desire additional differentiating codes, the "**[_<Etc.>]**".

When filenames begin in this manner with a ***<Date and time>***, followed by a string that uniquely identifies the ***DataSeries***, and end with a string that uniquely identifies the data-***Processing*** stage, the resulting files generally will sort easily into ***Event***s with the help of typical system file-name-sorting and file-listing algorithms. It also will be clear to colleagues what the name means. Therefore, the portions of the filename after the ***<Date and time>*** should as specific as is required for the particular ***Array*** and ***Agency***. Ideally, a filename should include a full SNCL name and strings identifying the type of ***DataSeries*** and data-***Processing*** leading to the file. As long as an FDSN network code (***Agency***) appears in the name, there should be no naming conflicts with other ***Array***s.

Another example may make this convention clearer for the case of a SNCL name:

```
20041115_093415_CH4AW.NC.HNZ.01_V2_A.COSM
```

identifies a "Volume 2" vertical acceleration ("HNZ") record from a temporary USGS instrument co-located with the CSMIP instrument "Cholame 4AW" near Parkfield. The corresponding CSMIP record might be named

```
20041115_093414_36412.CE.HNZ.01_V2_A.COSM
```

in this case using the CSMIP five-digit name for the same ***GeoLocation***. There can never be confusion of a differing ***DataSeries*** because of the different FDSN network name "CE".

Required and Lesser Tags

VTF tags come in four flavors: ***REQUIRED***, ***CONDITIONAL****ly required*, ***Recommended***, *and* ***Optional***. On rare occasions, you also will encounter ***Avoid***, which include ***Optional*** tags that are not recommended for use but are kept for backward compatibility and similar reasons.

All ***REQUIRED*** tags *must* be present in every VTF file, even when there is no value corresponding to the tag. ***REQUIRED*** tags with no values must be specified as NULL (*without quotes* for all variable types, including for text strings)—see discussion and table below. Of course, ***Processing*** software may not be able to function if required tags have NULL values (which are translated into a NaN (IEEE "not a number" value) or into an empty string, depending on tag type), so such values partly serve as reminders to the author that they still have values that probably should be filled in before the file is shipped. Nevertheless, if a required tag is simply unavailable, ship the file and let the users make due, "some data" generally being preferable to "no data".

Conditionally required (***CONDITIONAL***) tags are required in certain contexts that are explained in the comments associated with each ***CONDITIONAL*** tag. For example, when poles and zeros are used, all associated tags are then ***REQUIRED***, including that the number of poles and zeros become required and each of the poles and zeros themselves are also required.

Recommended tags are those which form a thoroughly documented record, but which may not be available for all data, particularly for legacy data, and are not essential to that record's use.

Optional tags are those which generally should be included if available but are not ***REQUIRED***.

Whether a tag is ***REQUIRED***, ***CONDITIONAL***ly required, ***Recommended***, or ***Optional*** is indicated by the comment to the immediate right of or just below the tag definition. (Please remove these annotations from working VTF files.)

Finally, because of the comments specifying this degree of requirement or need sometimes wrap past the end of the line for some of the longer-named tags and for some tag values that are long lists of possibilities provided there rather than in another table, it is often necessary to break a line that defines a tag. Please do not break these lines in actual use.

Notation

As mentioned above, angle brackets, "<>", form a description of what should be substituted at that spot. Similarly, square brackets, "[]", delineate optional or occasional items.

In addition, a construct like "[a | b]" means "choose either a or b, *but* you must choose at least one of them". That is, you make a choice from a list of "options", not that having a value there at all is optional.

Users familiar with Unix manuals will find this pattern familiar, while others may find it curious. The "|" character is the Boolean (logical) "OR" operator in many computer languages, thus the notion of "one OR the other". Since the notation is terse and fairly widely used, we have adopted it here. In places where it is less than obvious in meaning, we try to make that meaning clearer through examples, as in the ugly formal definition of an ISO ***DateTime*** value.

Organization of this Document

This formal definition of the VTF is presented as (1) the primary format statement given above, and (2) a large set of tables below, which define and describe the tags and supporting information.

Table 1 enumerates the terse set of allowed variable types and their associated suffixes. (*Please* see the ON PRECISION section below.)

Table 2 lists all recognized physical units (we would be happy to add more as needed).

Tables 3A-3R are the bulk of the format description. These tables describe the numerous tags comprising the "header" as well as a number of lists of text strings that can be entered as the values of certain tags (for example, accelerometer-model names).

Table 4 describes the main ***DataSeries*** listing matrix (the time series or spectra). Note that some of the supporting tags in the class ***DataSeries*** are in tables 3A-3R.

ISO Date-and-Time Format for *DateTime* Tags

When a date ± time variable is required, we use the "ISO 8601:2004" ***DateTime*** standard format, with a few (widely used) exceptions and permitting any degree of right-side truncation to obtain lowered precision. Please do tell the user whatever you know, even when those data do not have much precision or are incompletely known.

Our modified ISO format is

```
YYYY-MM-DD hh:mm:ss.sss[Z|[+|-]hh[:mm]]
```

As mentioned, you may use this format to any level of precision, including by using any number of fractional digits in "ss.sss" (including just "ss", the nearest second), down to the least precise ***DateTime***, which would contain only "YYYY" (note that YYYY must always be a four-digit year in the Gregorian calendar, Common Era).

If you do not provide an explicit ***TimeError*** for a given ***DateTime***, users are likely to infer that the actual uncertainty is implied by the degree of precision given in the ***DateTime*** value. In a particularly clear example, ***DateTime***s given as only "YYYY-MM-DD" may reasonably be assumed to have occurred sometime during that (generally UTC) day, unless otherwise stated. (Indeed, many ***DateTime*** variables only require accuracy to one day.) In any case, this implication is

often intentional on the part of the data provider—just be aware of the likely implications of a truncated ***DateTime***, absent a ***TimeError***.

The primary distinction of the VTF from the ISO standard is replacing ISO Standard's "T" between the date and time with a blank so that it reads "…-DD hh:…" rather than "…-DDThh:…", thus is a little easier on the eye. This alteration also is widely used.

(For descriptions of ISO 8601:2004, see http://www.cl.cam.ac.uk/~mgk25/iso-time.html, or the official source, http://www.iso.org/iso/en/prods-services/popstds/datesandtime.html.)

Note that the end, "[Z|[+|-]hh[:mm]]" reduces to "Z" for UTC time ("Zero" or "Zulu" time). The "[+|-]hh[:mm]" part gives a time offset from UTC to any local time zone in hours and possibly minutes (a few time zones are 15 or 30 minutes off the hour). For example, "[Z|[+|-]hh[:mm]]" is "–08" (eight hours behind UTC) for Pacific Standard Time in the USA, and "+05:30" (five and a half hours ahead of UTC) for India Standard Time.

Examples: This description was finished at "2007-01-05 23:59:55Z" (rigorous ISO 8601:2004 would render that "2007-01-05T23:59:55Z"). If I only admit when this happened to the hour, to the day, to the month, or to the year, it would be "2007-01-05 23Z", "2007-01-05", "2007-01", or "2007". The latter three may also include the "Z" if desired. Absent specificity, the date are likely to be assumed to be on the local calendar (some legacy data) but more likely in UTC.

On Precision: This is Important!

The COSMOS formats (as well as Blue Book derived formats) are inherently text based and, therefore, are *not* referenced to any computer's internal precision, such as 64-bit IEEE floating point or 32-bit integers.

If you are creating your own VTF files, *it is strongly recommended* that you print floating point numbers with a sufficient number of digits to reflect the actual precision of your data, plus one or two more digits to limit round-off error (an old technique in the design of computers).

If you are reading VTF files, be certain to allocate binary space of adequate precision in your computer so that it preserves the values presented in text of the VTF file (thus, the label ***"_dbl"*** suggesting use of "double precision"). Because computation is now fast and inexpensive, we recommend using double-precision or better for all ***_dbl*** and ***_cpx*** values—there is almost nothing to loose and plenty to gain.

Also because the VTF file is text, the precision is only as good as presented in the VTF file, notwithstanding ***"_dbl"*** and other notations.

Null Values

The following special values shall be used both on input and output whenever a value is unknown.

Representation of NULL value	**Comment**	**Most often translates into**
<some tag>_txt = NULL;	Null text—NO QUOTES	Empty string
<some tag>_int = NULL [units];	Null integer	***ThisFile.NullIntValue*** or the largest allowed negative value on that computer
<some tag>_dbl = NULL [units];	Null double precision	NaN‡
<some tag>_cpx = NULL NULL [units]; or <some tag>_cpx = NULL [units];	Null complex double	NaN NaN‡

‡ When read into computational software, a NULL value generally will cause the parser to return a NaN (IEEE-float value "not a number") or an empty string. But in the case of integers, or on systems that do not support NaN, the values returned generally will be the largest possible (finite) negative values for that computer.

If provided in the VTF file, the three values ***ThisFile.NullComplexValue***, ***ThisFile.NullRealValue***, or ***ThisFile.NullIntValue*** may be returned instead, as appropriate to the computer, the application, and the wishes of the user. (All VTF parsers should include a user option for overriding these NULL value translations with other values, such as the user's own favorite "null" number, like 999. However, we strongly urge that the NULL values be a improbable value (so not 999) and that the user never use zero as a NULL value since zero is far too often a real, intended data value, not at all NULL. A maximal negative is unlikely

in nearly all situations.) Finally, note that "maximal negative" does not mean negative infinity (which can be portrayed in IEEE floats)—this value should be reserved for its rare but valid uses.

When any VTF file is created that DOES NOT have available *all* of the REQUIRED tag values, then it is required that the "missing" tags be written in anyway and given the value NULL, as above. Also, these NULL values are used to fill out the "empty" header supplied at the COSMOS VDC Web site as an educational and quick-use tool. Finally, the NULL values may be used as place holders for anticipated data to be supplied later.

THE COSMOS VDC TAGGED FORMAT TABLES (VTF.1.0)

Table 1. Variable-Types Suffix List

Suffix	Storage type
_txt	Text strings, always enclosed within double quotes (""). Use no blank padding at the ends; text justification is never needed.
_int	Integer values (32-bit). Preferably two's complement (thus in the range ±2,147,483,647, that is, $\pm(2^{31}-1)$.
_dbl	Double-precision (64-bit) floating point, preferably IEEE.
_cpx	If a complex value is required, **two *_dbl* values** are given, real then imaginary, separated by white space, unless the value is real, in which case either **<real value>** or [**<real value> 0**] are both allowed and equivalent.

The significance of variable types is informational and suggestive (except as they imply particular formats for the VTF text values). The implications of these suffixes is that users should allocate comparable binary storage to accommodate the equivalent of the stated data type and precision.

There is no longer much reason to use single-precision floating-point or complex numbers, computation being fast and inexpensive, so we urge the use of storage by using the variable types implied by these suffixes.

Table 2. Recognized Units

SI units, or at least metric units, are preferred but not required, recognizing the engineering context of strong motion data and applications.

Name	**Unit**
Time	
years	Large time units used mainly to indicate ***TimeError***s (where not implied by rounding of the associated ***DateTime***s)
months	
days	
hours	
minutes	
s	seconds (***do not use "sec"***, which is not the SI abbreviation)
Hz	Hertz; inverse seconds (***do not use "sps"***)
ms	milliseconds
us	microseconds
mHz	millihertz (0.001 Hz)
Gravitational acceleration	
g_local	**Local** Earth acceleration at the ***Sensor GeoLocation****
mg_local	0.001 of local Earth acceleration
ug_local	0.000001 of local Earth acceleration
g_standard	**Standard** Earth acceleration (9.80665 m/s/s)
mg_standard	0.001 of standard Earth acceleration
ug_standard	0.000001 of standard Earth acceleration

Scaling	
cm/g_local	centimeters offset on film record per local g
cm/g_standard	centimeters offset on film record per standard g
mm/g_local	millimeters offset on film record per local g
mm/g_standard	millimeters offset on film record per standard g
in/g_local	inches offset on film record per local g
in/g_standard	inches offset on film record per standard g
Linear distances and rates	
km	kilometers
km/s	kilometers per second
mi	statute miles (1.609344 km)
mi/s	statute miles per second
m	meters
m/s	meters per second
m/s/s	meters per second per second
cm	centimeters
cm/s	centimeters per second
cm/s/s	centimeters per second per second (***do not use "gal"***)
ft	feet (30.48 cm)
ft/s	feet per second
ft/s/s	feet per second per second
in	inches (2.54 cm)
in/s	inches per second
in/s/s	inches per second per second
um	micrometers (***do not use "microns"*** **or "** **μ";** use "u" for ASCII compatibility)
mil	0.0001 inches, typically a machinist's unit
Area	
<distance unit>^2	squared <distance unit>, for example, km^2

Angles and angular rates		
deg	degrees (always decimal)	To be used for all azimuth and orientation values.
deg/s	degrees per second	
deg/s/s	degrees per second per second	
rad	radians (always decimal)	Mainly used as the units of rotational time series.
rad/s	radians per second	
rad/s/s	radians per second per second	
mrad	milliradians (always decimal)	
mrad/s	milliradians per second	
mrad/s/s	milliradians per second per second	
urad	microradians (always decimal)	
urad/s	microradians per second	
urad/s/s	microradians per second per second	
Miscellaneous		
counts or count	digital counts (the singular is used for the denominator of some compound unit name, as in Conversion Factors below	
	a simple number (unitless)	
%	percent	
%FS	percent of (single sided) full scale	
Voltage		
V	Volts	
mV	milliVolts	
uV	microVolts	
Conversion factors		
(<unit>)/(<unit>)	General conversion constant from the denominator's **<unit>** to the numerator's **<unit>.** The parentheses are required when a compound unit is used, for example, ***Sensor.Sensitivity = "(cm/s/s)/count";***	

Stress	
MPa	1,000,000 Pascal
Pa	Pascal (Newtons per square meter)
Baryes	dynes per square cm
bars	stress as bars (0.1 MPa); roughly atmospheres
psi	pounds per square inch (absolute)
psig	pounds per square inch (“gauge”—relative to atmospheric)
Energy	
joules	SI unit of energy
ergs	CGS unit of energy
Energy/moment	
Nm	Newton-meters
dyne-cm	dyne-centimeters
Strain	
ustrain	microstrain
Logarithmic relative power or amplitude	
dB	decibels
dB/octave	decibels per factor-of-two in frequency
dBm	decibels referenced to 1 mV
dBa	decibels referenced to 1 m/s/s
Spectra	
(<unit>)/Hz	Fourier amplitude spectrum, for example, (cm/s/s)/Hz
(<unit>)/(root Hz)	rms spectrum, for example, (cm/s/s)/(root Hz)
(<unit>)^2/Hz	power spectral density, for example, (cm/s/s)^2/Hz

* The Tag ***Array.EarthGravity*** defines the value of local Earth acceleration at the ***Array*** and the ***Sensor***, thus defining the unit “g_local”. (This value is very unlikely to change across an ***Array***, as we narrowly define it, to any degree important in strong-ground-motion applications. It certainly can vary across the network of one ***Agency***, generally in the parts per thousand.)

Table 3A. Certain General and Special Header Tags

```
ThisFile.Format_txt = "VTF.<major.minor";                                  || REQUIRED
            || Example:                                                    || 1st tag
            || ThisFile.Format_txt = "VTF.1.0";
            ||
            || This line MUST be the first in the file.

ThisFile.CharacterEncoding_txt = [ "US-ASCII" | "UTF-8" | "UTF-16" ];      || REQUIRED
            || Informs users and software which of the these character-     || 2nd tag
            || encoding standards is used in this VTF file.
            ||
            || Ideally, this tag should be second in the file to make
            || life easier for both parsers and humans. However, this
            || positioning is not required (the VTF file itself is quite
            || likely to contain a few invisible bytes at the beginning,
            || specifying its encoding standard for use by the computer's
            || operating system).
            ||
            || UTF-8 and -16 are, respectively, one- and two-byte encodings
            || of Unicode™ (ISO-10646). UTF-8 contains all 127 of the ASCII
            || characters, as well as most other widely used characters, this
            || in the same storage space as simple ASCII. However, UTF-8 is
            || incomplete in that it is missing Chinese and several other
            || major scripts. On the other hand, UTF-16 is complete, but
            || it uses about double the storage space of UTF-8.
            ||
            || Please use either UTF-8 or UTF-16 for all VTF files, giving
            || preference to UTF-8 where practical, and to UTF-16
            || where comprehensiveness is needed.
```

```
ThisFile.DataSeriesCOSMOSidentifier_txt = "<value>";                    || CONDITIONAL
            || REQUIRED if this file is served from the COSMOS VDC;
            || Optional if served directly from data owner (Agency).
            ||
            || The COSMOS VDC unique identifier that has been assigned
            || by the VDC and associated with this DataSeries. Providers
            || and users may both apply it subsequently for clarity, if
            || desired, but it must remain unchanged from the VDC's
            || assigned value.
            ||
            || Below, see ThisFile.DataSeriesHistory(n) for a means
            || to store a history of these unique identifiers and/or of
            || the filenames of precursor files.
            ||
            || In addition, DataSeries.AgencysIdentifier is used to
            || provide the provider's own identifiers for this record.


ThisFile.Preparation.Agency_txt = "<value>";                            || REQUIRED
            || Whatever individual, group, or Agency prepared this VTF
            || file; take this value from table 3E.
ThisFile.Preparation.DateTime_txt = "YYYY-MM-DD[ hh:mm:ss[Z|[+|-]hh[:mm]]]";   || REQUIRED
            || The date (± time) on which this COSMOS tagged-format (VTF)
            || file was created. It is not the creation DateTime of any
            || source file that may be translated into or subsumed by this
            || VTF file nor of any data copied into this record from other
            || sources (for example, earthquake-source data).
            ||
            || Instead, use Processing.DateTime to identify the
            || originating-Agency's authoritative creation DateTime
            || for the DataSeries in this file. Various metadata also
            || have DateTime tags to indicate their origins separately.
            ||
            || Since this VTF file often will be generated from an Agency's
            || original records kept in some other format, but not the other
            || way around, Processing.DateTime will not be later than
            || ThisFile.Preparation.DateTime. Of course, it is possible
            || for these two DateTimes to be identical if the originating-
            || Agency directly creates this tagged-format file.
```

```
ThisFile.DataSeriesHistory[(n)]_txt = “<DataSeries ID>”;                    || Optional
            || Any current or previous DataSeriesCOSMOSidentifier
            || and/or filenames of a predecessor DataSeries in the
            || recording and processing sequence leading to this
            || DataSeries.
            ||
            || If given, this tag array should begin with the oldest
            || identifier, index (1), and proceed to the current
            || identifier, index (N)—that is, in time order.
            ||
            || It to be understood that filenames are mercurial, that
            || anyone along the chain of records can easily change one.
            || The DataSeriesCOSMOSidentifier is preferable when available
            || because it is guaranteed to be unique for all time while
            || filenames given here are not guaranteed to be meaningful.

ThisFile.MetadataValid[(n)].Start.DateTime_txt =
                                “YYYY-MM-DD[ hh:mm:ss[Z|[+|-]hh[:mm]]]”;    || Optional
ThisFile.MetadataValid[(n)].Stop.DateTime_txt =
                                “YYYY-MM-DD[ hh:mm:ss[Z|[+|-]hh[:mm]]]”;    || Optional
ThisFile.MetadataValid[(n)].Agency_txt = “<from table 3E(b)>”;              || CONDITIONAL
            || As distinct from ThisFile.Preparation.DateTime, these tags
            || provide an optional means of indicating the date (± time)
            || interval during which the provider believes that the metadata
            || in this header that is supplied by the indicated Agency are
            || valid in their entirety (an interval often called an “epoch”).
            || Users occasionally request such epochal information.
            ||
            || If either or both the Start.DateTime and the Stop.DateTime
            || is/are given, then the associated Agency is REQUIRED
            || (meaning, of course, that the stated epoch applies only
            || to metadata generated by said Agency).
            ||
```

|| The most common application of these epochal tags is likely
|| to be just leaving them out, thus accepting their default
|| values.
||
|| The default value of either tag is effectively the moment
|| of VTF-file creation but specifically nearer end of this
|| file's ***DataSeries***, with no other promises made. That is,
|| the values are valid for the duration of the ***DataSeries***.
||
|| If given at all, it is likely that one set of these epochal
|| tags will be given, and is likely to (but not required to)
|| refer to the same ***Agency*** as ***ThisFile.Preparation.Agency***,
|| the author of this VTF file. That is, the author of this VTF
|| file is not likely to make any promises for other suppliers
|| of metadata, such as earthquake ***Hypocenter*** data.

```
ThisFile.ValidBand.HighPass.Corner_dbl = <value> [ Hz | s ];                    || Optional
ThisFile.ValidBand.LowPass.Corner_dbl = <value> [ Hz | s ];                     || Optional
ThisFile.ValidBand.Agency_txt = "<from table 3E(b)>";                           || Optional
ThisFile.ValidBand.Comments_txt = "<Comments>";                                 || Optional
```

|| In the event that these are preliminary ("quick release")
|| ***DataSeries***, it is often the case that response spectra and
|| other information are valid only over a limited bandwidth.
|| If this is so, we recommend an explicit statement of the
|| valid frequency or period band *via* these tags. Obviously,
|| ***HighPass*** and ***LowPass*** have opposite meanings when expressed
|| in period than when they are expressed in frequency.
||
|| See also, ***Processing.HumanReview***, ***Processing.Stage***,
|| ***Processing.ConstantsUsed***, and ***Processing.Problem.Status***.
||
|| While we think them unlikely to be needed, for completeness
|| we provide space for both ***Comments*** and an ***Agency*** associated
|| with this band-limit determination. (If not specified here,
|| the ***Agency*** may safely be assumed to be the ***Processing.Agency***.)

```
ThisFile.NullComplexValue_dbl = <value> <value>;                    || Recommended
ThisFile.NullRealValue_dbl = <value>;                               || Recommended
ThisFile.NullIntValue_int = <value>;                                || Recommended
            || These values are not used in the VTF per se. They are used
            || only as "option two of four" (below) for parser handling of
            || NULL tag values.
            ||
            || For NULL-valued tags, from high to low priority:
            || ------------------------------------------------------------
            || "Option one" is to use user-supplied values as given to the VTF
            || reading routine; these are to override the other three "options",
            || though such action is not recommended because of the potential
            || for value conflicts.
            ||
            || "Option two" is to use the values given by these ThisFile.Null…
            || tags for numerics and empty strings for _txt. (That is, if you desire
            || Options three or four, do not provide these tags or remove them from
            || a file sent to you.)
            ||
            || "Option three", the preferred one, is to use an IEEE NaN for _dbl and
            || a pair of these for _cpx values, empty strings for _txt values, and
            || the largest possible negative integer (in most cases, [1 - 2^31] =
            || -2,147,483,647) for _int.
            ||
            || "Option four" is to use the largest possible negative values
            || on that computer for all numerics and empty strings for _txt.
            || ------------------------------------------------------------
            ||
            || Note that with "Option one of four" all parsers must allow
            || users to override these three NULL-values, as well as the
            || preferred default of "Option three".
```

```
ThisFile.Annotations[(m)].TextValue_txt = "<Comments>";                    || Optional
ThisFile.Annotations[(m)].Agency_txt = "<from table 3E(b)>";               || CONDITIONAL
            || Agency which contributed the comments Annotations(m).TextValue
            || and those comments. Agency is REQUIRED if TextValue is present.
            || (That is, it is wise to identify the source of each TextValue
            || because those comments can come from multiple Agencies.)
            ||
            || If you need more than one line of text put it all into one,
            || long string with "lines" separated by the string "/\n/".
ThisFile.Citation_txt[(n)] = "<citation>";                                 || Optional
ThisFile.URL_txt[(n)] = "<Web address>";                                   || Optional
            || If there is/are report(s) available for this ThisFile,
            || provide a citation with one or both of these tags.
```

Therefore, typical VTF files will begin with:

```
ThisFile.Format_txt = "VTF.1.0";
ThisFile.DataSeriesCOSMOSidentifier_txt = "<value>";
ThisFile.Preparation.Agency_txt = "<value>";
ThisFile.Preparation.DateTime_txt = "YYYY-MM-DD";
```

where the various values are inserted to identify this record and its origins.

Table 3B(a). Type of *DataSeries*

```
DataSeries.PhysicalParameter_txt = "<from table 3B(b)>";                   || REQUIRED
            || Units are given by DataSeries.OrdinateUnits in table 3R, not here.
            || This tag provides a means of giving a somewhat more detailed text
            || description of the type of data than units alone would provide.
            || DataSeries.PhysicalParameter is in the style of the Blue Book
            || text description of the data (table 3B(b)).
```

```
DataSeries.NonLinearComputation[(m)].Description_txt = "<value>";                    || CONDITIONAL
DataSeries.NonLinearComputation[(m)].URL_txt = "<Web address>";                      || CONDITIONAL
            || At least one such tag is REQUIRED if the physical parameter
            || is a nonlinear response spectrum or the result of some other
            || nonlinear computation, in which case the method of computation
            || is to be described here, either by using a well-known name or
            || a citation (Description) or a Web site (URL) containing a
            || similar description of the computation.
```

Table 3B(b). Codes for Types of *DataSeries*

Allowed values	*	Meaning	Typical units for *DataSeries.OrdinateUnits*
"UnProcessed Acceleration"	A	"Volume 1" linear acceleration with calibrations applied but baseline NOT processed, though a simple mean may be removed	cm/s/s, g_local, counts, and so forth
"Processed Acceleration"	A	"Volume 2" linear acceleration with baseline fully processed, ready for double integration, possibly filtered too.	cm/s/s, g_standard, counts, and so forth
"Calibration Input Signal"	CAL	A signal input to the named instrument to calibrate it (in any appropriate units).	cm/s/s, cm/s, cm, g_local, counts, and so forth
"Acceleration Correction Baseline"	BL	† A full time series of the total corrections applied to a "Volume 1" acceleration time series to generate an acceleration time series which integrates successfully into the same velocity and displacement time series reported by the providing ***Agency***.	cm/s/s, g_standard, counts, and so forth

“Velocity”	V	Linear velocity	km/s, m/s, cm/s, in/s
“Absolute Displacement”	D	Linear absolute displacement	m, cm, in
“Relative Displacement”	D	Linear relative displacement (for example, displacement computed from an acceleration record)	m, cm, in
“Rotational Acceleration”	RA	Angular acceleration	deg/s/s
“Rotational Velocity”	RV	Angular velocity	deg/s
“Rotational Displacement”	RD	Angular displacement	deg
“Absolute Pressure”	P	Absolute pressure	Pa, Baryes, psi
“Relative Pressure”	GP	“Gage pressure”, that is, pressure relative to atmospheric	Pa, Baryes, psi
“Volumetric Strain”	S	Volumetric strain	ustrain
“Linear Strain”	S	Linear strain	ustrain
“Response Spectra (Sa)”	Sa	Acceleration Response Spectra, Sa, Sv, Sd, pseudovelocity (ω*Sd = PSV), or pseudoacceleration (ω*ω*Sd = PSA)	cm/s/s, and so forth
“Response Spectra (Sv)”	Sv		cm/s, and so forth
“Response Spectra (Sd)”	Sd		cm, and so forth
“Response Spectra (PSA)”	PSA		cm/s/s, and so forth
“Response Spectra (PSV)”	PSV		cm/s, and so forth
“Geometric Mean PSA”	gPSA	Geometric mean of horizontal-component linear PSAs.	cm/s/s, and so forth
“Median PSA over 90 degrees”	mPSA	Median linear PSA for horizontal component rotated through 90 degrees, preferably in one-degree increments.	cm/s/s, and so forth

“Nonlinear Response Spectra (Sa)”	nSa	Nonlinear response spectra. Describe method of computation in ***DataSeries.NonLinearComputation.Description*** or ***.URL.***	cm/s/s, and so forth
“Nonlinear Response Spectra (Sv)”	nSv		cm/s, and so forth
“Nonlinear Response Spectra (Sd)”	nSd		cm, and so forth
“Nonlinear Response Spectra (PSA)”	nPSA		cm/s/s, and so forth
“Nonlinear Response Spectra (PSV)”	nPSV		cm/s, and so forth
“Geometric Mean Nonlinear PSA”	gnPSA	Geometric mean of horizontal-component nonlinear PSAs.	cm/s/s, and so forth
“Median Nonlinear PSA over 90 degrees”	mnPSA	Median nonlinear PSA for horizontal component rotated through 90 degrees, preferably in one-degree increments.	cm/s/s, and so forth
“Acceleration Amplitude Spectrum”	DFT	Absolute value of DFT	for example, (cm/s/s)/Hz
“Acceleration Complex Spectrum”	CFT	Complex DFT Spectrum	
“Acceleration RMS Spectrum”	RMS	RMS of noise in acceleration (square root of power spectrum)	for example, (cm/s/s)/(root Hz)
“Acceleration Power Spectrum”	PWR	Power spectrum of ***Event***s or noise recorded in acceleration	for example, (cm/s/s)^2/Hz
<similarly for “Velocity”, and so forth>			
“_____”	“_”	USER-EXTENSIBLE LIST (use established, terse, clear, English names or phrases which must begin with **“User’s description: ”**)	<time-series or spectral units>

*Abbreviations recommended for use in VTF filenames to identify type of trace contained in the file.

†Corrections to preliminary velocity and displacement time series can be and should be translated back to the original acceleration record, and this corrected acceleration record ideally should be integrated to produce the processed velocity and displacement ***DataSeries*** actually distributed. Some authors prefer to correct, integrate, correct, and integrate again, however this process does not yield an acceleration trace that integrates correctly to displacement, and the practice is discouraged. It is easiest for users if corresponding acceleration, velocity, and displacement time series are related to each other by simple, trapezoidal-rule integration.

Table 3C(a). Record Types

```
DataSeries.Cause_txt = "<value>";                                          || REQUIRED
            || A value from the upper three segments of table 3C(b)
            || describing what caused this record to be recorded. Note
            || that some coordination is required with PhysicalParameter.
DataSeries.Calibration.InputType_txt = "<value>";                          || CONDITIONAL
            || If DataSeries.Cause_txt is a "… Calibration", then this
            || tag is also REQUIRED and must be equal one of the values in
            || the last segment of table 3C(b) or to a user-created name.
DataSeries.Calibration.InputAmplitude[(n)]_dbl = <value> <units>;          || CONDITIONAL
            || Recommended for "Step", "Square Wave", and "Random Steps"
            || calibrations whenever the value is known. The (n) here
            ||  and in the next tag may be used when the values vary
            ||  during the calibration, as in "Random Steps".
DataSeries.Calibration.InputStepDuration[(n)]_dbl = <value> s;             || Optional
            || Recommended for "Square Wave" calibrations if known and
            || for "Random Steps" if at all possible. To accommodate
            || the latter, this is the half period, the duration of a peak
            || or trough.
DataSeries.Calibration.InputURL_txt = "<Web address>";                     || Optional
            || Recommended for signals that are too complex to describe
            || well here—Web address(es) of a formal description, which
            || may be a time series. Indeed, this URL may point to a
            || "Calibration Input Signal" (CAL) type of VTF file.
```

Table 3C(b). Record-Type and Input-Type Codes

Record type	**Meaning**
Use any of the following as values for ***DataSeries.Cause***:	
"Seismic Trigger"	An Earth signal triggered the recorder
"Remote Trigger"	A radio, landline, or other external trigger
"Preset Trigger"	A timed, preset trigger
"Manual Trigger"	A human finger on a button at the ***DAU***
"Function Test"	Some recorder-functionality test other than calibration
(**Avoid** the following four)	
~~"Sensor Calibration"~~	~~The ***Sensor*** is being calibrated~~
~~"Amplifier Calibration"~~	~~The amplifier is being calibrated~~
~~"Recorder Calibration"~~	~~The recorder as a whole is being calibrated~~
~~"Other Calibration"~~	~~Some other element of the system is being calibrated?~~
(Preferred calibration "***Event***" names)	
"Sensor Through Recorder Calibration"	A Calibration pulse or sequence is introduced into the ***Sensor*** and recorded after the ADC (or equivalent location for analog recorders).
"Preamplifier Through Recorder Calibration"	A Calibration pulse or sequence is introduced into the first-stage amplifier and recorded after the ADC or equivalent.
"Filter Through Recorder Calibration"	A Calibration pulse or sequence is introduced after the amplifiers but before the (anti-alias) filters and recorded after the ADC or equivalent.
"Recorder Calibration"	A Calibration pulse or sequence is introduced into the ADC inputs (after the final amplifiers and filters for analog).

Use any of the following as values for ***DataSeries.Calibration.InputType***:	
“Step”	Heavyside function, a step in acceleration.
“Square Wave”	Square wave of known period and amplitude.
“Random Steps”	A signal consisting of step offsets of random size at random intervals. (A “random binary” ≡ “telegraph”—two level—calibration signal is considered a special case of “Random Steps”—constant amplitude and varying duration.)
“~~_____~~”	USER-EXTENSIBLE LIST (use terse, clear, English descriptions which must begin with **“User’s description: ”**)

NOTE: The lined-out items and their proposed replacements need to be reconsidered with proper consultation by electrical engineers. What is the intended distinction between an “amplifier” and “recorder” calibration here? How does one calibrate the amplifier, but not the ADC (analog-to-digital converter)? It can be done by elimination, or by isolation from the ADC and recorder, but most systems cannot do this—it is a laboratory test. More discussion is needed and the distinctions need to be drawn here. We offer the subsequent suggestions, which follow the points at which most systems (DASs) allow signals to be injected and recorded.

Table 3D. Geographic Naming Information

```
GeoLocation.Name[(m)].Agency_txt = “<from table 3E(b)>”;                    || REQUIRED
            || Owner or responsible group who/which assigned the following
            || GeoLocation identifier(s) (that is, station names) and related
            || data. That is, WHOSE naming, coordinates, and other data
            || values are these?
            ||
            || EXACTLY ONE SUCH TAG IS REQUIRED for every (m) appearing with
            || any of the following “GeoLocation.Name” tags. Since at least
            || one such GeoLocation.Name is REQUIRED, at least one such Agency
            || must be identified. Of course, the (m) must correspond between
            || the various tags.
```

```
GeoLocation.Name[(m)].DateTime_txt = "YYYY-MM-DD[ hh:mm:ss [Z|[+|-]hh[:mm]]]";      || Recommended
            || This date (± time) is either when this GeoLocation.Name was
            || assigned (a "starting date"), or when these naming data were
            || obtained from the Agency.
```

At least one of ***GeoLocation.Name.SNCL***, ***GeoLocation.Name.ExpandedSNCL***, or ***GeoLocation.Name.ShortName*** is REQUIRED. More than one may be given.

```
GeoLocation.Name[(m)].SNCL_txt = "SSSSS.NN.CCC.LL";                              || CONDITIONAL
            || The FDSN SNCL format "Station.Network.Channel.Location";
            || GeoLocation name. (In SNCL terminology, GeoLocation
            || would be "Station", hence "SNCL" and "SSSSS".)
GeoLocation.Name[(m)].ExpandedSNCL_txt = "SSSSSS.NN.CCC.LLL";                    || CONDITIONAL
            || For use when the user would like to follow the SNCL format
            || but doing so is not practical because field lengths are
            || longer than those defined formally by FDSN. For example,
            || a six-digit or longer GeoLocation.Name or long "Location"
            || (LL) fields designating channels within large Arrays (and
            || certainly across the network operated by one Agency) will
            || require larger SNCL fields.
            ||
            || The field lengths for "SSSSSS" and "LLL" are shown only for
            || illustration. There is no limit on the length of any field
            || except "CCC", which should follow FDSN conventions. We also
            || recommended that the "NN" field be used as defined by FDSN
            || (therefore, if your network does not yet have an FDSN code,
            || we recommend that you obtain one—it's easy and FDSN codes
            || are widely recognized as coordinated, international network
            || symbols).
            ||
            || The ExpandedSNCL is the VTF preferred solution for names not
            || strictly fitting within the FDSN SNCL format. ExpandedSNCL
            || is self-contained, systematic, and a format familiar to many
            || users, as well as having some potential for compatibility with
            || future FDSN naming evolution.
            ||
```

|| Whether this extension will be adopted by the FDSN is uncertain,
|| but we recommend it in order to be more accommodating of
|| international practices, such as increasingly common use of
|| ***GeoLocation*** names longer than five characters, as well as
|| increasingly ambitious networks with large numbers of sensors,
|| potentially into the thousands.

GeoLocation.Name[(m)].ShortName_txt = "<value>"; || CONDITIONAL
|| Any other alpha, numeric, or mixed name of any length, but
|| generally short. (This tag will often be used as a primary
|| name, but note that ***SNCL*** or ***ExpandedSNCL*** are the
|| preferred naming formats for ***GeoLocation***s in the VTF.)

GeoLocation.Name[(m)].Index_txt = "<value>"; || CONDITIONAL
|| This tag may be REQUIRED in the context of some ***Array***s,
|| as when neither ***SNCL*** nor ***ExpandedSNCL*** is used and a single
|| ***ShortName*** identifies an entire ***Array***. (If a SNCL name
|| lacking a sufficient "LL" field is used, ***Index*** also may
|| be needed.) In such cases, ***Index*** (± "CCC" codes) must be
|| used to identify particular locations and component
|| orientations within the ***Array***.
||
|| ***Index*** generally is used in conjunction with ***ShortName*** and
|| only rarely with ***SNCL*** or ***ExpandedSNCL*** names. This tag
|| is included here principally for backward compatibility.
|| We urge that ***SNCL*** or ***ExpandedSNCL*** names be used wherever
|| feasible.

GeoLocation.Name[(m)].Description_txt = "<value>"; || Optional
|| A longer description of ***GeoLocation*** coordinates, such as
|| a geographic place name with a terse site description
|| (for example, "Left abutment of … dam").
||
|| Many institutions subdivide these ***Description***s into
|| several parts, each assigned to different variables,
|| cells, or lines for their internal processing uses.
|| For the VTF, please indicate these divisions with
|| the VTF new-line sequence, "***|\n|***".

```
GeoLocation.Name[(m)].Address_txt = "<Address>";                                  || Optional
                || Street Address of the GeoLocation or of any structure
                || containing the instrument. In some cases, Address might
                || be a property description more like a tax-collector's
                || parcel number or a Range-Township-Section description
                || or their international equivalents. If not clearly a
                || simple street Address, compose this name so that users
                || in other nations will easily understand its meaning
                || after decades.
                ||
                || Include the word "CONFIDENTIAL" in Address if the
                || structure- or property-owner wishes the location or
                || the presence of the instrument kept confidential
                || Or simply omit the whole of Address to assure ongoing
                || security.
                ||
                || Separate lines of the Address with the "/\n/" character
                || sequence.
                ||
                || In the case of structural or geotechnical arrays subsumed
                || within some other, typically regional networks of an Agency,
                || the Description and Address should be consistent with the
                || Array.Identifier described below, and these tags together
                || should identify the Array properly, the set of logically
                || related Sensors and DAUs deployed over a structure or work
                || site.
```

On *Sensor GeoLocation* Naming

There are three ways of naming a GEOGRAPHIC THREE-DIMENSIONAL LOCATION, and at least ONE of these CONDITIONAL tags is REQUIRED: ***GeoLocation.Name.SNCL***, ***ExpandedSNCL***, or ***ShortName***. The ***SNCL*** or ***ExpandedSNCL*** tags are the preferred VTF naming formats.

(Users of numeric names please note that "SNCL" names are entirely compatible with numerical names. Simply represent the numerals as characters and put them into the "SSSSSS" string. In any case, a string representation is inherent throughout VTF and does not abandon either generality or specificity.)

The SNCL name was carefully thought out by FDSN and IRIS and is quite general. Thus, a large proportion of names can be translated into a compatible SNCL name without loss of generality. (Sadly, however, the "SSSSS" portion is too short for a number of naming formats which use six or more characters in their primary ***GeoLocation.Name*** code. There are circumstances in strong-motion work (for example, structural and geotechnical ***Array***s) where one could make good use of a significantly longer "LL" field to name components within an ***Array***. It is for these purposes that we provide ***ExpandedSNCL***.) Nevertheless, numeric names certainly are included in the SNCL format up to five digits (and in ***ExpandedSNCL*** up to any length). Unique two-character network names (the "N" of "SNCL") are available through FDSN for use in these names. We use these FDSN codes extensively elsewhere in the VTF to identify networks and we urge their adoption by all strong-motion networks.

The other method of naming a LOCATION is ***GeoLocation.Name.ShortName***, which provides for text, numeric, or mixed strings of any type and size, often with ***Index*** and additional details.

With any of these three primary naming tags, the user may include the ***Index*** tag. However, this "sub-name" tag, equivalent to the "LL" field of ***SNCL*** or ***ExpandedSNCL***, is given primarily for backward compatibility. We urge that ***Index*** not be used and that SNCL names be used in their stead. (***Index*** replaces the "Location Index of sensor site" is COSMOS v1.20 fixed-field format ***Sensor*** information and used to identify ***Sensor*** elements within ***Array***s.)

The name tag ***GeoLocation.Name.Description*** is for an additional narrative description, and may be used, if desired. It is not a substitute for using one of the three primary naming tags (***SNCL***, ***ExpandedSNCL***, or ***ShortName***). This "Long Description" is the former "Station identification" of COSMOS v1.20 and is meant to contain a descriptive naming phrase, sometimes the same name that is abbreviated in an alphabetic ***GeoLocation*** name (for example, the SNCL "SSSSS" field). An example of these descriptive names is "CA: Murietta Hot Springs; Skinner Dam, Left Abutment, Control Bldg." for the USGS NESMP station with "SSSSS" code "0720".

Note that because one can use ***Subscript***s, there is no need for the various aliases and alternate names that existed in COSMOS v1.20. Simply create more ***GeoLocation.Name(n).ShortName*** tags required to provide all names of that point

in 3D space. Each of these tags requires a corresponding ***Name(n).Agency*** to identify that ***Name***'s owner and/or creator. Between them, ***ShortName***, ***SNCL*** and ***ExpandedSNCL*** replace "Station number", "Station alphanumeric code", and "Secondary station number" of the COSMOS fixed format v1.20. Similarly, ***Subscript***s and the two tags ***GeoLocation.Name(m).Agency and DateTime*** offer the necessary mechanism for describing the timing of name changes, any secondary naming, resurveying of a ***GeoLocation***, small moves of a ***GeoLocation***, and any out-of-date names that may have been used in the past for a given ***GeoLocation***.

SNCL Naming Details

GeoLocation.Name.SNCL names must follow the FDSN SNCL format rules while ***ExpandedSNCL*** must mimic those rules closely, including use of the same "CCC" sensor orientation-and-type ("channel") codes. For ***ExpandedSNCL***, the "SSSSS" and "LL" (and, reluctantly, the "NN") fields may be lengthened as needed, but should remain as terse as is compatible with the data-provider's practices. If either of these SNCL tags is used as the primary ***GeoLocation.Name***, then the network codes in ***Array.Agency*** and ***GeoLocation.Name.Agency*** should agree with the network-code field ("NN") of the SNCL name except, perhaps, in rare cases of multiple names for the same ***GeoLocation*** used by different ***Agencies***.

The FDSN SNCL naming conventions are: Any five or fewer alpha and/or numeric characters for the ***GeoLocation*** name "SSSSS"; an FDSN-assigned two-character "network" code, "NN"; a SEED-format "channel" code, "CCC"; and, optionally, a two-character "tie-breaking" "location" code, "LL", to be used in cases where you have more than one channel with the same "SSSSS.NN.CCC" name (as will often be the case in structural and geotechnical ***Array***s). An "LL" might be used, for example, if one specified a structure by "SSSSS.NN", a location within a structure by "LL", and the orientation and type of sensor at that location by "CCC". (Since many components inside a structure point the same direction, there will routinely be conflicting names unless one includes the "LL" code to distinguish specific locations [and/or orientations] within the structural ***Array***.)

For the SEED-format channel code, "CCC", consult Appendix A of the SEED manual, which may be downloaded from http://www.iris.edu/manuals/SEEDManual_V2.4.pdf. Succinctly, the three letters correspond to (1) a frequency band, (2) the type of ***Sensor***, and (3) the orientation of the ***Sensor***. The most common SEED-format CHANNEL codes for accelerometers sampled at 80 Hz or higher would be "HNZ", "HNN", and "HNE", for truly cardinally oriented

components, or “HNZ”, “HN2”, and “HN3” for a vertical and two non-cardinal, orthogonal, horizontals, and possibly “HNZ”, “HNT”, and “HNR” for a beam-forming ***Array*** or components rotated toward the source azimuth (“R” and “T” meaning “Radial” and “Transverse” in this context), as will be the case for some geotechnical ***Array***s. The “N” means “acceleration”, and the “H” means a lower-corner period of 10 s or longer and a sampling rate of 80 Hz or faster.

Note that SNCL names *are case sensitive*, but there are ***no*** rules in the “SSSSS” part about what order or amount of alphas and numerics there may be. In other words, users may put five-digit or shorter numeric names into ***SNCL*** and any six-digit or longer numeric names into the ***ExpandedSNCL*** name.

If you are using not ***ExpandedSNCL*** or ***SNCL*** (that is, you are using only ***ShortName***, possibly in conjunction with ***Index***), then *you must also include* an ***Array.Agency*** identification tag from table 3E(b) (the value generally should equal the value of ***GeoLocation.Name.Agency***). In all cases, the FDSN/IRIS/SNCL network name code (the second column in table 3E(b)) is strongly preferred in this application also, since it is assigned and recognized through international agreements.

Table 3E(a). Array Variables, including Network/Owner/Agency Codes

```
Array.Agency_txt = “<from table 3E(b)>”;                                    || CONDITIONAL
                || The Array's owner or operator of the recording network.
                ||
                || REQUIRED unless using either SNCL or ExpandedSNCL.
                || If present, generally should equal the value given to
                || GeoLocation.Name.Agency.
                ||
                || This tag refers to the Agency (typically an organization
                || but sometimes an individual) operating this geotechnical,
                || structural, or similarly compact Array. It is not meant
                || to be the owner/operator of any larger network subsuming
                || this local Array. However, it will commonly be the same
                || as Name.Agency.
                ||
```

```
            || (For example, although the USGS's NESMP operates many
            || structural Arrays, of various names or addresses, the
            || Array.Agency, Name.Agency, and Processing.Agency tags
            || will all be "NP". In contrast, both the NGA and PEER
            || data sets would normally use a different value for
            || Processing.Agency—themselves, because they do their
            || own processing of other Agencies' records.)
            ||
            || CAUTION: The DateTime associated with this Agency is
            || assumed to be the same as Processing.DateTime.

Array.Identifier_txt = "<value>";                                    || Optional
            || If the structural, geotechnical, or otherwise distinct
            || Array of which this instrument is part has some overarching
            || name other than that associated with the GeoLocation.Name
            || given in Array.Agency, provide it here.
            ||
            || Make this Identifier consistent with the values of
            || GeoLocation.Name.Description and Address when these tags
            || also describe structural and other local Arrays.

Array.Owner_txt = "<from table 3E(b)>";                              || Avoid
            || Table 3E identification of Array owner, if different
            || from Array.Agency. It is unlikely that this tag will
            || be any more than rarely used since nearly all Arrays
            || have a name or code that implies the Owner.
Array.Operator_txt = "<from table 3E(b)>";                           || Avoid
            || Table 3E identification of Array operator, if different
            || from Array.Agency or Array.Owner.
Array.Affiliate_txt = "<from table 3E(b)>";                          || Avoid
            || Identification of some other Array affiliate, if the above
            || names are an incomplete list of Agencies associated with
            || this Array.
```

The three tags just above are partially conflicting with ***Array.Agency*** and ***ThisFile.MetadataValid.Agency*** and are only given to provide backward compatibility; their use is *not* recommended.

```
Array.EarthGravity_dbl = <value> cm/s/s;                                    || CONDITIONAL
            || Local value of Earth acceleration at the Array. This value
            || DEFINES the unit "g_local", and therefore is REQUIRED if the
            || units "g_local", "mg_local", or "ug_local" are used anywhere
            || including if the values of the time series used here were
            || converted from one originally expressed in g_local. In other
            || words, Array.EarthGravity may be the scaling factor used
            || to convert "Volume 1" units of g_local into absolute units
            || such as cm/s/s when the original is only known in units of
            || local Earth acceleration, and, if so, this conversion factor
            || must be given.

RawSeries.GravityCalibratedAgainst_txt = [ "g_local" | "g_standard" ];     || CONDITIONAL
RawSeries.GtoGalConversionConstant_txt = [ "g_local" | "g_standard" ];     || CONDITIONAL
            || Both are REQUIRED whenever the RawSeries was converted from
            || acceleration values expressed in "g_local" or "g_standard"
            || into absolute units such as "cm/s/s".
            ||
            || In some (many?) cases, unfortunately, these two values will
            || differ, as when the instrument is calibrated against a local
            || Earth-acceleration value (generally by a flip test at the site),
            || but they are converted with the standard value, 980.665 cm/s/s.
            || Although this distinction generally makes only a several
            || part-per-thousand difference and is smaller than some
            || other uncertainties, it is best to document it.
            ||
            || RawSeries.GravityCalibratedAgainst indicates whether values
            || of acceleration given in this VTF file as absolute units like
            || "cm/s/s" originally were expressed (generally in the RawSeries)
            || in units of either "g_standard" (980.665 cm/s/s) or "g_local"
            || (defined by Array.EarthGravity) and converted to absolute
            || units using the value RawSeries.GtoGalConversionConstant.
            ||
            || Note that this distinction is implicit in the manner in which
            || the Agency calibrated its instruments and processed its
            || data. In practice, many operators do calibrations by doing
            || a flip test on site at the Sensor's GeoLocation, so that it
            || is calibrated relative to Array.EarthGravity. These data
            || typically are converted by g_standard.
```

```
RawSeries.ScaleToDataSeries_dbl = <value> <units>;                    || CONDITIONAL
            || Similarly, if the RawSeries was converted from any unit,
            || such as "counts", to some other unit, <value> is that
            || (multiplicative) conversion constant and <units> are in
            || the sense <output units> / <input units>, for example,
            || (cm/s/s)/count to convert from a RawSeries in digital
            || counts to a DataSeries in gals.
```

Table 3E(b). COSMOS and FDSN Array/Network Codes

COSMOS Code	FDSN Code	Network; Owner
USCGS	-	U.S. Coast and Geodetic Survey
NESMP	NP	National Engineering Strong Motion Program (NESMP), U.S. Geological Survey (USGS)
USBR	RE	U.S. Bureau of Reclamation
ACOE	-	U.S. Army Corps of Engineers
CDMG or CGS	CE	Calif. Geological Survey, formerly Div. of Mines and Geology (CGS/California Strong Motion Instrumentation Program (CSMIP))
CIT	CI	Calif. Institute of Technology
UCB	BK	Berkeley Digital Seismic Network (BDSN); University of California, Berkeley
CWB	TW	Taiwan Central Weather Bureau (CWB)

-	AA	Anchorage Strong Motion Network; Geophysical Institute, Univ. of Alaska, Fairbanks
-	AV	Alaska Volcano Observatory; USGS, Anchorage
-	AZ	ANZA Regional Network; University of California, San Diego and USGS, Menlo Park, Calif.
-	BO	NIED (Bosai-Ken Network); National Institute of Earth Science and Disaster Prevention
-	CN	Canadian National Seismic Network; Geological Survey of Canada
-	NC	USGS Northern California Regional Network; USGS, Menlo Park, Calif.
-	NN	Western Great Basin/Eastern Sierra Nevada; Univ. of Nevada, Reno
-	SN	Southern Great Basin Network; Univ. of Nevada, Reno
-	US	U.S. National Seismic Network; ANSS backbone of USGS/NEIC, USGS/ASL and EarthScope Project of IRIS
-	UU	University of Utah Regional Network; University of Utah
-	UW	Pacific Northwest net; University of Washington and USGS
-	WY	Yellowstone Seismic Network; University of Utah (formerly by USGS, Menlo Park)
	**	**All other FDSN/IRIS/SNCL network names are subsumed as acceptable names here.**
_____		USER-EXTENSIBLE LIST (for networks not named above, use established, terse, clear, English names or well-known abbreviations which must begin with **"User's description: "**)

NOTES:

A complete set of these SNCL network abbreviations can be found at the following URLs.

For permanent networks: http://www.iris.washington.edu/stations/networks.txt.

For portable networks: http://www.iris.edu/stations/networks.portable.txt.

Array.Agency is REQUIRED unless the SNCL name ***GeoLocation.Name.SNCL*** or the SNCL-like name ***GeoLocation.Name.ExpandedSNCL*** is used.

Geographic-Position Issues

Regarding the following, table 3F, we believe that two methods are necessary and sufficient for specifying the three-dimensional position and of a ***GeoLocation*** and the orientation of a ***Sensor*** in space:

(1) Absolute coordinates and orientations (that is, locations within established geographic datums and orientations as azimuths from geographic true North and declinations from plumb vertical (that is, to the local gravity acceleration vector); or

(2) locations relative to a specified reference location and orientations relative to a reference orientation (such as the azimuth of a structure's long axis), with both references given in the absolute frames of (1).

Description method (2) typically would be used for structural and geotechnical ***Array***s and certain other compact ***Array***s. While it may be the case that the reference point and orientation are not precisely known, they should always be given, even when large corresponding error estimates must also be given. It is better to have a general idea where the ***Array*** is than no idea at all. (Of course, if "no idea at all" is all there is, give the value NULL.) It will often be the case in situations where absolute reference location and azimuth are poorly known that the offsets in location and orientation from these references will be well characterized (as in meters along, across, and up a structure and whether the sensor is wall-parallel or -perpendicular). Even if not, provide the available information and reasonable error estimates for same.

Both forms of ***Sensor*** location (for example, ***GeoLocation.Location.Latitude***) are addressed in table 3F. Both forms of ***Sensor*** orientation are addressed in table 3K(a). There are more- and less-preferred methods for both, the less preferred methods provided for backward compatibility.

The first 13 tags of table 3F are the only place in which the absolute location of the ***Sensor*** is defined. The first five define the absolute location while the latter eight define an absolute (***Array***) reference location point plus the offsets from that point to the ***Sensor***. Absolute locations typically are used in single-***GeoLocation*** "***Array***s" (typically free-field sites) while relative locations are typically used in structural and other ***Array***s containing ***GeoLocations*** distributed over some larger but relatively compact area or volume of observation.

Of the CONDITIONAL tags among those first 13, at least one complete set must be selected, either the first three (absolute) or the latter six (relative).

On the Accuracy of Geographic Positions

If care is taken with conversions into VTF text and if IEEE double precision variables are used throughout in binary ***Processing***, then in theory it is possible to store locations accurate to 15.65 digits (52-bit mantissa). (One would write out 16 to 17 digits to avoid round-off errors and maintain this degree of accuracy.) For ***Latitude***, this precision equates to about 25 pm (picometers), vastly better than is required. Geographic location precision of 10 cm (differential GPS accuracy) requires at least seven digits after the decimal for ***Latitude***. ***Longitude*** values are similarly or more precise when given with the same number of digits after the decimal point.

Table 3F. Other Geographic and Array Tags—Coordinates

Absolute geographic position of the ***Sensor*** (there may be ***(n)*** location attempts by one or more ***Agencies***, including a history of small station relocations (that is, by a few meters to tens of meters, a common exigency to accommodate changing field conditions):

Please note that it is generally desirable to provide exactly one such ***Location***, the one pertaining to the ***DataSeries*** in this VTF file (and, therefore, no explicit ***(n)***). However, we provide the option of giving a history of ***GeoLocation*** movements and remeasurements.

If more than one ***GeoLocation.Location*** is given, put last (the highest index so "most recent" values) the authoritative values applicable to this ***DataSeries*** and be sure its ***DateTime*** is consistent with the ***DateTime*** of this VTF file (***ThisFile.Preparation.DateTime*** and ***ThisFile.MetadataValid***).

Absolute geographic position of the proof mass:

```
GeoLocation.Location[(m)].Latitude_dbl = <value> deg;                    || CONDITIONAL
            || Absolute, positive North, decimal degrees
GeoLocation.Location[(m)].Longitude_dbl = <value> deg;                   || CONDITIONAL
            || Absolute, positive East, decimal degrees
GeoLocation.Location[(m)].Elevation_dbl = <value> <units>;               || CONDITIONAL
            || Elevation above datum (meters preferred).
            ||
            || NOTE: While Sensor Elevations have been routinely
            || omitted from strong-motion site coordinates (these not
            || generally being of interest to engineers), they are of
            || interest to seismologists and some research engineers.
            || Elevation should be given in all future data sets if
            || at all possible.
            ||
            || See ElevationOfGradeLevel (and
            || NumberOfStories.[Above|Below]Grade) for all sensors
            || located away from grade level.

GeoLocation.Location[(m)].ElevationOfGradeLevel_dbl = <value> <units>;   || CONDITIONAL
            || Depth of proof mass below surface (meters preferred).
            ||
            || REQUIRED if the sensor is more than one meter from grade
            || level (as in a borehole or other underground location).
            || Also viable for above-grade installations, but these, in
            || general, should be supplied by using relative coordinates.
            ||
            || This tag is the only way, other than Comments, to indicate
            || a borehole or equivalent location. Use it with absolute or
            || relative coordinates, or to give an explicit statement
            || of grade-level elevation.


GeoLocation.Location[(m)].HorizontalDatum_txt = "<value>";               || Recommended
            || Table 3G, "Horizontal" or "Mixed" datum
GeoLocation.Location[(m)].VerticalDatum_txt = "<value>";                 || Recommended
            || Table 3G, "Vertical" or "Mixed" datum
```

```
GeoLocation.Location[(m)].HorizontalError_dbl = <value> <units>;                || Optional
GeoLocation.Location[(m)].VerticalError_dbl = <value> <units>;                  || Optional
            || (km preferred for <units>)
            ||
            || We note the possibility of representing an uncertainty
            || ellipsoid and seek comment on this matter.
```

Relative geographic positions consist of an absolute reference point and offsets from there to the ***Sensor***s:

```
Array.Location[(m)].Latitude_dbl = <value> deg;                                 || CONDITIONAL
            || Absolute of reference, positive North, decimal degrees
Array.Location[(m)].Longitude_dbl = <value> deg;                                || CONDITIONAL
            || Absolute of reference, positive East, decimal degrees
Array.Location[(m)].Elevation_dbl = <value> <units>;                            || CONDITIONAL
            || Absolute of reference, elevation above datum (meters
            || preferred).
            ||
            || In the case of ANY BOREHOLE or other underground sensors,
            || you also must supply ElevationOfGradeLevel. You also may
            || supply NumberOfStories.[Above|Below]Grade, defined below.

Array.Location[(m)].HorizontalDatum_txt = "<value>";                            || Recommended
            || Table 3G, "Horizontal" or "Mixed" datum
Array.Location[(m)].VerticalDatum_txt = "<value>";                              || Recommended
            || Table 3G, "Vertical" or "Mixed" datum

Array.Location[(m)].HorizontalError_dbl = <value> <units>;                      || Optional
Array.Location[(m)].VerticalError_dbl = <value> <units>;                        || Optional
            || (km preferred for <units>)
            ||
            || We note the possibility of representing an uncertainty
            || ellipsoid and seek comment on this matter.
```

```
GeoLocation.Location[(m)].NorthOffset_dbl = <value> <units>;                    || CONDITIONAL
GeoLocation.Location[(m)].EastOffset_dbl = <value> <units>;                     || CONDITIONAL
GeoLocation.Location[(m)].ElevationOffset_dbl = <value> <units>;                || CONDITIONAL
            || North-, East-, and upward-positive offsets from the
            || Array.Location…, the reference location, for use in
            || Array-element positioning. (Meters are preferred
            || for all these units.)
GeoLocation.Location[(m)].WhichStory_dbl = <value>;                             || Optional
            || Relates to GeoLocation.Building.NumberOfStories.…
            || indicating at which story of the structure the
            || Sensor is to be found. The grade-level floor is
            || at story "0.0", upper floors have positive values,
            || and basements have negative values. Use of "_dbl"
            || implies an option to specify partial stories, for
            || example, mid-column locations.
```

For all *GeoLocation*s:

```
GeoLocation.Location[(m)].Agency_txt = "<from table 3E(b)>";                    || REQUIRED
            || Attribution of the above GeoLocation coordinates
            || (either relative or absolute). Owner (Agency) of
            || set (m) of the GeoLocation.Location coordinates.
GeoLocation.Location[(m)].DateTime_txt =
                                "YYYY-MM-DD[ hh:mm:ss [Z|[+|-]hh[:mm]]]";       || Optional
            || Either the date (± time) on which this Location was
            || determined, or the date (± time) on which these data
            || were obtained from the Agency. If not given, this
            || DateTime may be assumed to be Processing.DateTime
            || or ThisFile.Preparation.DateTime, that is, either
            || the authoritative creation date of this DataSeries or
            || of this VTF file, with no clear preference.
```

```
Array.Location[(m)].Agency_txt = "<from table 3E(b)>";                    || CONDITIONAL
            || Attribution for the above Array (reference) location.
            || Owner/provider of this set of coordinates. Often the
            || same as GeoLocation.Location.Agency, but REQUIRED
            || if an Array.Location is given.
Array.Location[(m)].DateTime_txt = "YYYY-MM-DD hh:mm:ss[Z|[+|-]hh[:mm]]";  || Optional
            || Either the DateTime that this Location was determined,
            || or the DateTime these data were obtained from the Agency.
            || If not given, assumed to be Processing.DateTime, the
            || authoritative creation date of this DataSeries.

Array.TotalRecorders_int = <value>;                                       || Recommended
            || Number of recorders in the recording system of this Array.
Array.TotalChannels_int = <value>;                                        || Recommended
            || Total number of channels in the recording system of this
            || Array. (Note also DAU.TotalChannels, needed because
            || there may be several DAUs in the Array. Similarly, the
            || tags Sensor.ArrayChannel and Sensor.DAUchannel are used
            || to identify this DataSeries' particular channel in the
            || Array and DAU.)
```

Table 3G. Geodetic Elevation Datums

Type of datum	Name of datum
Horizontal	"NAD83"
	"NAD27"
	"NZGD49"
Mixed	"WGS72"
	"WGS84"
	"ITRF00"
Vertical	"GEOID99"
	"NAVD88"
	"NGVD29"
USER-EXTENSIBLE LIST (use established, terse, clear, English names which must begin with **"User's description: "**)	"_____"

This list needs to be expanded to include, at least, the numerous datums used internationally, and perhaps those of historical interest. Full names and citations also would be useful additions.

Please use direct messages to the authors and/or the user-extensible-list option to help us make this list more complete. In particular, indicate the standard abbreviations and types of datums (horizontal, vertical, or both) for any or all of the following datums:

Adindan; Afgooye; Ain_El_Abd_1970; Alaska_(NAD-27); Alaska/Canada_NAD-27; Anna_1_Astro_1965; ARC-1950_mean; ARC-1960_mean; Ascension_Island_'58; Astro_B4_Sor.Atoll; Astro_Beacon_“E”; Astro_Pos_71/4; Astronomic_Stn._'52; Australian_Geodetic_1984; Bahamas_(NAD-27); Bellevue_(IGN); Bermuda_1957; Bogota_Observatory; Bukit_Rimpah; Camp_Area_Astro; Campo_Inchauspe; Canada_Mean_(NAD27); Canal_Zone_(NAD27); Canton_Island_1966; Cape; Cape_Canaveral_mean; Carribean_(NAD27); Carthage; Central_America_(NAD27); Chatham_1971; Chua_Astro; Corrego_Alegre; Corrego_Alegre_(Provisional); Cuba_(NAD27); Cyprus; Djakarta(Batavia); DOS_1968; Easter_lsland_1967; Egypt; European_1950; European_1950_mean; European_1979_mean; Finnish_Nautical_Chart; Gandajika_Base; Geodetic_Datum_'49; Ghana; Greenland_(NAD27); Guam_1963; Gunung_Segara; Gunung_Serindung_1962; GUX_1_Astro; Herat_North; Hjorsey_1955; Hong_Kong_1963; Hu-Tzu-Shan; Indian; Iran; Ireland_1965; ISTS_073_Astro_'69; Johnston_Island_'61; Kandawala; Kerguelen_Island; Kertau_'48; L.C._5_Astro; La_Reunion; Liberia_1964; Luzon; Mahe_1971; Marco_Astro; Masirah_Is._(Nahrwan); Massawa; Merchich; Mexico_(NAD27); Midway_Astro_'61; Mindanao; Minna; Montjong_Lowe; Nahrwan; Naparima_BWI; North_America_'83; North_America_1927_mean; Observatorio_1966; Old_Egyptian; Old_Hawaiian_mean; Old_Hawaiian_Kauai; Old_Hawaiian_Maui; Old_Hawaiian_Oahu; Oman; Ordnance_Survey_of_Great_Britain_'36; Pico_De_Las_Nieves; Pitcairn_Astro_'67; Potsdam_Rauenberg_DHDN; Provisional_South_American_1956_mean; Provisional_South_Chilean_1963; Puerto_Rico; Pulkovo_1942; Qornoq; Quatar_National; Rome_1940; S_42; S.E.Asia_(Indian); SAD-69/Brazil; Santa_Braz; Santo_(DOS); Sapper_Hill_'43; Schwarzeck; Sicily; Sierra_Leone_1960; South_American_1969_mean; South_Asia; Southeast_Base; Southwest_Base; Tananarive_Observatory_'25; Thai/Viet_(Indian); Timbalai_1948; Tokyo_mean; Tristan_Astro_1968; Unites_Arab_Emirates_(Nahrwan); Viti_Levu_1916; Wake-Eniwetok_'60; WGS-72; WGS-84; Yacare; and Zanderij, (Dana, 1997).

Table 3H(a). Site Type and the Built Environment

Geologic, geophysical, and geotechnical considerations:

There may be more than one determination, ***(m)***, of the following values. Further, the sense of "Recommended" here is "strongly recommended" for at least some subset of these site-condition characterizations.

```
GeoLocation.Site.Vs[(m)].IntervalSvelocity_dbl = <value> <velocity units>;          || CONDITIONAL
GeoLocation.Site.Vs[(m)].FromDepth_dbl = <value> <distance units>;                 || CONDITIONAL
GeoLocation.Site.Vs[(m)].ToDepth_dbl = <value> <distance units>;                   || CONDITIONAL
GeoLocation.Site.Vs[(m)].Agency_txt = "<from table 3E(b)>";                        || Recommended
GeoLocation.Site.Vs[(m)].DateTime_txt = "YYYY-MM-DD[ …]";                          || Recommended
GeoLocation.Site.VsSubscript_int = <points to preferred (m)>;                      || CONDITIONAL
            || The first three of these tags are REQUIRED when any of them
            || is given; otherwise they are Recommended. The index (m) presumes
            || there may be several measurements of GeoLocation for a site,
            || often made by differing methods.
            ||
            || IntervalSvelocity is the average V_S (m/s preferred) over
            || some depth interval indicated by FromDepth and ToDepth—the
            || shallower and greater depth (preferably in meters) over which
            || the average is taken. These values either were supplied by
            || the stated Agency at the stated date (± time), or created by
            || it on that date (± time).
            ||
            || For example, Vs30 would be represented by FromDepth_dbl = 0 m;
            || and ToDepth_dbl = 30 m; with the Vs30 interval velocity given
            || in IntervalSvelocity, generally in m/s.
            ||
            || If multiple values (m) are given above, then VsSubscript is
            || REQUIRED and points to the values considered "preferred" for
            || this Sensor.
```

GeoLocation.Site.Code[(m)].Agency_txt = "<from table 3E(b)>"; || Recommended
|| The organization or individual (***Agency***) that provides
|| this ***Site.Code***.
GeoLocation.Site.Code[(m)].DateTime_txt = "YYYY-MM-DD hh:mm:ss[Z|[+|-]hh[:mm]]"; || Optional
|| Either the date (± time) that this ***Site.Code*** was first used,
|| or the date (± time) the code was obtained from the ***Agency***.
GeoLocation.Site.Code[(m)].Citation_txt = "<description of codes>"; || Recommended
GeoLocation.Site.Code[(m)].URL_txt = "<Web address>"; || Recommended
|| If possible, cite a description the type of the ***Site.Code***
|| used here.
GeoLocation.Site.Code[(m)].TextValue_txt = "<the site-condition code>"; || Recommended
|| Any site-condition code developed by any organization
|| or individual. The values for ***Agency*** may be those listed
|| in table 3E, or any other "owner" of these codes, but not
|| necessarily the organization that developed the code.
||
|| For example, for NEHRP Vs30 site codes:

NEHRP Code	VS30 Range (m/s)
A	> 1500
B	760 to 1500
C	360 to 760
D	180 to 360
E	< 180

|| the values generally would be
|| ***GeoLocation.Site.Code(n).Agency_txt*** = "<***DataSeries*** network code>";
|| ***GeoLocation.Site.Code(n).DateTime_txt*** = "<Year-Month-Day>";
|| ***GeoLocation.Site.Code(n).Citation_txt*** = "NEHRP (revised)";
|| ***GeoLocation.Site.Code(n).TextValue_txt*** = ["A" | "B" | "C" | "D" | "E"];
||
|| Other examples of codes are the GeoMatrix three-character code,
|| *Campbell's* GEOCODE, UBC site class codes (*Wills and others*, 2000,
|| and their subsequent work), and *Bray and Rodriquez-Marek's* (1997,
|| and their subsequent work) site codes. It is likely that there
|| are many others.

```
GeoLocation.Site.CodeSubscript_int = <points to preferred (m)>;                  || CONDITIONAL
            || If more than one site code is given here, indicate
            || which (m) is "preferred" above. REQUIRED if more
            || than one index is given, but may be NULL to indicate
            || "no stated preference" or "unknown".

GeoLocation.Site.IsoPleth.ShearVelocity_dbl = <velocity> <units>;                || CONDITIONAL
GeoLocation.Site.IsoPleth.Depth_dbl = <depth> <units>;                           || CONDITIONAL
            || The depth to a particular velocity horizon at
            || the Site, and the wave speed at said isopleth.
            ||
            || If Depth is given, so must be ShearVelocity.

GeoLocation.Site.Bedrock.Depth_dbl = <value> <units>;                            || Optional
GeoLocation.Site.Bedrock.Comments_txt = "<Comments ± unit name>";                || Optional
            || Depth to bedrock. For example, Depth in kilometers to
            || the Franciscan formation. Comments provides the means
            || to describe what this bedrock is, typically a unit name
            || and "[ top | bottom ] of …". (It is OK to give only
            || Comments naming the bedrock unit since sometimes this
            || is known but depth is not. In this case, it may be
            || wise to include a general Basin description, too.)
            ||
            || If either of these tags is given, then both are
            || RECOMMENDED.

GeoLocation.Site.Basin.Depth.Description_txt = [ "Shallow" | "Deep" ];           || Optional
GeoLocation.Site.Basin.Depth.Value_dbl = <value> <units>;                        || Optional
GeoLocation.Site.Basin.Depth.Error_dbl = <value> <units>;                        || Optional
GeoLocation.Site.Basin.Name_txt = "<basin's name>";                              || Optional
GeoLocation.Site.Basin.Citation_txt = "<Citation>";                              || Optional
GeoLocation.Site.Basin.Comments_txt = "<Comments>";                              || Optional
```

|| The depth, at the ***Sensor***'s ***GeoLocation,*** of a sedimentary
|| and/or volcaniclastic basin or any surface low-velocity
|| unit having a similar effect on the data. When a numeric
|| ***DepthValue*** is unknown but a general sense of the basin is
|| available, ***GeneralDepth*** is this general assessment of ***Basin***
|| depth, for example, as used by *Bray and Rodriquez-Marek* (1997),
|| who separate "Shallow" from "Deep" at 60-m depth. In contrast,
|| ***DepthValue*** is an explicit depth estimate (for example, in
|| meters). ***Name***, ***Citation***, and ***Comments*** identify the ***Basin***, to
|| which the information applies, as well as any present issues.
||
|| **CAUTION:** At some point, some sort of ***BasinEdge*** information
|| also should be defined as for the above six classes, describing
|| distances from the ***GeoLocation*** or the ***Event.Hypocenter*** to the
|| edges of ***Hypocenter*** relevant ***Basin***s, and perhaps the detailed
|| geography of the ***Basin***, but this is a complex and potentially
|| expansive issue not yet addressed.

GeoLocation.Site.Geology[(m)]_txt = "<value>"; || Recommended
|| Surface-unit geologic name (for example, "Qal"). While
|| this is an important value, we note that it is widely
|| believed to be an insufficient replacement for a
|| ***GeoLocation.Site.Code***.
||
|| If the Earth structure beneath the station is
|| known or inferred, the user may list a layered
|| medium by using ***(m)***, with ***(1)*** always being the
|| surface layer.

GeoLocation.Site.SurfaceAge_txt = "<value>"; || Optional
|| Age or epoch of the surface geologic unit. For
|| example, "Holocene" or "8000 years".

GeoLocation.Site.SurfaceGrainSize_txt = ["Aggregate" | "Coarse" | "Fine"]; || Optional
|| Grain size of the surface geologic unit

```
GeoLocation.Site.Citation_txt = “<Citation>”;                                  || Optional
GeoLocation.Site.URL_txt = “<Web address>”;                                    || Optional
            || A citation or pointer to a provider's detailed site
            || description (geology, geophysics, geotechnical,
            || structures, and any other site information
            || affecting response).

GeoLocation.Site.GeotechnicalFeature[(m)].Identifier_txt = “<value>”;          || Optional
GeoLocation.Site.GeotechnicalFeature[(m)].Citation_txt = “<value>”;            || Optional
GeoLocation.Site.GeotechnicalFeature[(m)].URL_txt = “<Web address>”;           || Optional
GeoLocation.Site.GeotechnicalFeature[(m)].Distance_dbl = <value> <units>;      || Optional
GeoLocation.Site.GeotechnicalFeature[(m)].Type_txt = “<value>”;                || Optional
GeoLocation.Site.GeotechnicalFeature[(m)].Distributor_txt =
                                          [ “GVDC” | “<other value>” ];        || Optional
GeoLocation.Site.GeotechnicalFeature[(m)].Agency_txt = “<value>”;              || Optional
            || The COSMOS Geotechnical Virtual Data Center (GVDC) will soon
            || become a leading means of reaching large stores of detailed
            || geotechnical data. The tags here anticipate the “unique
            || Identifier” of features in that virtual data base.
            ||
            || Additionally, these tags are general enough to reference
            || geotechnical data from any source, either by using the
            || Identifier, Citation, or URL, or by using data from an original
            || source Agency often through a Distributor, such as the GVDC.
            ||
            || These features are likely to include boreholes, CPT soundings,
            || trench logs, SASW and other seismic profiles, and so forth.
            ||
            || Distance provides the distance from the GeoLocation of this
            || DataSeries measurement to the location of the geotechnical
            || feature. Similarly, Type names the type of feature, preferably
            || by using the names of GVDC’s xml structures identifying the
            || feature (for example, “hole” for boreholes).
```

The built environment *versus* reference and free field:

One set of the following CONDITIONAL tags is REQUIRED to properly describe the siting conditions of this individual ***Sensor*** at this specific ***GeoLocation***.

We note that it might be thought that these should be ***Array*** tags, however, different ***GeoLocations*** within an ***Array*** of instruments can be structural, reference or free-field, and in different settings within a structure (abutment, mid-span, and so forth). For example, consider the San Francisco Bay Bridge, which has abutments on hard rock and floating in deep Bay mud, has reference sites in boreholes and on the surface of both environments, and has ***Sensor***s in numerous settings along the 10-km-long structure, on steel or concrete, old and new.

Therefore, the description must be specific to each ***GeoLocation*** within the ***Array***, specific to the single record represented in this file. While current tags do not capture the full complexity of possible descriptions, the way is opened for complete descriptions, and the process is at least brought to the point where distinctions can be made between reference sites and those on the structure.

Note that tags are grouped and a logical set of related tags is required to complete this section. If the ***GeoLocation*** is in reference of free-field conditions, then only the first tag is needed. If it is a built setting, one of the structure sets of tags is required.

Note that where an index ***(m)*** appears below, it is being used in most cases to indicate that some structure varies significantly from one area to another, or that there are several, adjacent or nearly adjacent structures, with these parts possibly performing differently or interacting in some way (for example, they may have contrasting mode periods, or there might be collisions between them, or the grouping creates a "many body" soil-structure interaction). In this case, it is preferable also to include the corresponding ***<TypeOfStructure>Subscript*** tag to tell the user which description applies best to this specific ***GeoLocation.Location*** (coordinates).

Between the REQUIRED ***GeoLocation.StructureInfluence***, and the set of CONDITIONAL tags (***NonStructuralDeploymentDescription***, ***OtheStructure.Description***, ***GeoLocation.Building.…***, ***GeoLocation.Bridge.…***, and ***GeoLocation.Dam.…***) at least a basic description of ***Sensor*** siting conditions is REQUIRED (a sufficient set of these tags).

One tag that applies to all ***GeoLocation***s, all ***Sensors***:

```
GeoLocation.StructureInfluence_txt =
                    [ "Free field" | "Reference" | "In or on structure" ];      || REQUIRED
            || Select which of the three strings best describes the degree to
            || which the built-environment (structures) impact the response of
            || this Sensor—in effect, no influence, possibly some influence,
            || or significant influence. Please give preference to using the
            || COSMOS/ANSS standards in making this distinction.
```

Descriptors that apply to all non-structural ***GeoLocation***s and ***Sensor***s:

```
GeoLocation.NonStructuralDeploymentDescription[(m)]_txt = "<Terse Description>";    || CONDITIONAL
            || One or more text description(s) from table 3H(b) which describe
            || the installation details of either a free-field or a reference
            || Sensor at this GeoLocation.
            ||
            || This tag is used only for nonstructural installations. For
            || Sensors in/on structures, use a set of the tags below.
            ||
            || It is assumed that all (m) pertain simultaneously and equally.
```

Descriptors that apply to all structural ***GeoLocation***s and ***Sensor***s:

```
GeoLocation.StructurePedigree.DateApplied.DateTime_txt = "YYYY-MM-DD[ …]";          || Optional
GeoLocation.StructurePedigree.DateApplied.TimeError_dbl =
                                        <value> [ years | months | days | … ];      || Optional
GeoLocation.StructurePedigree.DesignCodeUsed[(m)]_txt = "<Building Code(s) Used>"; || Optional
            || The estimated or actual DateTime of design or construction,
            || generally to the nearest year only, thus specifying the year
            || controlling structural qualities. That is, the inception,
            || completion, or other controlling date (± time), of the
            || structure and the uncertainty in this design-controlling
            || DateTime.
            ||
            || Also, the building code or codes (m) under which this
            || structure was designed and built. For example, "2006 IBC".
```

Descriptors that apply only to buildings:

```
GeoLocation.Building[(m)].StructuralSystem[(n)]_txt = "<Terse Description>";       || CONDITIONAL
            || Recommended when the GeoLocation is in or on a building.
            || Use one or more descriptions from table 3H(c).
            ||
            || (If a code from table 3H(e or f) is desired, instead use
            || GeoLocation.Building.HAZUSoccupancyClass and/or
            || GeoLocation.Building.HAZUSbuildingType.)
            ||
            || Use tags enumerated elsewhere below when the structure
            || is not a building (is a Bridge, Dam, or OtherStructure).
            ||
            || It is assumed that all (n) pertain equally.

GeoLocation.Building[(m)].HAZUSbuildingType_txt = "<From table 3H(e)>";            || Optional
GeoLocation.Building[(m)].HAZUSoccupancyClass_txt = "<From table 3H(f)>";          || Optional
            || Two HAZUS codes indicating the type of building, and its
            || occupancy pattern. These tags may be used in place of or
            || in addition to other descriptors.
```

GeoLocation.Building[(m)].NumberOfStories.AboveGrade_int = <value>; || CONDITIONAL
GeoLocation.Building[(m)].NumberOfStories.BelowGrade_int = <value>; || CONDITIONAL
|| Recommended if the structure is a building. Pairs with
|| ***GeoLocation.Building.StructuralSystem(m)*** to form a
|| complete description. Can be ***Subscript***ed ***(m)*** if the
|| building is complex, with sections of differing height
|| or structural type.
||
|| Note the relationship to ***GeoLocation.Location.WhichStory***,
|| and with ***ElevationOfGradeLevel***.

GeoLocation.Building[(m)].Height_txt = "[Low | Medium | High] Rise"; || CONDITIONAL
|| Recommended if the structure is a building and the
|| number of stories is not known precisely. Pairs with
|| ***GeoLocation.Building.StructuralSystem(n)*** to form
|| a complete description.
||
|| CSMIP usage is
|| "L" = "Low Rise" = 1 to 3 stories
|| "M" = "Medium Rise" = 4 to 7 stories
|| "H" = "High Rise" = 8 or more stories.
|| and is a recommended usage.
||
|| Exactly one ***GeoLocation.Building.NumberOfStories***
|| or ***GeoLocation.Building.Height*** is REQUIRED if the
|| ***GeoLocation*** is in or on a building.

GeoLocation.Building[(m)].Description_txt = "<Terse Description>"; || CONDITIONAL
|| A brief description of an instrumented building not
|| elsewhere described or describable. If you need
|| more than one line of text, use ***Subscript***s ***(m)***, or
|| put the entire description into one long string with
|| "lines" separated by the character string "***|\n|***".
GeoLocation.BuildingSubscript_int = <value of ***(m)*** where this ***Sensor*** is>; || CONDITIONAL
|| If there are more than one ***(m)*** above, then this tag is
|| REQUIRED and points to the ***GeoLocation.Building[(m)]…***
|| pertaining specifically to this ***Sensor***.

Descriptors that apply only to bridges:

```
GeoLocation.Bridge[(m)].Description_txt = "<Terse English Description>";          || CONDITIONAL
            || Brief description of an instrumented bridge not
            || elsewhere described or describable. If the Bridge
            || has more than one section, use Subscript (m).
            ||
            || For multiple lines in one description use one, long string
            || with "lines" separated by the character string "|\n|".
GeoLocation.Bridge[(m)].HAZUSoccupancyClass_txt = "<From table 3H(f)>";           || Optional
            || A HAZUS code indicating the occupancy pattern of this
            || bridge. Tag may be given in place of or in addition to
            || other descriptors.
            ||
            || We note that HAZUS does not currently list bridge-
            || occupancy descriptors for vehicular lifelines so
            || this is a placeholder at present.

GeoLocation.Bridge[(m)].SubstructureMaterial_txt = "<from table 3H(d)>";          || CONDITIONAL
GeoLocation.Bridge[(m)].SuperstructureConfiguration_txt = "<from table 3H(d)>";   || CONDITIONAL
GeoLocation.Bridge[(m)].NumberOfSpans_txt = "<from table 3H(d)>";                 || CONDITIONAL
GeoLocation.Bridge[(m)].AbutmentType_txt = "<from table 3H(d)>";                  || CONDITIONAL
GeoLocation.Bridge[(m)].SpanContinuity_txt = "<from table 3H(d)>";                || CONDITIONAL
GeoLocation.Bridge[(m)].ColumnsPerBent_txt = "<from table 3H(d)>";                || CONDITIONAL
            || If the GeoLocation structure type is a bridge and the
            || information is available, then use these six values
            || preferentially to classify the structure according to the
            || descriptive vectors of Basöz and Kiremidjian (1995, 1996).
            || (If there are strongly contrasting sections of the bridge,
            || these may be enumerated by Subscripting (m), as with the
            || East and West sections of the San Francisco Bay Bridge.)
            ||
            || The following are the equivalent vector-element names of
            || Basöz and Kiremidjian (1996):
            ||
```

|| y1n: ***GeoLocation.Bridge.SubstructureMaterial***
|| y2n: ***GeoLocation.Bridge.SuperstructureConfiguration***
|| y31: ***GeoLocation.Bridge.NumberOfSpans***
|| y32: ***GeoLocation.Bridge.AbutmentType***
|| y33: ***GeoLocation.Bridge.SpanContinuity***
|| y34: ***GeoLocation.Bridge.ColumnsPerBent***
||
|| If any one is given, the first four (for single-span bridges), or all six (for multiple-span bridges), should be included. However, for situations where not all are available, subsets are allowed and may be mixed with ***GeoLocation.Bridge.Description***.
||
|| Note that a few combinations of ***SubstructureMaterial*** and ***SuperstructureConfiguration*** are believed not to exist, at least in California (see reference). Weighted combinations of values also are possible, but are not treated here (yet) except by the option to use ***Subscript***ing ***(m)*** to describe separately several major sections of the bridge. Possible values are listed in table 3H(d).

GeoLocation.BridgeSubscript_int = <value of ***(m)*** where this ***Sensor*** is>; || CONDITIONAL
|| If there are more than one ***(m)*** above, then this tag is REQUIRED and points to the ***GeoLocation.Bridge(m)…*** that pertains specifically to this ***GeoLocation***, this ***Sensor***.

Descriptors that apply only to dams:

GeoLocation.Dam[(m)].Description_txt = "<Terse Description>"; || CONDITIONAL
|| Brief description of the dam from table 3H(c), or a brief description of an instrumented dam not elsewhere described or describable. If there is more that one, structurally unique section of the ***Dam***, use ***Subscript***s ***(m)*** to indicate these different parts. For a particular section of the ***Dam*** needing multiple descriptors put it all into one string with "lines" separated by the character string "**|\n|**".

Noting that the descriptions in table 3H(b) are very general, we seek a more detailed description of ***Dam***s. This descriptive technique should be clear, concise, fairly complete, and have at least some degree of acceptance by, and standardization within, the engineering community. Please direct suggestions to the authors.

```
GeoLocation.DamSubscript_int = <value of (m) where this Sensor is>;                || CONDITIONAL
            || If there are more than one (m) in what follows, this tag is
            || REQUIRED and points to the GeoLocation.Dam(m) pertaining
            || specifically to this GeoLocation, this Sensor.
```

Descriptors that apply to any structure not provided for elsewhere above:

```
GeoLocation.OtherStructure[(m)].Description_txt = "<Terse Description>";           || CONDITIONAL
            || A brief description of an instrumented structure not
            || elsewhere described (the entries near the bottom of
            || table 3H(c), which is a USER-EXTENSIBLE list).
            ||
            || If there is more than one contrasting section of the
            || structure, use Subscripts, (m).
            ||
            || For any particular section of the structure put it all
            || into one long string with "lines" separated by the
            || character string "|\n|".
GeoLocation.OtherStructure[(m)].HAZUSoccupancyClass_txt = "<From table 3H(f)>";    || Optional
            || A HAZUS code indicating the occupancy pattern of the
            || structure. May be given in place of, or in addition
            || to, other descriptors.
GeoLocation.OtherStructureSubscript_int = <value of (m) where this Sensor is>;     || CONDITIONAL
            || If there are more than one (m), more than one contrasting
            || section of the structure, then this tag is REQUIRED and
            || points to the (m) pertaining specifically to this Sensor.
```

Table 3H(b). Siting Conditions, Reference, and Free Field

Name	**Description [or additional information]**
Free field, ground response, or reference sites (***GeoLocation.NonStructuralDeploymentDescription(m)***):	
"Small fiberglass shelter"	Small, prefabricated, fiberglass shelter (typically 1-2 m on each dimension).
"Small metal shelter"	Small, prefabricated, metal building (typically 1-2 m on each dimension).
"Small wooden shelter"	Small, prefabricated or place-built, wood or wood-frame structure (typically 1-2 m on each dimension).
"Shallow"	***Sensor***s buried/set in ground (shallow, near surface).
Nota Bene	In the case of reference sites in light or mid-weight buildings, use the apropos "Building Type" listing below, but append the text "**|\n|REFERENCE**" in capitals to any other descriptors.
"Other Free Field"	Unspecified or unknown free-field site.
"Other Reference"	Unspecified or unknown reference site.
Array types (***GeoLocation.NonStructuralDeploymentDescription(m)***):	
"Geotechnical Array"	Geotechnical ***Array***
"Dense Array"	***Array*** with mean spacing of 1.0 to 3.0 km
"Very Dense Array"	***Array*** with mean spacing of 0.3 to 1.0 km
"Ultra-dense Array"	***Array*** with mean spacing of <0.3 km
"Other Array"	Unspecified ***Array***
If all else fails:	
"_____"	USER-EXTENSIBLE LIST (use established, terse, clear, English names or phrases OR tersely cite a paper which must begin with "**User's description:** "). Try to keep well below 60 characters.

Table 3H(c). Built Environment Values (CSMIP)

***Building* structural system (*GeoLocation.Building.StructuralSystem(m)*)**	**Append one of these Building Heights ****
"Unreinforced Masonry Bearing Wall"	(This appended text value is now replaced by ***GeoLocation.Building.Height***)
"Unreinforced Masonry with Load Bearing Frame"	
"Non-ductile Concrete Moment Frame"	
"Concrete Tilt-up"	
"Reinforced Masonry Wall without Moment Frame"	
"Concrete Shear Wall without Moment Frame"	
"Reinforced Masonry Shear Wall without Moment Frame"	
"Concrete Shear Wall and Moment Frame"	
"Steel Braced Concrete Frame"	
"Precast Concrete (Not Tilt-up)"	
"Precast Reinforced Concrete (Not Tilt-up)"	
"Braced Steel Frame"	
"Reinforced Masonry Shear Wall and Moment Frame"	
"Ductile Concrete Moment Frame"	
"Moment Resisting Perimeter Steel Frame"	
"Wood Frame"	
"Base Isolated"	
"Hospitals since 1989 (OSHPD)"	
"Retrofitted"	
"Other Building Type"	

***Dam*s (*GeoLocation.Dam.Description*):**	
"Concrete Gravity Dam"	Concrete gravity dam
"Concrete Arch Dam"	Concrete arch dam
"Engineered Fill Dam"	Engineered fill dam
"Hydraulic Fill Dam"	Hydraulic fill dam
"Unknown-Fill Dam"	An earth-fill dam of uncertain type
"Small Unengineered Dam"	Small unengineered dam (for example, cattle pond)
"Rock Dam"	Rock dam with impervious core
"Other Dam"	Unspecified dam
Other Structures (***GeoLocation.OtherStructure.Description(m)*** or ***GeoLocation.Building.StructuralSystem(m)*):**	
"Parking Structure"	Parking structure
"Multistory Mall"	Multistory mall
"Stadium"	Sports or performance open-air structure
"Other Building"	Unspecified building
"Other Structure"	Unspecified structure
"_____"	USER-EXTENSIBLE LIST (give a terse, clear, English description, which must begin with **"User's description: ").**

**For the CSMIP building-type listing, the appended notations: "L" = Low Rise (1 to 3 stories); "M" = Medium Rise (4 to 7 stories); "H" = High Rise (8 or more stories) is ; "OSHPD" = Office of Statewide Health Planning and Development. (Shakal and others 2002).

NOTE: The specific (maximum) number of a building's stories above and/or below grade may be given explicitly in ***GeoLocation.Building.NumberOfStories***, or as a general range in ***GeoLocation.Building.Height***.

Table 3H(d). Siting Conditions, Bridges

Bridge values (*Basöz and Kiremidjian*, 1995, 1996):	
"Concrete"	y11, ***GeoLocation.Bridge.SubstructureMaterial***
"Steel"	y12, ***GeoLocation.Bridge.SubstructureMaterial***
"Timber"	y13, ***GeoLocation.Bridge.SubstructureMaterial***
"Masonry"	y14, ***GeoLocation.Bridge.SubstructureMaterial***
"Concrete Girder"	y21, ***GeoLocation.Bridge.SuperstructureConfiguration***
"Steel Girder"	y22, ***GeoLocation.Bridge.SuperstructureConfiguration***
"Steel Truss"	y23, ***GeoLocation.Bridge.SuperstructureConfiguration***
"Suspension"	y24, ***GeoLocation.Bridge.SuperstructureConfiguration***
"Arch"	y25, ***GeoLocation.Bridge.SuperstructureConfiguration***
"Single"	y31, ***GeoLocation.Bridge.NumberOfSpans***
"Multiple"	
"<number>"	
"Monolithic"	y32, ***GeoLocation.Bridge.AbutmentType***
"Not Monolithic"	
"Continuous"	y33, ***GeoLocation.Bridge.SpanContinuity***
"Discontinuous"	
"Single"	y34, ***GeoLocation.Bridge.ColumnsPerBent***
"Multiple"	
"<number>"	
Bridge values for ***GeoLocation.Bridge.Description*** only	
"Other Bridge"	Unspecified bridge

Table 3H(e). Site Type and Built Environment Values—HAZUS Codes

No.	HAZUS building type	Description	Height			
			Range		Typical	
			Name	Stories	Stories	Feet
1	W1	Wood, Light Frame (≤5,000 sq. ft.)		1 – 2	1	14
2	W2	Wood, Commercial and Industrial (>5,000 sq. ft.)		All	2	24
3	S1L	Steel Moment Frame	Low-Rise	1 – 3	2	24
4	S1M		Mid-Rise	4 – 7	5	60
5	S1H		High-Rise	8+	13	156
6	S2L	Steel Braced Frame	Low-Rise	1 – 3	2	24
7	S2M		Mid-Rise	4 – 7	5	60
8	S2H		High-Rise	8+	13	156
9	S3	Steel Light Frame		All	1	15
10	S4L	Steel Frame with Cast-in-Place Concrete Shear Walls	Low-Rise	1 – 3	2	24
11	S4M		Mid-Rise	4 – 7	5	60
12	S4H		High-Rise	8+	13	156
13	S5L	Steel Frame with Unreinforced Masonry Infill Walls	Low-Rise	1 – 3	2	24
14	S5M		Mid-Rise	4 – 7	5	60
15	S5H		High-Rise	8+	13	156
16	C1L	Concrete Moment Frame	Low-Rise	1 – 3	2	20
17	C1M		Mid-Rise	4 – 7	5	50
18	C1H		High-Rise	8+	12	120

19	C2L	Concrete Shear Walls	Low-Rise	1 – 3	2	20
20	C2M		Mid-Rise	4 – 7	5	50
21	C2H		High-Rise	8+	12	120
22	C3L	Concrete Frame with Unreinforced Masonry Infill Walls	Low-Rise	1 – 3	2	20
23	C3M		Mid-Rise	4 – 7	5	50
24	C3H		High-Rise	8+	12	120
25	PC1	Precast Concrete Tilt-Up Walls		All	1	15
26	PC2L	Precast Concrete Frames with Concrete Shear Walls	Low-Rise	1 – 3	2	20
27	PC2M		Mid-Rise	4 – 7	5	50
28	PC2H		High-Rise	8+	12	120
29	RM1L	Reinforced Masonry Bearing Walls with Wood or Metal Deck Diaphragms	Low-Rise	1 – 3	2	20
30	RM1M		Mid-Rise	4+	5	50
31	RM2L	Reinforced Masonry Bearing Walls with Precast Concrete Diaphragms	Low-Rise	1 – 3	2	20
32	RM2M		Mid-Rise	4 – 7	5	50
33	RM2H		High-Rise	8+	12	120
34	URML	Unreinforced Masonry Bearing Walls	Low-Rise	1 – 2	1	15
35	URMM		Mid-Rise	3+	3	35
36	MH	Mobile Homes		All	1	10

Table 3H(f). Occupancy Classes—HAZUS Codes

HAZUS occupancy class	Description
AGR1	Agriculture Facilities and Offices
COM1	Retail Trade
COM2	Wholesale Trade
COM3	Personal and Repair Services
COM4	Professional/Technical Services
COM5	Banks
COM6	Hospital
COM7	Medical Office and Clinic
COM8	Entertainment & Recreation
COM9	Theaters
COM10	Parking Garages
EDU1	Grade Schools and Admin. Offices
EDU2	Colleges and Universities
GOV1	Government - General Services
GOV2	Government - Emergency Response
IND1	Heavy Industrial
IND2	Light Industrial
IND3	Food/Drugs/Chemicals
IND4	Metals/Minerals Processing
IND5	High Technology
IND6	Construction Facilities and Offices
REL1	Churches and Non-Profit Organizations

RES1	Single Family Dwellings
RES2	Manufactured Housing
RES3A	Duplex
RES3B	3 to 4 Units
RES3C	5 to 9 Units
RES3D	10 to 19 Units
RES3E	20 to 49 Units
RES3F	At lease 50 Units
RES4	Temporary Lodging
RES5	Institutional Dormitories
RES6	Nursing Homes
UNK	Unknown

NIBS (2002) and California Governor's Office of Emergency Services (2004).

Table 3I. Event (Earthquake) Information

CAUTION: There are two forms of ***Subscript***ing for most ***Event*** tags, each with a distinct, specific use and purpose:

Subscripting certainly may be used to show ***Event.…*** solutions from multiple institutions (which we recommend), or to show a history of solutions from one institution (which we do not recommended). The form of that ***Subscript***ing is "***Event.Hypocenter(n). …***".

Similarly, a ***Subscript*** may be used when a record contains the traces of more than one ***Event***, as may happen in very active aftershock and swarm sequences. This form of that ***Subscript***ing, in contrast, is "***Event(m)….***". This is a form of ***Subscript***ing that should never be used for any other purpose or confusion may result about what information goes with what event on a multiple-***Event*** record.

If any of the ***Event.Hypocenter.…*** tags are given, then all marked CONDITIONAL are REQUIRED:

```
Event[(m)].CommonName_txt = “<value>”;                                  || Optional
            || A common name for this earthquake, such as “1994 Northridge”.
Event[(m)].Hypocenter[(n)].Agency_txt = “<value>”;                      || CONDITIONAL
            || Originating provider (Agency from table 3E). This Agency
            || applies to the next 11 tags. If an Event.Hypocenter is
            || provided, this tag is REQUIRED.
Event[(m)].Hypocenter[(n)].Identifier_txt = “<Agency's Identifier>”;    || Recommended
            || (Unique) identifying string issued by the Agency
            || issuing the Event information. for example, “nc51153328”,
            || or “[Test] Record of ...”

Event[(m)].Hypocenter[(n)].DateTime_txt =
                              “YYYY-MM-DD hh:mm:ss.s[ss][Z|[+|-]hh[:mm]]”;  || Recommended
            || DateTime is either the date of computation or the date
            || these data were obtained from the Agency. It is not the
            || origin time of the Event, which is the next tag:

Event[(m)].Hypocenter[(n)].OriginTime.DateTime_txt =
                              “YYYY-MM-DD hh:mm:ss.s[ss][Z|[+|-]hh[:mm]]”;  || CONDITIONAL
            || ISO DateTime of Event origin time. Do not use
            || Event.Hypocenter.DateTime for origin times.
Event[(m)].Hypocenter[(n)].OriginTime.TimeError_dbl = <value> s;        || Optional
            || Estimated two-sigma error

Event[(m)].Hypocenter[(n)].Latitude_dbl = <value> deg;                  || CONDITIONAL
            || Positive North, decimal degrees
Event[(m)].Hypocenter[(n)].Longitude_dbl = <value> deg;                 || CONDITIONAL
            || Positive East, decimal degrees
Event[(m)].Hypocenter[(n)].Depth_dbl = <value> <units>;                 || CONDITIONAL
            || Depth below datum (km preferred)

Event[(m)].Hypocenter[(n)].HorizontalDatum_txt = “<from table 3G>”;     || Optional
            || “Horizontal” or “Mixed” datum (rarely known)
Event[(m)].Hypocenter[(n)].VerticalDatum_txt = “<value table 3G>”;     || Optional
            || “Vertical” or “Mixed” datum (rarely known)
```

```
Event[(m)].Hypocenter[(n)].HorizontalError_dbl = <value> <units>;                || Optional
Event[(m)].Hypocenter[(n)].VerticalError_dbl = <value> <units>;                  || Optional
            || (km is the preferred for <units>)
            ||
            || We note the possibility of representing a full uncertainty
            || ellipsoid and seek comment on this matter. For now, these
            || tags describe only an error spheroid.

Event[(m)].Hypocenter[(n)].URL_txt = "<Web address>";                            || Optional
Event[(m)].Hypocenter[(n)].Citation_txt = "<Citation>";                          || Optional
Event[(m)].Hypocenter[(n)].Comments_txt = "<Comments>";                          || Optional
            || Any Comments or Web site URL regarding these Event data.
            ||
            || If in Comments you need more than one line of text for a
            || particular (n), put them all into one, long string with
            || "lines" separated by the character string "/\n/".
```

*Magnitude*s, moment tensors, focal solutions, and ShakeMap tags may or may not be from the same *Agency* as the rest of the *Event* information, so those *Agencies* must be identified separately.

```
Event[(m)].Magnitude[(n)].Type_txt =
                  [ "Mw" | "MS" | "mb" | "MD" | "ML" | "Me" | "Mn" | … ];     || CONDITIONAL
Event[(m)].Magnitude[(n)].Value_dbl = <value>;                                   || CONDITIONAL
Event[(m)].Magnitude[(n)].ValueError_dbl = "<value>";                            || Optional
Event[(m)].Magnitude[(n)].Agency_txt = "<from table 3E(b)>";                     || CONDITIONAL
Event[(m)].Magnitude[(n)].DateTime_txt = "YYYY-MM-DD[ …]";                       || CONDITIONAL
Event[(m)].Magnitude[(n)].NumberOfStationsUsed_int = <value>;                    || Optional
Event[(m)].Magnitude[(n)].URL_txt = "<Web address>";                             || Optional
Event[(m)].Magnitude[(n)].Citation_txt = "<Citation>";                           || Optional
Event[(m)].Magnitude[(n)].Comments_txt = "<Comments>";                           || Optional
            || The CONDITIONAL Event.Magnitude tags above are all REQUIRED
            || if any of them is given.
            ||
            || ValueError is optional and is the accuracy of Value. (The
            || DateTime is either the date (± time) when the Agency computed
            || this magnitude, or the date (± time) when these data were
            || obtained from the named Agency. Omit if unavailable.)
```

||
|| The strings **“Mw”**, **“MS”**, **“mb”**, **“MD”**, **“ML”**, **“Me”**, and **“Mn”**
|| mean, respectively: the Moment ***Magnitude*** (*Hanks and Kanamori*, 1979);
|| the teleseismic 20-s surface-wave ***Magnitude*** (*Gutenberg and
|| Richter*, 1956); the teleseismic body-wave ***Magnitude*** (IASPEI,
|| 2005, formula); the local-***Event*** coda-duration ***Magnitude***; any
|| equivalent of the Wood-Anderson local-***Event*** Richter ***Magnitude***
|| (*Richter*, 1935); the energy ***Magnitude*** (*Choy and Boatwright*,
|| 1995); and the Nuttli ***Magnitude*** (*Nuttli*, 1963).
||
|| This is a USER-EXTENSIBLE LIST. Any user-defined magnitude
|| name must begin with **“User’s own: ”** and end with the short
|| name of the user’s magnitude method. For example, one could
|| name a magnitude “Mx” and thus indicate **“User’s own: Mx”.**
||
|| ***Event.Magnitude.NumerOfStationsUsed*** is the number of stations
|| contributing to the computation of this ***Magnitude.***
||
|| ***URL*** is Recommended for alternately defined magnitudes, “Other”,
|| and for any “Mw” defined in a manner significantly different
|| from that used by the USGS. The USGS equation is an IASPEI-
|| recommended modification of the method of *Hanks and Kanamori*
|| (1979; equation (7)) (which rearrangement makes a difference
|| of only about 0.03 units):
||
|| Mw=(2/3)(log(Mo)-16.1),
||
|| where Mo is in units of dyne-cm. Since there is such a small
|| difference from the original Hanks and Kanamori equation (7),
|| that equation also may be used for “Mw” without giving a ***URL***:
||
|| Mw=(2/3)log(Mo)-10.7.
||
|| In any case, this ***URL*** should point to a complete definition
|| of the magnitude indicated as “Other” or “Mw”, including the
|| computational method used.
||

```
            || If, in Comments, you need more than one line of text
            || for a particular Event(m), or a particular solution for
            || Magnitude(n), put it all into one string with "lines"
            || separated by the character string "/\n/".
```

Note that ***Moment*** is different from a Moment ***Magnitude***, though they are related, of course.

```
Event[(m)].Moment.Value_dbl = <value> < Nm | dyne-cm >;                  || Optional
Event[(m)].Moment.Agency_txt = "<table 3E>";                             || CONDITIONAL
Event[(m)].Moment.DateTime_txt = "YYYY-MM-DD[ hh:mm:ss[Z|[+|-]hh[:mm]]]"; || CONDITIONAL
Event[(m)].Moment.URL_txt = "<Web address>";                             || Optional
Event[(m)].Moment.Citation_txt = "<Citation>";                           || Optional
Event[(m)].Moment.Comments_txt = "<Comments>";                           || Optional
            || DateTime is either the date (± time) of computation or
            || when these data were obtained from the named Agency.
            || In some cases, the Moment will need an explanation,
            || Comments. URL points to some reference.

Event[(m)].FaultRupture[(n)].Length_dbl = <value> <units>;               || Optional
Event[(m)].FaultRupture[(n)].Width_dbl = <value> <units>;                || Optional
Event[(m)].FaultRupture[(n)].SurfaceArea_dbl = <value> <units>;          || Optional
Event[(m)].FaultRupture[(n)].MeanSlip_dbl = <value> <units>;             || Optional
Event[(m)].FaultRupture[(n)].BestSlip_dbl = <value> <units>;             || Optional
Event[(m)].FaultRupture[(n)].RiseTime_dbl = <value> s;                   || Optional
Event[(m)].FaultRupture[(n)].SlipVelocity_dbl = <value> <units>;         || Optional
Event[(m)].FaultRupture[(n)].StaticStressDrop_dbl = <value> <units>;     || Optional
Event[(m)].FaultRupture[(n)].Mach_dbl = <value> <units>;                 || Optional
Event[(m)].FaultRupture[(n)].ShallowestAsperityDepth_dbl = <value> <units>; || Optional
Event[(m)].FaultRupture[(n)].FaultName[(i)]_txt = "<name>";              || Optional

Event[(m)].FaultRupture[(n)].Agency_txt = "<table 3E>";                  || CONDITIONAL
Event[(m)].FaultRupture[(n)].DateTime_txt =
                                "YYYY-MM-DD[ hh:mm:ss[Z|[+|-]hh[:mm]]]"; || CONDITIONAL
```

```
Event[(m)].FaultRupture[(n)].URL_txt = "<Web address>";                    || Optional
Event[(m)].FaultRupture[(n)].Citation_txt = "<Citation>";                  || Optional
Event[(m)].FaultRupture[(n)].Comments_txt = "<Comments>";                  || Optional
            || Various measures or models of average rupture behavior for
            || rupture model (n).
            ||
            || MeanSlip might be estimated from Moment.Value in units of
            || dyne-cm and FaultRupture.SurfaceArea in units of km^2 as
            ||
            ||      MeanSlip = Moment.Value / (3.58*10^11 * SurfaceArea * 10^10).
            ||
            || Similarly, StaticStressDrop in bars might be estimated from
            || Moment.Value and SurfaceArea as
            ||
            ||      (7/16) * Moment.Value / (SurfaceArea * 10^10 / π)^1.5 / 10^6.
            ||
            || Mach is the ratio of rupture velocity to S-wave velocity.
            ||
            || For example, FaultName might be "San Andreas fault". It is
            || the fault on which the rupture is believed to have occurred.
            || The (i) is for situations like the 1992 Landers earthquake,
            || where the rupture subsumes faults with several names.
```

If an ***Event.MomentTensor*** is specified, all the CONDITIONAL tags below are REQUIRED:

```
Event[(m)].MomentTensor[(n)].Agency_txt = "<table 3E>";                    || CONDITIONAL
Event[(m)].MomentTensor[(n)].DateTime_txt =
                                "YYYY-MM-DD[ hh:mm:ss[Z|[+|-]hh[:mm]]]";   || CONDITIONAL
            || DateTime is either the date (± time) of computation or
            || when these data were obtained from the named Agency.
```

Event[(m)].MomentTensor[(n)].CoordinateSystem_txt =
<"[U|D][N|S][E|W]" in (x,y,z) order>; || CONDITIONAL

|| These are the directions of the positive axes. The most common
|| coordinate system is that of *Aki and Richards* (1980), that is,
|| ***CoordinateSystem*** "NED".
||
|| However, Harvard moment tensors are ***CoordinateSystem*** "USE".

Event[(m)].MomentTensor[(n)].M(i)_cpx = <value> [Nm | dyne-cm]; || CONDITIONAL

|| These are the *six* values of a symmetric moment tensor in
|| Newton-meters or dyne-centimeters (n=1,2,3,4,5,6). They
|| are given as a vector in this order only:
||
|| -   -
		1 2 3
		4 5
		6
	- -
	Again, while the above is the most common in numerical
	processing, Harvard does it differently:
	- -
		1 6 5
		2 4
		3
	- -

```
            || Therefore, the Agency producing this VTF file is responsible
            || for rearranging the Harvard (or any other atypical) order into
            || the order listed first above when storing such moment tensors
            || in a VTF file.

Event[(m)].MomentTensor[(n)].URL_txt = "<Web address>";                    || Optional
Event[(m)].MomentTensor[(n)].Citation_txt = "<Citation>";                  || Optional
Event[(m)].MomentTensor[(n)].Comments_txt = "<Comments>";                  || Optional
            || If you need more than one line of text in Comments,
            || use one long string with the "lines" separated by
            || the character string "/\n/".


Event[(m)].FocalSolution[(n)].Agency_txt = "<table 3E>";                   || CONDITIONAL
Event[(m)].FocalSolution[(n)].DateTime_txt =
                                  "YYYY-MM-DD[ hh:mm:ss [Z|[+|-]hh[:mm]]]"; || CONDITIONAL
Event[(m)].FocalSolution[(n)].Strike_dbl = <value> deg;                    || CONDITIONAL
Event[(m)].FocalSolution[(n)].Dip_dbl = <value> deg;                       || CONDITIONAL
Event[(m)].FocalSolution[(n)].Rake_dbl = <value> deg;                      || CONDITIONAL

Event[(m)].FocalSolution[(n)].Mechanism_txt = "<Use NGA definitions>";     || Optional
Event[(m)].FocalSolution[(n)].PaxisPlunge_dbl = <value> deg;               || Optional
Event[(m)].FocalSolution[(n)].PaxisTrend_dbl = <value> deg;                || Optional
Event[(m)].FocalSolution[(n)].TaxisPlunge_dbl = <value> deg;               || Optional
Event[(m)].FocalSolution[(n)].TaxisTrend_dbl = <value> deg;                || Optional

Event[(m)].FocalSolution[(n)].URL_txt = "<Web address>";                   || Optional
Event[(m)].FocalSolution[(n)].Citation_txt = "<Citation>";                 || Optional
Event[(m)].FocalSolution[(n)].Comments_txt = "<Comments>";                 || Optional
            || All CONDITIONAL FocalSolution tags are REQUIRED if any of
            || them is given. As a group they are Optional as are those
            || labeled Optional.
            ||
            || DateTime is either the date (± time) of computation or when
            || these data were obtained from the named Agency.
            ||
```

|| ***Strike*** is 0-360 degrees clockwise from North, ***Dip*** is 0-90°
|| down to the right when facing in the ***Strike*** direction (thus,
|| the hanging wall is on the viewer's right), and ***Rake*** is
|| rupture motion 0-360 degrees counterclockwise from the
|| ***Strike*** direction when viewed from the hanging wall side).
||
|| These are the same definitions used by *Aki and Richards*
|| (1980), p.106.
||
|| ***MeanSlip*** is the average amount of slip over the entire
|| rupture surface, for example in cm.
||
|| ***Mechanism*** is the NGA-style summary of the faulting mechanism:
||

Value	Rake Angles
"Strike Slip"	-180 < ***Rake*** < -150
	-30 < ***Rake*** < 30
	150 < ***Rake*** < 180
"Normal"	-120 < ***Rake*** < -60
"Reverse"	60 < ***Rake*** < 120
"Reverse-Oblique"	30 < ***Rake*** < 60
	120 < ***Rake*** < 150
"Normal-Oblique"	-150 < ***Rake*** < -120
	-60 < ***Rake*** < -30

||
|| The four tags ***PaxisPlunge***, ***PaxisTrend***, ***TaxisPlunge***, and
|| ***TaxisTrend*** may be used to describe inferred orientations
|| of the maximum, "P", and minimum, "T", axes of the
|| compressive principal stresses.
||
|| For ***Comments***, If multiple "lines" of text are needed for a
|| particular ***(n)***, join them into a single text string, with
|| the line breaks indicated by the character string "***|\n|***".

```
Event[(m)].ShakeMap[(n)].Agency_txt = "<table 3E>";                                  || CONDITIONAL
Event[(m)].ShakeMap[(n)].DateTime_txt = "YYYY-MM-DD[ hh:mm:ss [Z|[+|-]hh[:mm]]]";   || CONDITIONAL
Event[(m)].ShakeMap[(n)].URL_txt = "<Web address>";                                  || CONDITIONAL
Event[(m)].ShakeMap[(n)].Citation_txt = "<Citation>";                                || Optional
            || If one of these tags is given, the first three tags are
            || Recommended. As a group they are Optional.
            ||
            || DateTime is either the date (± time) of computation or when
            || these data were obtained from the named Agency. URL is the
            || Web address of the ShakeMap for this Event. Agency is the
            || entity that computed the ShakeMap.
Event[(m)].ShakeMap[(n)].Comments_txt = "<Comments>";                                || Optional
            || If multiple "lines" of text are needed for a particular
            || (n), join them into a single text string, with the line
            || breaks indicated by the character string "|\n|".
```

In the following, where one of the CONDITIONAL tags is present for a given rupture segment *(i)*, all other such CONDITIONAL tags are REQUIRED for that *(i)*:

```
Event[(m)].DetailedFaultRupture[(n)].Segment[(i)].Latitude_dbl = <value> deg;        || CONDITIONAL
Event[(m)].DetailedFaultRupture[(n)].Segment[(i)].Longitude_dbl = <value> deg;       || CONDITIONAL
Event[(m)].DetailedFaultRupture[(n)].Segment[(i)].Depth_dbl = <value> km;            || CONDITIONAL
Event[(m)].DetailedFaultRupture[(n)].Segment[(i)].Length_dbl = <value> <units>;     || CONDITIONAL
Event[(m)].DetailedFaultRupture[(n)].Segment[(i)].Width_dbl = <value> <units>;      || CONDITIONAL
Event[(m)].DetailedFaultRupture[(n)].Segment[(i)].Strike_dbl = <value> deg;         || CONDITIONAL
Event[(m)].DetailedFaultRupture[(n)].Segment[(i)].Dip_dbl = <value> deg;            || CONDITIONAL
Event[(m)].DetailedFaultRupture[(n)].Segment[(i)].Rake_dbl = <value> deg;           || Optional
Event[(m)].DetailedFaultRupture[(n)].Segment[(i)].SurfaceArea_dbl = <value> <units>;|| Optional
            || These tags form vectors describing the geometry and the
            || sense of motion on each rupture-surface segment (i) in a
            || particular model (n) of a particular Event solution (m).
            ||
            || The Strike, Dip, and Rake are as defined (Aki and Richards,
            || 1980) as in Event.FocalSolution.
            ||
```

```
            || Latitude, Longitude, and Depth are of the "upper left" corner
            || of the segment, as viewed from the hanging-wall side. Depth is
            || relative to mean sea level.
            ||
            || SurfaceArea is the surface area of that Segment(i) of the rupture
            || model. Of course, this can be computed from Length and Width, so
            || if both are given they must be consistent.

Event[(m)].DetailedFaultRupture[(n)].Segment[(i)].MeanSlip_dbl = <value> <units>;   || Optional
Event[(m)].DetailedFaultRupture[(n)].Segment[(i)].BestSlip_dbl = <value> <units>;   || Optional
Event[(m)].DetailedFaultRupture[(n)].Segment[(i)].RiseTime_dbl = <value> s;         || Optional
Event[(m)].DetailedFaultRupture[(n)].Segment[(i)].SlipVelocity_dbl =
                                                              <value> <units>;  || Optional
Event[(m)].DetailedFaultRupture[(n)].Segment[(i)].StaticStressDrop_dbl =
                                                              <value> <units>;  || Optional
Event[(m)].DetailedFaultRupture[(n)].Segment[(i)].Mach_dbl = <value> <units>;       || Optional
Event[(m)].DetailedFaultRupture[(n)].Segment[(i)].ShallowestAsperityDepth_dbl =
                                                              <value> <units>;  || Recommended
            || Details of slip on Segment(i). MeanSlip is the mean slip on
            || that segment of the rupture surface, for example in meters.
            || Mach is the ratio of rupture velocity to S-wave velocity.
            || ShallowestAsperityDepth (depth to the top of the shallowest
            || asperity) is sometimes needed for ground-motion models.

Event[(m)].DetailedFaultRupture[(n)].Agency_txt = "<table 3E>";                     || CONDITIONAL
Event[(m)].DetailedFaultRupture[(n)].DateTime_txt =
                                  "YYYY-MM-DD[ hh:mm:ss [Z|[+|-]hh[:mm]]]";       || Optional
            || The above is a detailed rupture model (n) supplied by
            || this Agency on this date (± time), where DateTime is
            || either the model creation date or the date when it
            || was provided for this VTF file):
Event[(m)].DetailedFaultRupture[(n)].URL_txt = "<Web address>";                     || Optional
Event[(m)].DetailedFaultRupture[(n)].Citation_txt = "<Citation>";                   || Optional
Event[(m)].DetailedFaultRupture[(n)].Comments_txt = "<Comments>";                   || Optional
            || URL, Citation, and Comments, can be used to point to or
            || provide a more detailed description and/or attribution
            || of the DetailedFaultRupture(n) model.
            ||
```

```
|| To be clear: (m) is a particular Event, the (n) above
|| indicates a particular model while (i) above indicates
|| a particular rupture segment within that model. (There
|| will often be an (i), sometimes an (n), but only rarely
|| an (m) which would indicate one of several Events
|| recorded in this single DataSeries.)
```

Table 3J(a). Data-Acquisition Unit (DAU; Recorder; Data Logger) Information

"***DAU***" means "Data Acquisition Unit", the recording, timing, and telemetry system. It does not include the ***Sensor***. (Tat is, a "DAS" is a ***DAU*** plus ***Sensor***s in ANSS parlance.)

Where not otherwise indicated, provide values for the particular recording stream issuing the time series leading to this ***DataSeries***.

```
DAU.Model_txt = "<value table 3J(b)>";                                  || REQUIRED
DAU.Manufacturer_txt = "<value table 3J(b)>";                           || REQUIRED
DAU.SerialNumber_txt = "<value>";                                       || REQUIRED
            || For the DAU capturing the time series leading to
            || this DataSeries. All these values are given by
            || the DAU manufacturer.
DAU.StorageMedium_txt = [ "Film" | "Photo Paper" | "Floppy" | "Tape" | …
                  "Cassette" | "Hard Disk" | "Flash" | "CD" | "DVD" | … ];   || Optional
            || Select one of those media listed whenever applicable.
            ||
            || Others types will emerge as technology advances, thus, this
            || is a USER-EXTENSIBLE LIST. If none of the media listed above
            || applies, please invent some new, terse, very clear, English
            || name (which must begin with "User's description: ").
```

```
DAU.TotalChannels_int = <value>;                                          || Recommended
            || Total number of channels in this DAU (cf.,
            || Array.TotalChannels, Array.TotalRecorders,
            || Sensor.ArrayChannel, and Sensor.DAUchannel).
DAU.TotalChannelsRecorded_int = <value>;                                  || Optional
            || Total number of channels in this DAU.

DAU.ChannelGain_dbl = <value> dB;                                         || REQUIRED
            || The total gain applied between the output of the Sensor and
            || the input of the ADC (as decibels, so that zero means "no
            || gain applied", "unity gain").
DAU.AntiAliasFilter[(m)].Corner_dbl = <value> Hz;                         || CONDITIONAL
            || Analog or digital anti-alias (low pass) filter –3 dB point.
DAU.AntiAliasFilter[(m)].Decay_dbl = <value> dB/octave;                   || CONDITIONAL
DAU.AntiAliasFilter[(m)].Comments_txt = "<value>";                        || Optional
            || Generally, the analog anti-alias filter roll-off corner and
            || decay rate, but sometimes also a digital downsampling filter
            || (usually part of a "delta-sigma" ADC but sometimes computed
            || in a signal-processing unit to create multiple "streams" of
            || recorded data taken at differing sample rates).
            ||
            || The two CONDITIONAL tags are REQUIRED unless (as we advise)
            || the anti-alias filter(s) are subsumed as a parts of the
            || DAU.TransferFunction. If they can be subsumed in this way,
            || then these tags are Avoid.
            ||
            || Both the analog anti-alias filter and any digital "anti-
            || alias"  downsampling filter(s) properly are part of and
            || should be described as part of (often "most of") the
            || DAU.TransferFunction rather than being separated out and
            || called an AntiAliasFilter. These AntiAliasFilter tags
            || are provided as alternative, lesser descriptions for
            || use only when information is scarce.
            ||
```

|| Of course, if such a description is truly unavailable (it will
|| almost always be available from the ADC manufacturer), then
|| the tags ***AntiAliasFilter*** may be used instead. However, given
|| the complex nature of such ADC filters (typically defined by
|| "sync" or FIR filter functions) this is a poor substitute
|| for the full ***DAU.TransferFunction*** description—we do not
|| recommend using ***Decay*** and ***Corner*** to describe any anti-alias
|| filter, particularly not an ADC downsampling filter. If these,
|| truly must be used, then use the tag ***AntiAliasFilter.Comments***
|| to describe the digital downsampling filters more completely.
||
|| CAUTION: Unless the ***Sensor***'s bandwidth and all sources of
|| noise (such as EMI) exclude the possibility of energy above
|| half the internal sampling rate of a delta-sigma ADC, it still
|| will be necessary for the ***DAU*** to have an analog anti-alias
|| filter. There is some misunderstanding of this fact among
|| at least seismologists, who sometimes state that delta-sigma
|| ADCs do not require "external" (that is, analog) anti-alias
|| filtering. This claim is false.

The transfer function of the ***DAU*** is best described by poles and zeroes, or by their polynomial equivalents, ideally in the Laplace (*S*) plane but alternately in the *z*-plane. These are the preferred forms.

If little is known about the ***TransferFunction***, you may use the ***DAU.AntiAliasFilter.Corner***, ***Comments***, and ***DAU.ChannelGain*** tags for at least a general description of the ***DAU***. Remember to include ***Comments*** when the ADC is a delta-sigma type, that is, contains a digital downsampling filter, since those alternative descriptions typically will not describe such filters accurately.

DAU.TransferFunction.FilterName and one or the other set of tags below are jointly REQUIRED to the degree of a sufficient description whenever any is given. The ***Constant***s may or may not be needed, They default to a real value of one (1) if not provided (and the imaginary part will default to 0*j* of omitted, leaving a real).

Detailed definitions of these ***Filter*** and ***TransferFunction*** tags is to be found in the comments describing ***Processing.Filter***s.

```
DAU.TransferFunction.FilterName_txt = "<from table 3O(a)>";                    || CONDITIONAL
            || Preferably one of the following:
            ||     "S-plane Poles/Zeros"
            ||     "z-plane Poles/Zeros"
            ||     "S-plane Polynomial"
            ||     "z-plane Polynomial"
            ||     "Other Transfer Function"
            || as in table 3O(a). For less complete or general
            || descriptions of a TransferFunction, use the value
            || "Other Transfer Function", as when providing only
            || DAU.AntiAliasFilter and DAU.ChannelGain.

DAU.TransferFunction.Causality_txt = [ "Causal" | "Acausal" | "Zero Phase" ];  || Optional
            || The causality of the DAU.TransferFunction is implicit in
            || either of the following formal descriptions, but may be
            || indicated here if desired. Modern delta-sigma recorders
            || are in all cases lagged "Zero Phase", though this fact
            || matters primarily at high frequencies.
```

Either the set:

```
DAU.TransferFunction.PoleZero.Numerator.NumberOfRoots_int = <0, 1, 2, ...>;    || CONDITIONAL
DAU.TransferFunction.PoleZero.Denominator.NumberOfRoots_int = <0, 1, 2, ...>;  || CONDITIONAL
DAU.TransferFunction.PoleZero.Numerator.Root(m)_cpx = <real> <imaginary>;      || CONDITIONAL
DAU.TransferFunction.PoleZero.Denominator.Root(m)_cpx = <real> <imaginary>;    || CONDITIONAL
DAU.TransferFunction.PoleZero.Numerator.Constant_cpx = <real> <imaginary>;     || CONDITIONAL
DAU.TransferFunction.PoleZero.Denominator.Constant_cpx = <real> <imaginary>;   || CONDITIONAL
```

or the set:

```
DAU.TransferFunction.Polynomial.Numerator.NumberOfCoeff_int = <1, 2, 3, ...>;        || CONDITIONAL
DAU.TransferFunction.Polynomial.Denominator.NumberOfCoeff_int = <1, 2, 3, ...>;      || CONDITIONAL
DAU.TransferFunction.Polynomial.Numerator.Coefficient(m)_cpx = <real> <imaginary>;   || CONDITIONAL
DAU.TransferFunction.Polynomial.Denominator.Coefficient(m)_cpx =
                                                    <real> <imaginary>;      || CONDITIONAL

DAU.CountSize_dbl = <value> <units>;                                                 || REQUIRED
            || Size of one count (that is, the least significant bit) of the
            || DAU analog-to-digital converter. <units> may be V, mV, uV,
            || g_local, mg_local, ug_local, g_standard, mg_standard,
            || ug_standard, cm/s/s, or any other time-series unit that
            || makes sense in this context.
DAU.FullScale_dbl = <value> <units>;                                                 || Recommended
            || Same units as DAU.CountSize but the (two-sided
            || full-scale range of the digitizer. For example, a
            || Quanterra Q330 has 40 V peak-to-peak differential
            || inputs, so <value> would be 40.
DAU.WordLength_int = <value>;                                                        || REQUIRED
            || In bits. For example, and ADC (analog-to-digital converter)
            || recording in two's-compliment integers of length N, the ADC
            || output values will be in the range ±(2^N-1 - 1). So for N=16,
            || the range is ±32767 (a few systems also allow –32768, a
            || slight asymmetry inherent in two's-complement encoding).

DAU.DynamicRange_dbl = <value> dB;                                                   || Recommended
            || Effective system dynamic range, Sensors plus recorder, as
            || defined by ANSS, near 1 Hz or mid-band: Ratio between just-
            || clipping sine wave RMS and noise RMS in one-half-octave bin
            || centered at (about) 1.0 Hz (if that is well inside the pass-
            || band), or otherwise near the center of the pass-band.
```

```
DAU.EffectiveBits_dbl = <value>;                                              || Optional
            || Bits = log2(<total ADC counts> / (2 * <RMS noise counts>)).
            || Equally, the peak-to-peak g's versus the RMS noise g's are
            || reduced to "bits" by a base-2 log. Therefore, let "R" be
            || DAU.DynamicRange expressed as an amplitude ratio, hence
            || R = < peak-to-peak g's > / <root 2> / <noise RMS g's>. Thus,
            || DAU.EffectiveBits = log2(DAU.DynamicRange) - 0.5;  (It would
            || be "-1.0" to account for the double-sidedness of bit-counts
            || versus the single-sidedness of RMS, however, a dynamic range
            || involves a clipping sine wave while bit counts compares the
            || full range of the ADC—another square root of two separates
            || the metrics.)

DAU.CompressionUsed.Type_txt = [ "None" | "Steim" | "Reed-Solomon" | "Wavelet" | …
                           "Cosine" | "Other Lossless" | "Other Lossy" ];     || Optional
DAU.CompressionUsed.Citation_txt = "<Citation>";                              || Optional
            || Any compression algorithm applied to the data prior to its
            || storage or transmission by the DAU. The Type should be
            || supplied. At least in cases of "Other …" the Citation also
            || should be given.

DAU.Comments_txt = "<Comments>";                                              || Optional
            || Any comments about the DAU or its operation (for
            || items not covered by defined DAU tags). If multiple
            || "lines" of text are needed, join them into a single
            || text string, with the line breaks indicated by the
            || character string "/\n/".
```

NOTE: Pre-***Event*** and post-***Event*** durations have been moved to Record data, since they may vary by record. Film digitizer Y-step has moved similarly.

Table 3J(b). DAU Models and Manufacturers

Model Name	Manufacturer
Analog Recorders:	
"USCGS Standard"	U.S. Coast and Geodetic Survey
"AR-240"	Teledyne
"RFT-250"	
"RFT-350"	
"MO-2"	(New Zealand)
"MO2A"	
"RMT-280"	Teledyne
"SMA-1"	Kinemetrics
"SMA-2"	
"SMA-3"	
"CRA-1"	

Digital Recorders:	
“QDR”	Kinemetrics
“DSA-1”	
“DSA-3”	
“PDR-1”	
“PDR-2”	
“SSA-1”	
“SSA-2”	
“SSA-16”	
“SSR-1”	
“K2”	
“Etna”	
“Mt Whitney”	
“Everest”	
“DR-100”	Sprengnether
“DR-200”	
“DR-300”	
“DR-3016”	
“DR-3024”	
“DCA-300”	Terratech
“DCA-310”	
“DCA-333”	
“DCA-333R”	
“GSR-12”	GeoSIG (Terratech a sometime reseller)
“GSR-18”	
“IDS-3602”	Terratech (IDS)
“IDS-3602A”	Terratech (IDSA)

“A700”	Geotech
“A800”	
“A900”	
“A900A”	
“GEOS”	U.S. Geological Survey
“DST”	
“Earthworm”	
“TREMOR”	
“Q4120”	Quanterra
“Q4128a”	
“Q730”	
“Q736”	
“Q980”	
“72A”	RefTek
“130-ANSS”	
“130-SM”	
“VSE-355G3”	Tokyo Sokushin
Miscellaneous:	
“Other Analog”	Unknown; legacy
“Other Digital”	Unknown; legacy
“_____”	USER-EXTENSIBLE LIST (use established, terse, clear, English names which must begin with **“User’s description: ”**)

Manufacturers of *DAU*s and of *Sensor*s:	
"Akashi"	
"Canadian Geological Survey"	
"GeoSIG"	
"Geotech"	
"Guralp"	
"Kinemetrics"	
"Lennartz"	
"Mark Products"	
"(New Zealand)"	
"Quanterra"	
"RefTek"	
"Sprengnether"	
"Streckeisen"	
"Teledyne"	
"Terratech"	
"Terratech (IDS)"	
"Terratech (IDSA)"	
"Tokyo Sokushin"	
"USCGS"	
"U.S. Geological Survey"	
"Wilcoxon"	
"_____"	USER-EXTENSIBLE LIST (use established, terse, clear, English names, but not necessarily the full, official corporate name as long as it is quite clear. Any user-defined manufacturer name must begin with **"User's description: "**)

Table 3K(a). Sensor Information

```
Sensor.Model_txt = "<value table 3K(b)>";                                  || REQUIRED
Sensor.Manufacturer_txt = "<value table 3J(b)>";                           || REQUIRED
Sensor.SerialNumber_txt = "<value>";                                       || REQUIRED
            || For the Sensor capturing the time series leading to
            || this DataSeries. All these values are given by the
            || Sensor manufacturer.
```

The orientation in space of a vector (the ***Sensor***'s positive active axis) is always uniquely and fully representable in the three-space on or near a planet's surface by two numbers, inclination and azimuth (the third element of this vector in effect represented by the ***GeoLocatio.Location*** and the figure of the planet).

This two-element vector is the preferred description of orientation.

Inclination is always given relative to plumb (Earth acceleration vector) from 0 to 180° with zero meaning positive output for **upward ground motion**. (Absolute) ***Azimuth*** is always represented Eastward from North, ranging between 0 and 359.999…, with zero being North. Azimuth also may be represented relative to some stated **Array** (reference) ***Azimuth*** (in the same format as an absolute ***Azimuth***). This ***RelativeAzimuth*** is always given clockwise from the **Array** (reference) **Azimuth**, and is often the best option for an ***Array***, such as in a structure.

Absolute orientation:

```
Sensor.Azimuth.Value_dbl = <value> deg;                                  || CONDITIONAL
            || Sensor azimuth, clockwise from true geographic North (always
            || give zero "Azimuth" for an Inclination of 0 or 180°).
            ||
            || REQUIRED unless Sensor.RelativeAzimuth.Value and also
            || Array.Azimuth.Value are given.
Sensor.Inclination.Value_dbl = <value> deg;                              || REQUIRED
            || Positive up = 0, horizontal = 90, positive down = 180.
```

Relative azimuth:

```
Sensor.RelativeAzimuth.Value_dbl = <value> deg;                          || CONDITIONAL
            || Like Sensor.Azimuth.Value but 0 to 359.999… clockwise
            || from Array.Azimuth.Value rather than from true North.
            || In all cases, this compass direction must still be paired
            || with Sensor.Inclination.Value for a full description.
Array.Azimuth.Value_dbl = <value> deg;                                   || CONDITIONAL
            || Azimuth of reference "North" versus true geographic North,
            || following the same rules as for Sensor.Azimuth.Value.
            || The absolute Azimuth of the Sensor is the (modulo 360) sum
            || of Sensor.RelativeAzimuth.Value and Array.Azimuth.Value
            ||
            || So this means, for example, that a Sensor oriented to cardinal
            || northeast but represented relative to
            ||
            ||      Array.Azimuth.Value_dbl = 90 deg;
            ||
            || (East) would be given as
            ||
            ||      Sensor.RelativeAzimuth.Value_dbl = 315 deg;
            ||
```

```
            || notwithstanding that it also could have been represented in
            || absolute terms as
            ||
            ||      Sensor.Azimuth.Value_dbl  = 45 deg;
            ||
            || Sensor.RelativeAzimuth.Value and Array.Azimuth.Value
            || are jointly REQUIRED if Sensor.Azimuth.Value is not given.

Sensor.Azimuth.ValueError_dbl = <value> deg;                          || Optional
            || Two-sigma error estimate for absolute
Sensor.Inclination.ValueError_dbl = <value> deg;                      || Optional
            || Two-sigma error estimate for absolute
Sensor.RelativeAzimuth.ValueError_dbl = <value> deg;                  || Optional
            || Two-sigma error estimate for relative
Array.Azimuth.ValueError_dbl = <value> deg;                           || Optional
            || Two-sigma error estimate for reference
```

Alternate description:

```
Sensor.Orientation_txt = "<from table 3K(c)>";                        || Avoid
            || This tag is supplied only for backward compatibility
            || or, possibly, for preliminary use—its application
            || is discouraged.
```

Of the orientation tags above, ***Sensor.Azimuth.Value*** and ***Sensor.Inclination.Value*** are the preferred set.

However, in many ***Array***s the azimuths are given relative to a structure's orientation, or to some other reference azimuth, and in this case use the tag set ***Sensor.RelativeAzimuth.Value*** plus ***Array.Azimuth.Value***, but by absolute ***Sensor.Inclination.Value*** since floors are more or less level.

Only as a last choice should you ever use ***Sensor.Orientation***. (In any case, note that most of these can be translated into one of the other two representations.)

Associated tags are:

```
Sensor.ArrayChannel_int = <value>;                                  || Recommended
            || Array-channel that this Sensor is on
Sensor.DAUchannel_int = <value>;                                    || Recommended
            || The DAU-channel that this Sensor recorded on (if on
            || film, count down from the top, the first being 1, not 0)
            || (cf. Array.TotalChannels, Array.TotalRecorders,
            || and DAU.TotalChannels)


Sensor.Sensitivity_dbl = <value> <units>;                           || REQUIRED
            || Sensor.Sensitivity is in units, for film of cm/g_standard
            || or cm/g_local, for example. <units> for other systems are
            || V/g_standard, V/g_local, V/(cm/s/s) or V/(<units>), where
            || <units> is one of the other units of acceleration, velocity,
            || displacement, rotation, pressure, ustrain, and so forth.
            || Parentheses are required when <units> is compound,
            || as in the example.
            ||
            || NB: In all cases, this Sensitivity value is to be as measured
            || as close as possible to the raw Sensor—thus, in the case of
            || FBAs and similarly "conditioned" Sensors, at the output of that
            || Sensor system. Sensitivity is to be measured before any DAU
            || amplification and filtering, so well before digitization.

Sensor.FullScaleOut_dbl = <value> <units>;                          || REQUIRED
            || Peak output of the Sensor. The same measurement rules
            || as apply to Sensor.Sensitivity.
Sensor.FullScaleIn_dbl = <value> <units>;                           || Recommended
            || Peak unclipped Earth signal in <units> apropos to that
            || Sensor; generally g_local, g_standard, or cm/s/s for
            || accelerometers. The same measurement rules apply as
            || for Sensor.Sensitivity.
```

```
Sensor.SensitivityContext_txt = "<where measured>";                    || Recommended
            || The context in which Sensor.Sensitivity,
            || Sensor.FullScaleOut, and Sensor.FullScaleIn
            || were measured. For example, "At output of [ FBA |
            || geophone | sensor system ]".

Sensor.Causality_txt = [ "Causal" | "Acausal" ];                       || CONDITIONAL
            || All fully-analog Sensors are by definition "Causal".
            || However, sensors containing digital signal conditioning,
            || likely including those with a digital output, may
            || contain a delta-sigma ADC or some other "Acausal"
            || elements that make the whole "Acausal".
            ||
            || The default is "Causal" since this is most often the case.
            || This tag is REQUIRED if the Sensor (package) is in any
            || way "Acausal" (most commonly for Sensor packages having
            || built-in signal conditioning and digitizing and digital
            || output).
```

The ***Sensor.TransferFunction*** is most completely and generally described by ***PoleZero*** or the equivalent ***Polynomial*** coefficients, preferably in the Laplace- (*S*-)plane, but alternately in the *z*-plane. These are the preferred forms. Alternate, less precise forms are given below those forms.

If ***TransferFunction.FilterName*** is given, then the specified set of tags are jointly REQUIRED, to the extent they form a complete representation of the ***TransferFunction***.

In ***PoleZero*** representations, either the ***Numerator*** or the ***Denominator*** may have no roots in which case there will only be a ***Constant*** for that portion (and this may be omitted if it is equal to a real value of one (1) (the default value of ***Constant***; the imaginary part will default to 0*j* of omitted, leaving a real). The corresponding ***NumberOfRoots*** will be zero, no roots will be listed for that portion. It is common to gather both ***Constant***s into the ***Numerator***, but that is not required.

Polynomial representations, of course, have a constant built in at the zeroth-order ***Coefficient***, and **M+1** ***Coefficient***s are always required (***Subscript***ed from 1 through **M+1**), even if some of these are zeros.

Detailed definitions of these filter classes is to be found among the comments describing ***Processing.Filter***s and in table 3O(a).

```
Sensor.TransferFunction.FilterName_txt = "<from table 3O(a)>";                    || CONDITIONAL
            || One of the following:
            ||     "S-plane Poles/Zeros"
            ||     "z-plane Poles/Zeros"
            ||     "S-plane Polynomial"
            ||     "z-plane Polynomial"
            ||     "Other Transfer Function"
            || For less complete or general methods of describing sensor
            || response, use the latter value along with a "sufficient"
            || set of the tags described after Polynomial.
```

Use either the set:

```
Sensor.TransferFunction.PoleZero.Numerator.NumberOfRoots_int = <0, 1, 2, ...>;     || CONDITIONAL
Sensor.TransferFunction.PoleZero.Denominator.NumberOfRoots_int = <0, 1, 2, ...>;   || CONDITIONAL
Sensor.TransferFunction.PoleZero.Numerator.Root(m)_cpx = <real> <imaginary>;       || CONDITIONAL
Sensor.TransferFunction.PoleZero.Denominator.Root(m)_cpx = <real> <imaginary>;     || CONDITIONAL
Sensor.TransferFunction.PoleZero.Numerator.Constant_cpx = <real> <imaginary>;      || CONDITIONAL
Sensor.TransferFunction.PoleZero.Denominator.Constant_cpx = <real> <imaginary>;    || CONDITIONAL
```

or the set:

```
Sensor.TransferFunction.Polynomial.Numerator.NumberOfCoeff_int = <1, 2, 3, ...>;   || CONDITIONAL
Sensor.TransferFunction.Polynomial.Denominator.NumberOfCoeff_int = <1, 2, 3, ...>; || CONDITIONAL
Sensor.TransferFunction.Polynomial.Numerator.Coefficient(m)_cpx =
                                                    <real> <imaginary>;    || CONDITIONAL
Sensor.TransferFunction.Polynomial.Denominator.Coefficient(m)_cpx =
                                                    <real> <imaginary>;    || CONDITIONAL
```

Less complete or general descriptions of ***Sensor*** behavior:

```
Sensor.NaturalFrequency.Value_dbl = <value> Hz;                    || CONDITIONAL
            || Sensor's natural frequency.
Sensor.Damping.Value_dbl = <value>;                                || CONDITIONAL
            || Sensor's damping (as a fraction of critical).
            ||
            || This pair jointly REQUIRED if either is given.


Sensor.NaturalFrequency.ValueError_dbl = <value> Hz;               || Optional
Sensor.Damping.ValueError_dbl = <value>;                           || Optional
            || Estimated two-sigma errors.

Sensor.HighPass.Corner_dbl = <value> Hz;                           || CONDITIONAL
            || Sensor low-frequency corner (0.0 Hz for many
            || accelerometers)
Sensor.HighPass.Decay_dbl = <value> dB/octave;                     || CONDITIONAL
            || Sensor low-frequency roll-off, if
            || Sensor.HighPass.Corner is not zero.
            ||
            || This pair jointly REQUIRED if either is given.


Sensor.LowPass.Corner_dbl = <value> Hz;                            || CONDITIONAL
            || Sensor high-frequency corner, if present in
            || addition to Sensor.NaturalFrequency.Value.
Sensor.LowPass.Decay_dbl = <value> dB/octave;                      || CONDITIONAL
            || Sensor high-frequency roll-off.
            || This pair jointly REQUIRED if either is given.
```

```
Sensor.Comments_txt = "<Comments>";                                    || Optional
            || Specify which "Other" Sensors and any Comments about the
            || Sensor or its operation not otherwise covered by defined
            || Sensor tags.
            ||
            || If multiple "lines" of text are needed, join them into
            || a single text string, with the line breaks indicated by
            || the character string "/\n/".
```

NOTES:

(1) We realize that some of the "required" tags will not always be known, particularly for legacy data—these tags still must be present in the VTF file and given the value NULL (no quotes).

(2) "Gain applied to sensor output ..." has moved to ***DAU*** information, because amplifiers and buffers are part of the recorder. In the case of high-output analog ***Sensor***s (for example, force-balance accelerometers), the internal circuitry is considered part of the ***Sensor***.

(3) For ***Sensor*** azimuths, now see: ***Sensor.Azimuth.Value*** or ***Sensor.RelativeAzimuth.Value*** plus ***Array.Azimuth.Value*** or, as a last choice, ***Sensor.Orientation***.

(4) For location offsets (in meters, for example) of ***GeoLocations*** in ***Array***s, now see ***GeoLocation.Location.NorthOffset***, ***GeoLocation.Location.EastOffset***, and ***GeoLocation.Loca-tion.ElevationOffset*** in table 3F.

Table 3K(b). Sensor Models (use Table 3J(b) for Manufacturers)

Model name	**Manufacturer**
Accelerometers:	
"Optical-Mechanical" (SMA, RFT,…)	Any
"FBA-1"	Kinemetrics
"FBA-3"	
"FBA-11"	
"FBA-13"	
"FBA-13DH"	
"FBA-23"	
"FBA-23DH"	
"EpiSensor"	
"EpiSensor ES-U"	
"FBX-23"	Sprengnether
"FBX-26"	
"SSA 120"	Terratech
"SSA 220"	
"SSA 320"	
"731A"	Wilcoxon
"CMG-5"	Guralp
Velocity ***Sensor***s/Seismometers:	
"SS-1 Ranger"	Kinemetrics
"S-3000"	Sprengnether

"CMG-1"	Guralp
"CMG-3T"	
"CMG-3ESP"	
"CMG-40"	
"STS-1"	Streckeisen
"STS-2"	
"L4"	Mark Products
"L22D"	
"SA-102A"	(New Zealand)
Other ***Sensor***s (which may be specified in ***Sensor.Comments***):	
"Other Accelerometer"	Unknown or legacy ***Sensor*** (in preference, please use the User-Extensible option and provide a real model name)
"Other Velocity Seismometer"	
"Pressure Sensor"	Various (in preference, please use the User-extensible option and provide a real model name)
"Dilatometer"	
"Relative Displacement Sensor"	
"Rotational Sensor"	
"Other Sensor"	
"_____"	USER-EXTENSIBLE LIST (use established, terse, clear, English names which must begin with **"User's description: "**)

(It is clear that this table must grow. Please e-mail the authors or use the extensible-list option to communicate suggestions to us.)

Table 3K(c). Sensor Direction Codes for use with *Sensor.Orientation* only (Avoid)

Abbreviation	Orientation*
"1"-"360"	Horizontal azimuth, clockwise (East) from North to the nearest degree (in this case, *it is preferable to use* ***Sensor.Azimuth.Value*** with ***Sensor.Inclination.Value_dbl* = 90 deg;**).
"1001"-"1360"	Horizontal azimuth (deg + 1000) relative to Channel 1, used when the absolute orientation is unknown (for example, Channel 2 is "1090" if it is 90 deg East of Channel 1). In this case *it is preferable to use* ***Array.Azimuth.Value_dbl* = NULL;** which still allows one to express a ***Sensor.RelativeAzimuth.Value***, often with the values 0° and 90° or 0° and 270° to indicate orthogonal components in those relationships.
"Up"	Vertical, ground motion positive up (It is preferable to use ***Sensor.Inclination.Value_dbl* = 0 deg;** and ***Sensor.Azimuth.Value_dbl* = 0 deg;**).
"Down"	Vertical, ground motion positive down (It is preferable to use ***Sensor.Inclination.Value_dbl* = 180 deg;** and ***Sensor.Azimuth.Value_dbl* = 0 deg;**).
"Vertical"	Vertical, sense not known (this is a rare case in which the preferred methods of describing orientation are not adequate).
Specific to geotechnical and other active-source ***Array***s**	
"Radial"	Radial, positive outward from the source. (In these four cases, it is preferable to use ***Sensor.Inclination.Value_dbl* = <value> deg;** and ***Sensor.Azimuth.Value_dbl* = <value> deg;** (or the relative-azimuth equivalent) plus a SNCL name with "HNT" or "HNR" to indicate this relationship to a source.
"AntiRadial"	Radial, positive inward toward the source.
"Transverse"	Transverse, 90° clockwise in map view from the Radial component (to the right as viewed from the source).
"AntiTransverse"	Transverse, 90° clockwise in map view from AntiRadial component (to the left as viewed from the source).

Specific to structures (for example, bridges)**	
800 or "Longitudinal"	Along long axis of structure, positive in the more northerly of the two possible directions. (It is preferable to use the relative-azimuth option in these cases, with ***Sensor.RelativeAzimuth.Value_dbl* = 0 deg;** and ***Array.Azimuth.Value_dbl* = <value> deg;** for the structure and the long axis of the structure indicated by ***Array.Azimuth.Value*.**)
-800 or "AntiLongitudinal"	Along long axis of structure, positive in the more southerly of the two possible directions. In this case, it is better to use ***Sensor.RelativeAzimuth.Value_dbl* = 180 deg;** as above.
900 or "ShortAxis"	Transverse to the structure, 90° clockwise in map view from the Longitudinal component. (It is preferable to do as above, but with ***Sensor.RelativeAzimuth.Value_dbl* = 90 deg;**).
-900 or "AntiShortAxis"	Transverse to the structure, 90° clockwise in map view from the AntiLongitudinal component. (It is preferable to do as above, but with ***Sensor.RelativeAzimuth.Value_dbl* = 270 deg;**).
"H1"	Horizontal ***Sensor***, azimuths are completely unknown. (It is preferable to use ***Sensor.Azimuth.Value_dbl* = NULL;** to indicate complete ignorance.)
"H2"	
"Other Orientation"	Some other orientation (describe in ***Sensor.Comments***) (One can describe any linear or rotational sensor orientation with azimuth and inclination alone—it is much better to use one of the means provided to do so numerically, as described above.)

*The direction specified is the direction of motion of the ground (or the structural element to which the ***Sensor*** is attached) which corresponds to a positive value in the time series (or for film, of an upward motion of the trace).

**Definitions changed from v1.20 for clarity and to follow existing standards.

Table 3L. Data Series (Including Original): Time-Series or Spectrum Information

```
DataSeries.AgencysIdentifier_txt = "<Agency's Identifier>";                    || REQUIRED
            || for example, "ci14138080", "ak00039525", and so forth.
            ||
            || The record identification string used by the authoring
            || Agency to specify this specific DataSeries.
            ||
            || Not to be confused with DataSeriesCOSMOSidentifier, which is
            || the COSMOS VDC's unique identifier for the same DataSeries,
            || used in cross referencing with the GVDC.
            ||
            || Also, please do no confuse with the identification of the
            || "owner" of this record: ThisFile.Preparation.Agency.
            ||
            || AgencysIdentifier identifies only the DataSeries values
            || themselves, from the viewpoint of the DataSeries' creator.

DataSeries.TriggerNumber_int = <value>;                                        || Optional
            || DAU's trigger number or Event counter.
```

Time-series time stamps (part of the abscissa definition for all time series):

```
RawSeries.FirstSampleTime.DateTime_txt =
                              "YYYY-MM-DD hh:mm:ss.sss[Z|[+|-]hh[:mm]]";       || Optional
            || ISO DateTime (preferably to ms) of the first sample
            || in the original file.
```

```
DataSeries.FirstSampleTime.DateTime_txt =
                                "YYYY-MM-DD hh:mm:ss.sss[Z|[+|-]hh[:mm]]";      || CONDITIONAL
            || ISO DateTime (preferably to ms) of the first sample
            || in this file. REQUIRED for all time series,
            || whether or not the abscissa is implicit.
            ||
            || Note that this is the primary time stamp for this
            || file's time series (DataSeries) and is a key datum.
DataSeries.FirstSampleTime.JulianDay_int = <value>;                             || Avoid
            || The "Julian Day" (more correctly, the "Cardinal Day") is the
            || day of the year (from 1 to 366) and corresponds to "MM-DD" in
            || DataSeries.FirstSampleTime.DateTime. This tag is provided
            || for backward compatibility—the cardinal day is easily
            || computed and, therefore, redundant.
            ||
            || Note that there is no provision in our version of the ISO DateTime
            || for using a cardinal day, so "MM" and "DD" are required (but also
            || easily computed from the cardinal day).
```

Information related to time stamps:

```
RawSeries.FirstSampleTime.TimeError_dbl = <value> s;                            || Recommended
            || Estimated or computed two-sigma error of
            || RawSeries.FirstSampleTime.DateTime.
RawSeries.FirstSampleTime.Source_txt = "<from table 3M>";                       || CONDITIONAL
            || REQUIRED if RawSeries.FirstSampleTime.DateTime
            || is present.

DataSeries.FirstSampleTime.CorrectionApplied_dbl = <value> s;                   || CONDITIONAL
            || Correction which has been applied to a recorded first-sample
            || time to arrive at DataSeries.FirstSampleTime.DateTime, the
            || start time of this time series. The sign of this correction
            || is in the sense:
            ||      DataSeries.FirstSampleTime.DateTime =
            ||           <first-sample time on recorder's chronometer>
            ||           + DataSeries.FirstSampleTime.CorrectionApplied
            || Thus, a slow recorder clock produces a positive correction.
```

```
DataSeries.FirstSampleTime.TimeError_dbl = <value> s;                    || Recommended
            || Estimated or computed two-sigma error of
            || DataSeries.FirstSampleTime.DateTime.
DataSeries.FirstSampleTime.Source_txt = "<from table 3M>";               || REQUIRED
```

For spectral abscissa:

```
DataSeries.FirstBinFrequency_dbl = <value> Hz;                           || CONDITIONAL
DataSeries.FirstBinPeriod_dbl = <value> s;                               || CONDITIONAL
            || In the case of evenly sampled spectra, these tags are used in
            || conjunction with DataSeries.FrequencyBin and DataSeries.PeriodBin
            || to define an implicit abscissa for the DataSeries.
            ||
            || If DataSeries.PeriodBin is used as part of an implicit period
            || abscissa, then DataSeries.FirstBinPeriod is required (because
            || a 0-s period bin makes no sense). If DataSeries.FrequencyBin
            || is used as part of an implicit frequency abscissa, then
            || FirstBinFrequency may be used to define the lowest frequency
            || bin. It defaults to 0 Hz, which is apropos most power spectral
            || densities and the DFT of a real time series. In all other
            || cases, FirstBinFrequency  must be given explicitly or the
            || abscissa must be explicit.
```

In the case of an evenly-sampled ***DataSeries*** that does not have an explicit abscissa provided in the ***DataSeries*** listing matrix, exactly one of the following four is REQUIRED, along with ***DataSeries.FirstSampleTime.DateTime***, ***DataSeries.FirstBinFrequency***, or ***DataSeries.FirstBinPeriod***, which together define an implicit abscissa. Note that the units used here must be the same as those declared by ***DataSeries.AbscissaUnits***, one of the first two below for time series and one of the latter two for spectra. The chosen tag below must also make sense in combination with ***DataSeries.FirstSample-Time.DateTime***, ***DataSeries.FirstBinFrequency***, and ***DataSeries.FirstBinPeriod***.

For implicit, evenly spaced time series, the time axis begins at zero seconds and proceeds to later time by intervals declared by ***DataSeries.SamplesPerSecond*** or ***DataSeries.SampleInterval***. The absolute time of the first sample is given by ***DataSeries.FirstSampleTime.DateTime***. In the case of evenly sampled spectra with implicit abscissa, the first bin

always is assumed to be at **0 Hz** or **0 s** (the latter does not appear to us to make sense so is unlikely to be used; the former makes sense for the DFT spectrum of a real-valued time series, which we anticipate will be the most common use of ***FrequencyBin***).

```
DataSeries.SamplesPerSecond_dbl = <value> Hz;                          || CONDITIONAL
            || Sampling rate of time series
DataSeries.SampleInterval_dbl = <value> s;                             || CONDITIONAL
            || Sampling interval of time series
DataSeries.FrequencyBin_dbl = <value> Hz;                              || CONDITIONAL
DataSeries.PeriodBin_dbl = <value> s;                                  || CONDITIONAL
            || Bin spacing in a spectrum. (For PeriodBin, the first bin
            || at "zero period" might apply to PGA supplied with Sa, for
            || example, but will more often make no sense. Its use will
            || be rare.)

RawSeries.SamplesPerSecond_dbl = <value> Hz;                           || Optional
            || Sampling rate of foregoing time series, or ...
RawSeries.SampleInterval_dbl = <value> s;                              || Optional
            || ... sampling interval of foregoing time series, or ...
RawSeries.UnevenSampling_txt = [ "Yes" | "No" ];                       || Optional
            || ... that the DataSeries was unevenly sampled, as for
            || hand-digitized optical records. Default is "No",
            || that is, even sampling.


RawSeries.AbscissaUnits_txt = "<units>";                               || CONDITIONAL
            || Because there is no location in this VTF file for the
            || RawSeries values, this tag/value pair serves only to
            || acknowledge the units of measure of the source time
            || series and to indicate the units in which that rate
            || is to be expressed. For example, in a record taken
            || from film, we might have
            ||     RawSeries.AbscissaUnits_txt = "um";
```

DataSeries.AbscissaUnits is REQUIRED in all cases and is the single authoritative location in which the units of the abscissa are given (notwithstanding that those same units may well be used with related tags, and when so must at least not conflict with this ***RawSeries.AbscissaUnits***). In the case of implicitly defined even sampling intervals, the abscissa values are implied by a combination of ***DataSeries.SamplesPerSecond***, ***SampleInterval***, ***FrequencyBin***, or ***PeriodBin***, plus a starting value, either ***DataSeries.FirstSampleTime.DateTime*** for a time series or ***DataSeries.FirstBin-Frequency*** or ***DataSeries.FirstBinPeriod*** for spectra.

In the case of uneven sampling of the ***DataSeries*** in this VTF file, or of an author's desire to provide an explicit abscissa for other reasons, ***AbscissaFormat*** also is REQUIRED and there must be an additional (first) column (or pair of columns if the abscissa is complex valued) in the main ***DataSeries*** listing matrix.

What one might have expected as the corresponding first entry in ***DataSeries.OrdinateFormat*** is not there, not prescribed by ***OrdinateFormat***, but is instead given here as ***DataSeries.AbscissaFormat***. Thus, the format of an entire row of the ***DataSeries*** listing matrix is a concatenation of ***AbscissaFormat*** with ***OrdinateFormat***.

```
DataSeries.AbscissaUnits_txt = "<units>";                                  || REQUIRED
DataSeries.AbscissaFormat_txt = "<FORTRAN format specification>";          || CONDITIONAL
            || The units (and if provided explicitly, also the format of the
            || abscissa). For explicit abscissas AbscissaFormat will describe
            || the first column (real values) or first and second columns
            || (complex values) of the DataSeries listing matrix, typically
            || used in the case of uneven (irregular) sampling.
            ||
            || If the DataSeries is unevenly sampled or for any other reason,
            || including user preference an explicit abscissa is listed in the
            || DataSeries listing matrix, then both tags are REQUIRED as is/are
            || abscissa column(s) in the DataSeries listing matrix.
            ||
```

|| A single abscissa value is allowed for each row of the ***DataSeries***
|| listing matrix, however, this value can be either real or complex.
|| If it is complex, ***AbscissaFormat*** must have two entries, generally
|| abutting and always given as a single text string containing one
|| format specification for the real and one for the imaginary part
|| of each value. (Note that FORTRAN format-repetition counts are
|| not allowed, so there would have to be two explicit formats for
|| complex values.)

DataSeries.Span_dbl = <value> [s | Hz]; || Optional
|| The time duration or bandwidth of the whole ***DataSeries***.
||
|| Optional because it is computed easily from other values.
|| If given, therefore, it also *must* equal one of the following:
||
|| ***DataSeries.NumberOfSamples*** / ***DataSeries.SamplesPerSecond***
|| ***DataSeries.NumberOfSamples*** * ***DataSeries.SampleInterval***
|| ***DataSeries.NumberOfSamples*** * ***DataSeries.FrequencyBin***
|| ***DataSeries.NumberOfSamples*** * ***DataSeries.PeriodBin***
||
|| and so forth, OR, when there is/are abscissa column(s) in
|| the ***DataSeries*** listing matrix, the difference between the
|| last and first abscissa values.

RawSeries.Span_dbl = <value> s; || Optional
|| Same thing, but for the original record (which
|| must have been a time series, of course).

The following two tags relate to double-integration initial conditions. Also see ***RawSeries.Mean***, ***DataSeries.Mean***, ***Processing.InitialValue.Velocity*** and ***Processing.InitialValue.Displacement*** and/or the ultimate statement of baselines—providing them explicitly with ***DataSeries.PhysicalParameter = "Acceleration Correction Baseline";*** in a separate VTF file.

```
DataSeries.Pad.FirstOfOriginal_int = <value>;                                  || CONDITIONAL
DataSeries.Pad.LastOfOriginal_int = <value>;                                   || CONDITIONAL
            || Both are REQUIRED if the DataSeries retains padding on either
            || or both ends as used in Processing (as is recommended by Boore
            || and Akkar, 2003) and also by us.
            ||
            || These are the (integer) indexes of first and last samples of
            || the unpadded part of the DataSeries (the first sample of the
            || padded trace contained in this VTF file being index 1).
            ||
            || For example, if 10 zeros were added to each end of a 100-point
            || original time series, then we would have
            ||
            ||      DataSeries.NumberOfSamples_int = 120;          || (10 + 100 + 10),
            ||      DataSeries.Pad.FirstOfOriginal_int = 11;       || (10 + 1), and
            ||      DataSeries.Pad.LastOfOriginal_int = 110;       || (10 + 100).
            ||
            || The same would apply if no padding was appended to the rear in this
            || example, except that the trace length would be smaller by 10.
            || However, if no padding is prefixed to the front, then
            || DataSeries.Pad.FirstOfOriginal_int = 1;.


RawSeries.TriggerTime.DateTime_txt = "YYYY-MM-DD hh:mm:ss.sss[Z|[+|-]hh[:mm]]";   || Optional
DataSeries.TriggerTime.DateTime_txt = "YYYY-MM-DD hh:mm:ss.sss[Z|[+|-]hh[:mm]]"; || Optional
            || ISO DateTime (preferably to ms) of the TRIGGER time. Not to be
            || confused with the first-sample times, FirstSampleTime.
RawSeries.TriggerTime.TimeError_dbl = <value> s;                               || Optional
DataSeries.TriggerTime.TimeError_dbl = <value> s;                              || Optional
            || Estimated or computed two-sigma error
```

```
DataSeries.PreTriggerDuration_dbl = <value> s;                          || Recommended
            || Pre-Event buffer length (the amount of the trace
            || recorded prior to the moment of triggering).
DataSeries.PostDeTriggerDuration_dbl = <value> s;                       || Recommended
            || How long DAU records after de-triggering.
```

In modern systems, which only rarely are short of event-storage memory, we recommend a ***PreTriggerDuration*** of at least 60 s and a ***PostDeTriggerDuration*** of at least 120 s. While these intervals are significantly larger than typically are used presently, they aid greatly with:

1. baseline removal and integration,
2. ambient-noise characterization, and
3. making it much more likely that rapid sequences of triggered records will overlap, thus allowing unambiguous reconstruction of data from more ***Event***s in such a sequence.

There is no longer much, if any practical need for shorter pre- and post-***Event*** intervals. In contrast, their value to analysts is substantial, sometimes critical.

```
RawSeries.Peak.Value_dbl = <value> <units>;                             || Optional
            || Peak data value in the original record.
RawSeries.Peak.Locus_dbl = <value> [ s | Hz ];                          || CONDITIONAL
            || Abscissa, in seconds from the first sample (or for spectra
            || in period or frequency) of the original RawSeries where the
            || particular Peak was encountered.
            ||
            || REQUIRED if RawSeries.Peak.Value is present. If the peak
            || is at the Nth point of an evenly-sampled time series, then
            || Locus will be
            ||
            ||      N * <sampling interval>
            ||
            || in seconds. Naturally, there will be only one peak in a
            || RawSeries, typically a peak acceleration.
```

DataSeries.Peak[(n)].Value_dbl = <value> <units>; || Recommended

|| Peak data value in this (usually processed) ***DataSeries***. This
|| may be a PGA, a PGV, a maximum S_a, or other measure of shaking
|| strength, depending on what type of ***DataSeries*** this is.
||
|| In the case of response spectra, ***Peak(n).Value*** is necessarily
|| a tag array of length equal to the size of ***DataSeries.Response-***
|| ***SpectrumDamping(n)*** (that is, there will be one peak value per
|| spectrum, per damping). Use of ***Subscript***ing, ***Peak(n)***, is
|| restricted to that instance since PGA and so forth will have
|| exactly one such value per ***DataSeries***.

DataSeries.Peak[(n)].Locus_dbl = <value> [s | Hz]; || CONDITIONAL

|| The location in the ***DataSeries*** where
|| ***DataSeries.Peak(n).Value*** was encountered, expressed
|| *either* in time (in seconds since first sample) *or* the
|| period or frequency of the spectral maximum (for ***DataSeries***
|| which are spectra) of the values (and in the same order).
||
|| Thus, if the peak occurs at the Nth point of a ***DataSeries***,
|| this value is N * <sampling rate> for time series or
|| (N – 1) * <bin width> for evenly binned spectra. For
|| unevenly sampled spectra, this value simply corresponds
|| to the frequency of a particular spectral line.

DataSeries.Peak[(n)].ResponseSpectrumPeakTime_dbl = <value> s; || CONDITIONAL

|| If and only if ***DataSeries.Peak(n).Value*** are for response
|| spectra, then also record the TIME in the original time series
|| (seconds since first sample) at which this spectral maximum is
|| encountered. This tag also will be a tag array of length ***N***
|| equal to the size of ***ResponseSpectrumDamping(n)***, that is one
|| spectrum, one peak, one peak's period, and one TIME in the
|| ***RawSeries*** (the accelerogram) where this peak was encountered.
||
|| ***DataSeries.Peak(n).Locus*** and/or
|| ***DataSeries.Peak(n).ResponseSpectrumPeakTime*** are
|| REQUIRED if ***DataSeries.Peak(n).Value*** is present (both
|| must be given if the ***DataSeries*** is a response spectrum,
|| otherwise only the former).

```
RawSeries.FilmDigitizer.Ystep_dbl = <value> um;                          || CONDITIONAL
RawSeries.FilmDigitizer.Tstep_dbl = <value> um;                          || CONDITIONAL
RawSeries.FilmDigitizer.Tmean_dbl = <value> [ s | um ];                  || Optional
RawSeries.FilmDigitizer.Tsigma_dbl = <value> [ s | um ];                 || Optional
RawSeries.FilmDigitizer.Tminimum_dbl = <value> [ s | um ];               || Optional
RawSeries.FilmDigitizer.Tmaximum_dbl = <value> [ s | um ];               || Optional
            || If this record derives from a film-recorded RawSeries,
            || then these tags are the sizes of the digitizer X- (time)
            || and Y-steps (amplitude), and statistics pertaining to
            || them, as used in digitizing that film record.
            ||
            || CONDITIONALs are REQUIRED for film-derived records.


DataSeries.ResponseSpectrumDamping(n)_dbl = <value>;                     || CONDITIONAL
            || REQUIRED if the DataSeries are response spectra, Sa, Sv, Sd,
            || PSA, or PSV.
            ||
            || A tag array that is as large as (and in exactly the same order as)
            || the ordinate columns in the main DataSeries listing matrix. This
            || tag array gives the value(s) of "structure" damping used for
            || those calculations. It is fairly common, for example, to provide
            || response-spectral values at 2-, 5-, and 10-percent structural damping.
            ||
            || Values are in fractions of critical, never in percent. So, for
            || the above example, the damping would given as 0.02, 0.05, and 0.1.
            ||
            || Typically, there also will be an explicit-abscissa column in the
            || DataSeries listing matrix, giving the periods (or conceivably the
            || frequencies). An explicit abscissa is likely because it is almost
            || invariably the case that the sampling periods of the response
            || spectrum are unevenly spaced, often at periods approximating a
            || logarithmic sequence, or at some other irregular spacing.
```

A selected subset of linear S_a, S_v or, S_d values also may be given for time series (for S_a we recommend periods including 0.2, 0.3, 1.0 and 3.0 s for compatibility with ShakeMap and other processes). These selected values typically are included when the ***DataSeries*** is the processed ("volume 2") time series, as a convenience for users and for ShakeMap and similar analyses.

We note that this is an incomplete set of response-spectral tag names—one might desire to include PSA and PSV and both linear and nonlinear versions of any values given here. Please e-mail the authors if this turns out to be an issue—such tags are easily added.

```
DataSeries.Sa[(n)].Period_dbl = <value> s;                                  || CONDITIONAL
DataSeries.Sa[(n)].StructureDamping_dbl = <value>;                          || CONDITIONAL
DataSeries.Sa[(n)].Value_dbl = <value> cm/s/s;                              || CONDITIONAL
            || Period, damping fraction, and response-spectral acceleration
            || value sets. If any one such tag is present for a given (n),
            || then all three are REQUIRED for that (n).
DataSeries.Sv[(n)].Period_dbl = <value> s;                                  || CONDITIONAL
DataSeries.Sv[(n)].StructureDamping_dbl = <value>;                          || CONDITIONAL
DataSeries.Sv[(n)].Value_dbl = <value> cm/s;                                || CONDITIONAL
            ||
DataSeries.Sd[(n)].Period_dbl = <value> s;                                  || CONDITIONAL
DataSeries.Sd[(n)].StructureDamping_dbl = <value>;                          || CONDITIONAL
DataSeries.Sd[(n)].Value_dbl = <value> cm;                                  || CONDITIONAL
            || Equivalently for response-spectral velocity and displacement.
```

Most of the following are meaningful only when this VTF file's ***DataSeries*** is a time series.

```
RawSeries.RMS_dbl = <value> <units>;                                        || Optional
            || RMS of an original time series
DataSeries.RMS_dbl = <value> <units>;                                       || Optional
            || RMS of this VTF file's DataSeries (a time series)
```

```
DataSeries.CAV_dbl = <value> [ m/s | cm/s ];                                  || Optional
            || Cumulative Absolute Velocity of this time series
DataSeries.SI_dbl = <value> [ m | cm ];                                       || Optional
            || Housner Intensity (SI) of this time series
DataSeries.AriasIntensity_dbl = <value> [ m/s | cm/s ];                       || Optional
            || Arias Intensity of this time series
DataSeries.ObservationalMMI[(n)]_int = <value; 1-12>;                         || Optional
            || MMI of this time series, determined by human
            || inspection of effects at the site. (There may be
            || more than one observer with differing opinions.)
DataSeries.InstrumentalMMI_dbl = <value, 0.0-12.0>;                           || Optional
            || Instrumentally estimated MMI from this time series
            || (may, therefore, be noninteger). Generally saturates
            || at about 10.
            ||
            || (Many practitioners believe that the MMI scale effectively
            || saturates at intensity X, 10, with all higher values in
            || effect being synonyms for "10". However, this matter is
            || not yet "given wisdom".)
            ||
            || If other common forms of intensity measure are desired
            || please e-mail the authors with a scale name, Citation,
            || and description; such tags are easily added to the VTF.

DataSeries.Duration.Over5PctG_dbl = <value> s;                                || Optional
            || Duration of shaking more than 5 percent of g_standard
            || (980.665 cm/s/s, thus 49.03 cm/s/s) of this time series.
DataSeries.Duration.AriasInterval5To95Pct_dbl = <value> s;                    || Optional
            || Interval to accumulate from 5 to 95 percent of the final Arias
            || Intensity for this time series.

DataSeries.Pick[(n)].PDateTime_txt = "YYYY-MM-DD hh:mm:ss.ss[s][Z|[+|-]hh[:mm]]";   || Recommended
            || P-Wave arrival time Pick(s) at this GeoLocation (there may
            || be more than one opinion on the time of the pick).
DataSeries.Pick[(n)].SDateTime_txt = "YYYY-MM-DD hh:mm:ss.ss[s][Z|[+|-]hh[:mm]]";   || Recommended
            || S-Wave arrival time Pick(s) at this GeoLocation (there may
            || be more than one opinion on the time of the pick).
```

```
DataSeries.Pick[(n)].Agency_txt = "<table 3E>";                                  || CONDITIONAL
            || REQUIRED if either DataSeries.Pick.PDateTime or
            || DataSeries.Pick.SDateTime is given, and is the group
            || or individual (Agency from table 3E) that made the pick(s).
DataSeries.Pick[(n)].DateTimeMade_txt = "YYYY-MM-DD[ hh:mm:ss [Z|[+|-]hh[:mm]]]";  || CONDITIONAL
            || REQUIRED if either DataSeries.Pick.PDateTime or
            || DataSeries.Pick.SDateTime is given, and is the date
            || (± time) on which the pick(s) were made (that is, the
            || pedigree of the Pick(s), not the Pick time itself) to
            || facilitate tracking updates to the Pick(s).


RawSeries.Comments_txt = "<Comments>";                                           || Optional
            || Comments about the RawSeries on matters not covered
            || elsewhere.
DataSeries.Comments_txt = "<Comments>";                                          || Optional
            || Comments about the DataSeries on matters not covered
            || elsewhere.
            ||
            || These tags are for any Comments about the DataSeries or
            || RawSeries not otherwise covered by defined DataSeries or
            || RawSeries tags, or simply to hold a comment that the
            || providing Agency considers important.
            ||
            || If multiple "lines" of text are needed, join them into
            || a single text string, with the line breaks indicated by
            || the character string "|\n|".
```

Table 3M. Recorder Timing Method

The following values are used for ***RawSeries.FirstSampleTime.Source*** and ***DataSeries.FirstSampleTime.Source***.

Code	***DateTime* reference source for this record**
"No Clock Present"	No clock is present.
"Human-set Clock"	Recorder's clock was set by hand.
"Auxiliary, Continuous Clock"	Auxiliary, continuous clock (for example, a TCG, Time Code Generator).
"Recorded Radio DateTime Signal"	Radio time signal on record (for example, WWV, WWVB, WWVH, DCF).
"Radio-tracking Clock"	Precision chronometer which tracks a (generally binary coded) radio signal continuously (WWVB, and so forth).
"GPS one-time set Clock"	Precision chronometer set only once (in this recording period) from GPS signals.
"GPS intermittent Clock"	Precision chronometer set periodically from GPS signals.
"GPS-tracking Clock"	Precision chronometer that tracks GPS signals.
"Other DateTime Source"	Unspecified or unknown time source (it is better to use the User-Extensible option if the method is known).
"_____"	USER-EXTENSIBLE LIST (use established, terse, clear, English names which must begin with **"User's description: "**).

Table 3N. Processing Information

```
Processing.DateTime_txt = "YYYY-MM-DD hh:mm:ss.sss[Z|[+|-]hh[:mm]]";                || CONDITIONAL
            || Authoritative ISO DateTime when the record was processed
            || to its final form (that is, the form in this VTF file).
            ||
            || This tag-value pair is REQUIRED if this is not an
            || unprocessed RawSeries (thus, nearly always).
            ||
            || Note that Processing.DateTime identifies the originating-
            || Agency's authoritative creation DateTime for the DataSeries in
            || this file, while ThisFile.Preparation.DateTime is the DateTime
            || when this VTF file is created (often by means of translation from
            || the originating-Agency's format).
            ||
            || Clearly, therefore, Processing.DateTime must not be later
            || than ThisFile.Preparation.DateTime, but it is possible for
            || these two DateTimes to be the same if the originating-Agency
            || directly creates this VTF file.

Processing.Agency_txt = "<value>";                                                  || CONDITIONAL
            || Processing organization or individual (Agency from table 3E).
            || If said Agency also provides the file in VTF, this value likely
            || will be the same as ThisFile.Preparation.Agency.
            ||
            || REQUIRED if Processing.DateTime is present (which is nearly
            || always).
```

Processing.Filter information is REQUIRED if one or more ***Filter***s are applied during ***Processing***. Furthermore, if any ***Processing.Filter*** tag is given, then a "sufficiently descriptive" set of such tags is REQUIRED (enough that another practitioner can determine what was done and its affect on the ***DataSeries***).

If there is more than one ***Filter***, ***Subscript*** them as ***Processing.Filter(m)….***; if only one ***Processing.Filter*** is used, the ***Subscript*** may (should) be omitted and will default to "1".

Note that there are three methods for describing a filter—poles and zeros (***PoleZero***), ***Polynomial*** equivalents of those poles and zeros, and much less complete, general parameters (***OtherDescription***) such as corner frequencies and roll-off rates. The ***PoleZero*** and ***Polynomial*** forms are preferred, the former above the latter. Also note that there are a number of global ***Processing.Filter*** tags which can be applied to any ***Filter***.

In this order, we describe ***Filter*** global tags, ***PoleZero*** tags, ***Polynomial*** tags, and ***OtherDescription*** tags:

The following group of global tags apply to all three types of ***Filter*** description:

```
Processing.Filter[(m)].Comments_txt = "<Comments>";                          || CONDITIONAL
            || This tag is REQUIRED if an unusual filtering scheme
            || was applied or if any unusual issues were encountered.
            || This tag is Optional if you have other Comments.
            ||
            || If multiple "lines" of text are needed, join them into
            || a single text string, with the line breaks indicated by
            || the character string "|\n|"—that is, Comments is not
            || Subscripted.

Processing.Filter[(m)].Mode_txt = [ "Digital" | "Analog" ];                  || CONDITIONAL
            || Whether the filter is digitally computed or is an analog
            || (almost always an electronic) filter. If not given,
            || "Digital" is assumed, so this tag is REQUIRED for
            || "Analog" filters.
Processing.Filter[(m)].Domain_txt = [ "Time" | "Frequency" ];                || CONDITIONAL
            || The domain of digital-filter "Digital" computation or
            || "Analog" implementation (the latter usually "Time").
Processing.Filter[(m)].Causality_txt = [ "Causal" | "Acausal" | "Zero Phase" ];  || CONDITIONAL
            || Is the filter causal or acausal? "Zero Phase" is, a particular
            || type of "Acausal" filter, commonly an FIR filter, or else an
            || IIR filter that has been run in both directions across a time
            || series. (The latter not recommended because it produces "soft
            || shouldered" filters that are generally  6 dB down at their
            || intended –3-dB corner(s).)
```

```
Processing.Filter[(m)].FrequencyBandType_txt = "<from table 3O(b)>";                || CONDITIONAL
Processing.Filter[(m)].PeriodBandType_txt = "<from table 3O(b)>";                   || CONDITIONAL
            || Filter bandpass character (for example, "High Pass") from
            || table 3O(b).
            ||
            || Most scientists are likely to use FrequencyBandType, while
            || many structural engineers may prefer to use PeriodBandType.
            || (This distinction between types of abscissa, seconds or Hertz,
            || matters only to the meanings of "Low Pass" and "High Pass"
            || filters.  The distinction is at which end of the passband
            || that filter corner resides.  Thus, for example, is it
            || described by a "high-pass period" or a "high-pass
            || frequency".)

Processing.Filter[(m)].FilterName_txt = "<from table 3O(a)>";                       || REQUIRED
            || The common name of this Filter.
            ||
            || For PoleZero or Polynomial descriptions, use only one
            || of the following:
            ||     "S-plane Poles/Zeros"
            ||     "z-plane Poles/Zeros"
            ||     "S-plane Polynomial"
            ||     "z-plane Polynomial".
            || For OtherDescription, use one of the common names of the Filter,
            || such as "Ormsby" (a zero-phase cosine-tapered bandpass filter—
            || the name "Ormsby" is not commonly used outside strong-motion
            || seismology and earthquake engineering).
            ||
            || Naturally, for a PoleZero description, only the values
            || "S-plane Poles/Zeros" or "z-plane Poles/Zeros" from
            || table 3O(a) are allowed.
            ||
            || Similarly, for a Polynomial description, only the values
            || "S-plane Polynomial" or "z-plane Polynomial" from table 3O(a)
            || are allowed.
```

Poles and zeros, particularly Laplace (*S*-plane) poles and zeros, are the preferred method, with ***Polynomial***s being the next most desirable method. An exception, of course, is that it will generally be preferable to describe an FIR filter by its *z*-plain ***Polynomial.Numerator*** coefficients (which are the same as the filter's "taps").

If using a ***PoleZero*** representation, then a sufficient set of the following six tags is REQUIRED. Either the ***Numerator*** or the ***Denominator*** of a ***PoleZero*** description may have no roots and therefore be only a constant (which will often be just a "1"). In all cases, a ***Constant*** may be given for that portion (***Numerator*** or ***Denominator***), but will default to a real value of one (1) if not provided—often the desired result. (***Constant*** is given type _cpx for generality, but we expect it to be real in most cases; remember that if you do not give the imaginary part, it will default to 0*j*, a real). The corresponding ***NumberOfRoots*** will be zero and no ***Root***s will be listed for that portion. Where there are roots and the ***Constant = 1;*** the ***Constant*** will typically will be omitted, defaulted to a real-value of one (1). This commonly will be the case for filters with no net gain:

```
Processing.Filter[(m)].PoleZero.Numerator.NumberOfRoots_int = <0, 1, 2, ...>;       || CONDITIONAL
            || Number of zeros in the filter (must be ≥0).
Processing.Filter[(m)].PoleZero.Denominator.NumberOfRoots_int = <0, 1, 2, ...>;     || CONDITIONAL
            || Number of poles in the filter (must be ≥0).
Processing.Filter[(m)].PoleZero.Numerator.Root(n)_cpx = <real> <imaginary>;         || CONDITIONAL
            || A complex zero. Repeat (n) exactly
            || Processing.Filter.PoleZero.Numerator.NumberOfRoots times.
Processing.Filter[(m)].PoleZero.Denominator.Root(n)_cpx = <real> <imaginary>;       || CONDITIONAL
            || A complex pole. You must repeat (n) exactly
            || Processing.Filter.PoleZero.Denominator.NumberOfRoots times.
Processing.Filter[(m)].PoleZero.Numerator.Constant_cpx = <real> <imaginary>;        || CONDITIONAL
            || A net gain or attenuation, or unity, in the case of
            || a Numerator with no roots. Defaults to one (1) if
            || not provided.
Processing.Filter[(m)].PoleZero.Denominator.Constant_cpx = <real> <imaginary>;      || CONDITIONAL
            || A net gain or attenuation, or unity, in the case of
            || a Denominator with no roots. Defaults to one (1) if
            || not provided.
```

Polynomials are close relatives of pole/zero descriptions, being the enumeration of polynomial coefficients for the ***Numerator*** and ***Denominator*** of a filter transfer function, simply the ***PoleZero*** form multiplied out. These may be in the Laplace- (*S*) or *z*-plane, as indicated by ***Sensor.TransferFunction.FilterName***, the former preferred.

The order (maximum power) of the Polynomial is **N = *NumberOfCoeff* – 1**. The ordering of the ***Coefficient***s is from 0-to-N, corresponding to ***Subscript***s 1-to-***NumberOfCoeff***. Thus, these ***Coefficient***s are given in ascending order from the constant, ***n****=1*, up to ***n****=****NumberOfCoeff****=N+1*, the *Nth*-degree coefficient. Any zero-valued ***Coefficient*** also must be given explicitly to create a complete vector of ***NumberOfCoeff*** elements (some applications will be confused by missing ***Coefficient***s).

If either the ***Numerator*** or ***Denominator*** has no roots, then the value of ***Coefficient(1)*** will act as a simple gain factor while ***NumberOfCoeff*** will equal one (1). This is a somewhat different rule than for the ***PoleZero*** form where the number of poles or zeros may be zero (0), but a ***Constant*** is always present, even if defaulted to unity (1).

```
Processing.Filter[(m)].Polynomial.Numerator.NumberOfCoeff_int = <1, 2, 3, ...>;      || CONDITIONAL
Processing.Filter[(m)].Polynomial.Denominator.NumberOfCoeff_int = <1, 2, 3, ...>;    || CONDITIONAL
            || Both must be ≥1.
Processing.Filter[(m)].Polynomial.Numerator.Coefficient(n)_cpx = <value>;            || CONDITIONAL
Processing.Filter[(m)].Polynomial.Denominator.Coefficient(n)_cpx = <value>;          || CONDITIONAL
```

If not using either the ***PoleZero*** or ***Polynomial*** forms, the following tags are another way (in combination with ***Processing.Filter.FilterName***) to describe processing filters, albeit less generally or accurately. A sufficiently complete group of these tags is REQUIRED if neither poles and zeroes nor polynomial coefficients are given, but a filter was used.

```
Processing.Filter[(m)].OtherDescription.Corner[(n)]_dbl = <value> [ Hz | s ];       || CONDITIONAL
            || Filter Corner [generally –3 dB point(s) but sometimes the
            || –6 dB point(s), as for a time-domain filter run in both
            || directions], in across a time series units of either
            || period or frequency (which unit must match your choice
            || of FrequencyBandType or PeriodBandType). There may be
            || either one (low pass, high pass, notch), two (bandpass,
            || bandstop), or several Corners(n) (rare, but possible).
            || The Subscript (n) need only be present when there are
            || two or more Cornerss in the filter.
            ||
            || Note that giving Corner at the –6 dB point is allowed only
            || for a bidirectional time-domain Processing.Filter that is
            || properly identified by FilterName. It is not allowed for
            || any TransferFunction, neither Sensor nor DAU.

Processing.Filter[(m)].OtherDescription.Decay[(n)]_dbl = <value> dB/octave;         || CONDITIONAL
            || Roll-off rate(s) beyond the one or several Cornerss. The
            || (n) must match that in Corner(n) when there is more than
            || one Corner (for example, for bandpass filters).

Processing.Filter[(m)].OtherDescription.TransitionBandwidth_dbl = <value> Hz;       || CONDITIONAL
Processing.Filter[(m)].OtherDescription.StopBandAttenuation_dbl = <value> dB;       || CONDITIONAL
            || Needed for Ormsby filter, FIR brickwall filters, and
            || so forth.
```

Processing.Filter[(m)].OtherDescription.Length_dbl = <value> s; || CONDITIONAL

|| Length of filter operator, even if symmetric, when doing
|| time domain filtering (for example, FIR, IIR). For an FIR, this
|| value often is called the "number of taps" although a few
|| users define "taps" as <1 + ((***Length*** – 1) / 2)>, which is
|| the least number of multiply-accumulate operations required
|| for a symmetric (zero phase) FIR filter, notable for their
|| symmetric, odd-length impulse responses.
||
|| Note that this value is in seconds for generality.

Processing.Resampling[(m)].InitialSamplingRate_dbl = <old sample rate> Hz; || CONDITIONAL
Processing.Resampling[(m)].FinalSamplingRate_dbl = <new sample rate> Hz; || CONDITIONAL
Processing.Resampling[(m)].FilterSubscript_int = <index>; || CONDITIONAL

|| Legacy data were sometimes decimated to reduce computation
|| times and storage requirements (back when computers were much
|| less capable than they were in 2008). Similarly, some modern data
|| occasionally are up-sampled to make identification of peak values
|| and arrival times easier and more accurate.
||
|| The first two tags are REQUIRED if this ***DataSeries*** was either
|| decimated or up-sampled, with both values being samples per
|| second; the pair, therefore, implying the up- or down-sampling
|| factor.
||
|| If desired, users may employ ***Processing.Filter*** values to describe
|| resampling filter(s) more completely. If you do this, make it clear
|| with the corresponding ***Comments*** and ***FilterSubscript*** which ***Processing***
|| filter is, in fact, the ***Resampling Filter*** (that is, it will be
|| ***Processing.Filter(Processing.Resampling.FilterSubscript)***. So
|| the third tag is REQUIRED when using ***Filter*** to describe ***Resampling***.
||
|| The index ***(m)*** can be used to describe multiple steps in
|| decimation and/or up-sampling. For example, some legacy CSMIP
|| data first decimate by four then by an additional amount for
|| computing long-period results. This would be described
|| with ***(m)***, ***(1)*** being the initial decimation.

The following four tags relate to double-integration initial conditions. Also see ***DataSeries.Pad.FirstOfOriginal*** and ***DataSeries.Pad.LastOfOriginal***.

```
RawSeries.Mean_dbl = <value> <units>;                                          || Recommended
            || Mean value removed to make the previous time series
            || in this Processing path zero-mean.
DataSeries.Mean_dbl = <value> <units>;                                         || Recommended
            || Mean value of this time series.

Processing.Report[(n)].Citation_txt = "<citation>";                            || Optional
Processing.Report[(n)].URL_txt = "<Web address>";                              || Optional
            || If there is/are report(s) available for this DataSeries,
            || describing its Processing, provide a citation with one or
            || both of these tags.


Processing.InitialValue.Velocity_dbl = <value> <units>;                        || Recommended
Processing.InitialValue.Displacement_dbl = <value> <units>;                    || Recommended
            || Often it is necessary to begin an integration (of acceleration
            || to velocity, or velocity to displacement) from an initial value
            || other than zero in order to produce a trace with is found in
            || this VTF file as the DataSeries. The tag Displacement, for
            || example,  is the constant of integration and corresponds to
            || "remembering" the acceleration baseline initial correction
            || needed to integrate the RawSeries into either this VTF file's
            || DataSeries, or into an intermediary velocity signal.

Processing.InitialValue.AccelerationCorrectionRecord_txt = "<name of file or ID>";  || Recommended
            || This third tag points to some more general description of baseline
            || corrections (including the fact that an offset (mean) correction in
            || velocity is a Dirac delta in acceleration, an offset correction in
            || displacement is the derivative of a Dirac delta in acceleration,
            || and so forth). That is, AccelerationCorrectionRecord points to
            || a providing-Agency's filename, VTF filename, or unique identifier
            || for a time series in which the DataSeries.PhysicalParameter is
            || effectively "Acceleration Correction Baseline".
            ||
```

|| Corrections made prior to integration routinely include such
|| delta functions (typically convolved with a smoothing filter
|| for convenience). Smoothed baseline-correction time series also
|| regularly include by one or more higher-order corrections (such
|| as a piecewise-linear velocity baseline, which translate into
|| an acceleration step). Thus, the most complete description of
|| a baseline correction is a complete acceleration time series
|| of the same length and sampling rate as the (commonly padded)
|| acceleration ***RawSeries***.
||
|| (We do not now include velocity-correction equivalents because
|| these are subsumed with the acceleration correction. However,
|| it will be desirable to add such tags whenever the VTF begins
|| to be used for primary velocity records, such as those of the
|| Tokyo Sokushin VSE-355G3.)

Processing.ConstantsUsed_txt = ["Precise" | "Nominal" | "Poor"]; || Recommended
|| Describes the accuracy of conversion and correction constants
|| used in correcting and ***Processing*** the seismogram. The list of
|| possible values is from best to worst, with **"Nominal"** meaning
|| that up to a few percent total gain and phase error is possible
|| in the pass band, while **"Precise"** means that no more than
|| 1-percent total error is likely.

Processing.Problem[(m)].Status_txt = ["None" | "Problem of Unknown Status" |
"Corrected" | "Not Corrected" | "Presence of Problem Uncertain"]; || REQUIRED
|| If no issues with data ***Processing*** are known to have existed
|| at any time, then **"None"** (that is "No Problems") is given.
|| If problems with the ***DataSeries*** are known to be, or to have
|| been, present at any time, then you must include at least
|| one ***Processing.Problem.Comments(n)*** tag explaining what the
|| problems are or were. Such problems are either **"Corrected"**
|| or **"Not** [yet] **Corrected"**. **"Problem of Unknown Status"** may
|| be used when the correction status is unknown, but it is
|| believed there has been a problem. **"Presence of Problem**
|| **Uncertain"** just means there is neither exculpatory nor
|| indicting knowledge.
||

|| The ***Subscript (n)*** allows the listing of more than one
|| ***Problem***, but will often be only an implicit ***(1)***.

Processing.Problem[(m)].Comments[(n)]_txt = “Description”; || CONDITIONAL

|| If ***Processing.Problem.Status*** is anything other
|| than “None”, then this tag is REQUIRED for each ***(n)***
|| to describe the nature of the problem(s) and indicate
|| their current disposition.
||
|| If multiple “lines” of text are required for any
|| particular ***(n)***, then concatenate them into a single
|| text string with the lines separated by the character
|| string ***“|\n|”***.

Processing.Comments_txt = “<Comments>”; || Recommended

|| For general ***Comments*** about ***Processing*** that are not
|| appropriate to ***Processing.Problem.Comments*** or
|| ***Processing.SpecialProcessing***.
||
|| As above, for multiple “lines” of text in any particular
|| ***Comments(n)***, concatenate them into a single text string
|| with the lines separated by the character string ***“|\n|”***.

Processing.BlueBookVolume_int = [0-3]; || CONDITIONAL
Processing.Stage_txt = [“Raw” | “Preliminary” | “Final”]; || CONDITIONAL
Processing.HumanReview_txt = [“None” | “Reviewed” | “Updated”]; || REQUIRED
Processing.Instance_int = <positive integer>; || REQUIRED
Processing.SpecialProcessing_txt = “<Description>”; || CONDITIONAL

|| One of the two tags ***Processing.BlueBookVolume*** or
|| ***Processing.Stage*** is required to indicate the stage
|| of the ***Processing***. Furthermore, within each ***Processing***
|| stage, there may be one or more instances of the release
|| of a ***DataSeries***, indicated by ***Processing.Instance***. The
|| latter is REQUIRED, but will often be exactly **1**, the
|| first and only such ***Processing*** instance.
||

|| It also is REQUIRED that the degree of human analysis
|| of the data (*versus* automated computer ***Processing***) be
|| indicated. If the data are, so far, only processed
|| automatically then set ***Processing.HumanReview_txt*** = **"None";.**
|| (This often will be the case with the initial release of
|| data from modern, digital systems.) At subsequent steps
|| in the ***Processing*** scheme, use the other values to indicate
|| that the data have received a preliminary review by a human,
|| or that they have been processed fully and subjected to full
|| quality-control procedures. Legacy and older data generally
|| are predominantly human-processed and are, therefore, given
|| the value ***Processing.HumanReview_txt*** **= "Updated";** in most
|| cases. (We are not so optimistic as to offer the value
|| **"Final".**)
||
|| ***Processing.BlueBookVolume*** takes the values:
|| 0 = unprocessed data
|| 1 = converted and baseline-corrected ground motion
|| 2 = filtered acceleration, velocity, or displacement
|| 3 = a spectrum or spectra, including response spectra
||
|| Finally, if the ***DataSeries*** has been processed in a manner
|| that is not typical of the originating ***Agency***, indicate this
|| fact by including the tag ***Processing.SpecialProcessing***
|| (REQUIRED when there is unusual processing) and using its
|| contents to describe what was unusual about the ***Processing***.

NOTES:
(1) Channel numbers and sensing directions are now in table 3K(a).
(2) All geographic-location information is now in tags defined in table 3F.

Table 3O(a). Codes for Filter Types Used in Processing

Table 3O(a) describes the possible values of ***Processing.Filter(n).FilterName***. The first four values also may be used for ***Sensor.TransferFunction.FilterName*** or ***Sensor.TransferFunction. FilterName***.

Note that the causality of filters are indicated by the formal definitions of those filters and by ***Processing.Filter.Causality*** and ***DAU.TransferFunction.Causality***, while ***Sensor*** transfer functions are assumed to be causal. This may pose a conflict for ***Sensors*** packaged with digitizers, but it should be possible to differentiate the two systems—we take the position that a digitizer, even when packaged inside the ***Sensor***, is part of the ***DAU***, and the appropriate descriptive tags will be found there.

Phrase	**Any *FilterName***
"S-plane Poles/Zeros"	A list of complex Laplace-plane poles and zeros follows.
"z-plane Poles/Zeros"	A list of complex z-plane poles and zeros follows.
"S-plane Polynomial"	A list of real Laplace-plane polynomial coefficients.
"z-plane Polynomial"	A list of real z-plane polynomial coefficients
(the following not allowed for pole-zero descriptions)	
~~"None"~~	~~No filtering performed~~ (just leave out the variables entirely)
"Rectangular"	Rectangular
"Cosine Bell"	Cosine Bell
"Ormsby"	Ormsby
"Butterworth"	Butterworth, single direction (state "Causal" or "Acausal" please).
"Bessel"	Bessel, single direction (state "Causal" or "Acausal" please).

"IIR"	Generalized time-domain IIR filter.	It is best to treat these as z-plane poles and zeros.
"FIR"	FIR (state filter's causality).	
"Other Transfer Function"	Use for ***DAU*** or ***Sensor TransferFunction***s not described formally.	
"Other Processing Filter"	Any other ***Filter*** response not listed above.	
"_____"	USER-EXTENSIBLE LIST (use established, terse, clear, English names which must begin with **"User's description: "**).	

Table 3O(b). Codes for Filter Types Used in Processing

Similarly, table 3O(b) shows the values that ***Processing.Filter(n).FrequencyBandType*** or ***Processing.Filter(n).PeriodBandType*** may take.

Phrase	**Filter *FrequencyBandType* or *PeriodBandType***
"High Pass"	A high-pass filter, with respect to frequency or to period.
"Low Pass"	A low-pass filter, with respect to frequency or to period.
"Bandpass"	A band-pass filter.
"Bandstop"	A band-stop filter.
"Notch"	A notch filter.
"Deconvolution"	A transfer-function deconvolution filter.
"_____"	USER-EXTENSIBLE LIST (use established, terse, clear, English names which must begin with **"User's description: "**).

Table 3P. Relational Tags Involving Two Geographic Locations

```
EventGeoLocation.EpicentralDistance.Value_dbl = <value> km;                    || Optional
EventGeoLocation.EpicentralDistance.ValueError_dbl = <value> km;               || Optional
            || Epicenter-to-Sensor GeoLocation distance and error.

EventGeoLocation.ForeAzimuth.Value_dbl = <value> deg;                          || Optional
EventGeoLocation.ForeAzimuth.ValueError_dbl = <value> deg;                     || Optional
            || Epicenter-to-Sensor GeoLocation azimuth and error.
EventGeoLocation.BackAzimuth.Value_dbl = <value> deg;                          || Optional
EventGeoLocation.BackAzimuth.ValueError_dbl = <value> deg;                     || Optional
            || Sensor GeoLocation-to-Epicenter azimuth and error.

EventGeoLocation.Comments_txt = <value>;                                       || Optional
            || Comments, including any needed identification of which
            || Event and GeoLocation are being combined. We currently
            || do not have an elegant or unique way to do this.
            ||
            || If multiple "lines" of text are required, then
            || concatenate them into a single text string, with
            || the lines separated by the character string "/\n/".


GeoLocationRupture[(m)].ClosestRuptureDistance.Value_dbl = <value> km;         || Optional
GeoLocationRupture[(m)].ClosestSurfaceDistance.Value_dbl = <value> km;         || Optional
GeoLocationRupture[(m)].ClosestSeismogenicDistance.Value_dbl = <value> km;     || Optional
GeoLocationRupture[(m)].RMSdistance.Value_dbl = <value> km;                    || Optional
GeoLocationRupture[(m)].SegmentDistance.Segment(i).Value_dbl = <value> km;     || Optional
            || Values and …
GeoLocationRupture[(m)].ClosestRuptureDistance.ValueError_dbl = <value> km;    || Optional
GeoLocationRupture[(m)].ClosestSurfaceDistance.ValueError_dbl = <value> km;    || Optional
GeoLocationRupture[(m)].ClosestSeismogenicDistance.ValueError_dbl = <value> km;|| Optional
GeoLocationRupture[(m)].RMSdistance.ValueError_dbl = <value> km;               || Optional
GeoLocationRupture[(m)].SegmentDistance.Segment(i).ValueError_dbl = <value> km;|| Optional
            || their errors.
            ||
```

|| ***Sensor GeoLocation***-to-Fault-Rupture (and -to-Rupture-Surface-
|| Projection) closest distances. Such values are often used in
|| ground-motion model calculations). ***Subscript***s ***(m)*** and ***(i)*** refer
|| to a particular rupture model and must be the same as the ***Subscript***s
|| in ***Event.DetailedFaultRupture(m).Segment(i).Latitude*** and related tags.
||
|| Note that ***ClosestSurfaceDistance*** is the *Boore and others (1993, 1994, 1997)* distance, ***ClosestSeismogenicDistance*** is the *Campbell* (1997) distance, and ***ClosestRuptureDistance*** is *"ClstD"* in the NGA flatfile.
||
|| The first four tags are distances either to a simple, one-segment rupture model, or to a rupture taken as a whole. In contrast, the fifth tag is for distances to each segment of a modeled rupture surface area:
||
|| ***GeoLocationRupture(m).SegmentDistance.Segment(i).Value***
||
|| Of course, any of these tags can be computed from the ***Sensor***'s ***GeoLocation*** together with the rupture description in ***Event.DetailedFaultRupture.Segment(i)*** and, clearly, should be consistent.

GeoLocationRupture[(m)].SurfaceRupture_txt =
["Yes" | "No" | "Inferred" | "Unclear"]; || Optional
|| Whether or not this ***Event*** has an associated surface rupture.
|| (If nothing is known about surface ruptures, you may set the value to **NULL**, or leave out the tag.)
||
|| (While this and the next two tags might be more appropriately located in class ***Event***, we believe there is a significant need to make explicit the correlation with other ***GeoLocationRupture*** tags.)

GeoLocationRupture[(m)].SurfaceRuptureInference_txt = "<value>"; || Optional
|| The reason for any inference of surface rupture.

```
GeoLocationRupture[(m)].Comments_txt = <value>;                    || Optional
            || Comments, including any needed identification of which
            || Event and GeoLocation are referenced here.
            ||
            || If multiple "lines" of text are required, then
            || concatenate them into a single text string with
            || the "lines" separated by the character string "/\n/".
```

We note that there is no indication above of which ***Event.Hypocenter***, ***FaultRupture***, or ***DetailedFaultRupture*** and which ***Sensor GeoLocation*** are referenced by ***EventGeoLocation*** and ***GeoLocationRupture***. Should there be several of these tags listed in the header, then as a rule these ***EventGeoLocation*** and ***GeoLocationRupture*** values are likely to be assumed to reference the most authoritative versions of ***Event*** and ***GeoLocation***, unless otherwise indicated in ***EventGeoLocation.Comments*** or ***GeoLocationRupture.Comments.*** Currently, we do not know of any more elegant solution to this referencing issue. (Two-dimensional tag arrays are not supported and, in any case, double indexing is opaque in meaning, violating the "self-evident file" design goal.

Table 3Q. Private Tags (Avoid)

An ***Agency*** may (but ideally should not) define internal-use tag sets of the following format. ***Private*** tags, of course, may be followed by "ll" comments as desired. The ***Private*** tag definition is formally:

```
Private[(n)].TagName_txt = "<Terse Tag Name>";                     || CONDITIONAL
Private[(n)].DataType_txt = [ "_txt" | "_int" | "_dbl" | "_cpx"];  || CONDITIONAL
Private[(n)].TextValue_txt = "<Value assigned to tag>";            || CONDITIONAL
Private[(n)].TextValueError_txt = "<Value error assigned to tag>"; || Optional
Private[(n)].Units_txt = "<From table 2>";                         || CONDITIONAL
Private[(n)].Agency_txt = "<table 3E Code>";                       || CONDITIONAL
Private[(n)].MeaningAndUse_txt = "<Tag's meaning and use>";        || CONDITIONAL
```

where **<Terse Tag Name>** must be a self-evident descriptive phrase, *in English*. ***MeaningAndUse*** shall be added to define the meaning and use of the tag clearly, also *in English*. The goal is that any future user, who is completely *un*familiar with your work, *must be able to understand your tag* and why it is used.

The "CONDITIONALs" mean that once a ***TagName*** is created, for any ***(n)***, all other CONDITIONALs must be present too, including the ***DataType***, data ***Units*** (NULL if unitless, but remember that a unit "counts" is available and meaningful), and the individual or organization (***Agency***) creating that ***Private*** tag. The ***Value*** to which this tag is set must be included too, noting that every ***Value*** shall be encoded as text, including numerical data. ***TextValueError*** need not be given.

"**Avoid**" means the ***Private***-tag class generally should not be necessary and will be hard for others to use—try not to invoke it.

We have attempted to anticipate all user needs by providing appropriate tags, however, we expect the need for additional options will arise. Thus, ***Private*** tags may be used for quality-control information, data not yet supported by the VTF, specialized network data, and so forth, assuming that using some already-available ***Comments*** instead is not an acceptable alternative.

The ***Private***-tag mechanism is one way in which new tags may be proposed to COSMOS, although a more formal proposal is preferred (that is, please contact one of the authors).

VTF parsers will treat all ***Private*** tags as simple text when read for ***Processing***, and they will be passed through unchanged, as viable tag sets, within the data files themselves. That is, ***DataType*** is advisory, suggesting how to parse the ***Value***. (Your own parser may read them in, and use them as, particular types of values. Since the VTF source code will be available to all users, it should be a relatively simple programming job to modify parsers, output generators, and translators to read and write your ***Private*** tags.)

Table 3R. *DataSeries* Tags Closely Associated with Table 4

DataSeries.NumberOfSamples_int = <number of samples in the ***DataSeries***>; || REQUIRED

|| Number of samples in time (or bins in frequency, or periods), regardless of whether the interval between samples, frequency bins, or periods is regular or irregular. This value must equal the number of rows (lines) in the ***DataSeries*** listing matrix between "***DataSeries.NumberOfSamples_int*** = {" and "};", the sense being "the number of values per damping value" in the case of response spectra. Any zero-value-abscissa bin **is** included in this count.
||
|| It is also the number of lines in any ASCII listing accessed through any of the ***DataSeries.Remote…URL_txt*** tags. There is no provision at present for differing formats or line counts between ***DataSeries.DataSeriesValues*** and a remote file. So if the local and remote copies of the ***DataSeries*** differ in size or format, only one should be provided. The same is true of ***DataSeries.OrdinateUnits*** and a few related tags.

DataSeries.OrdinateUnits[(m)]_txt = "<units>"; || REQUIRED

|| A list of units for the ordinate(s) in the ***DataSeries*** listing matrix (from table 2), one index for each ordinate column in the ***DataSeries*** listing matrix. There will be as few as one set of units in the case of any ***DataSeries*** except response spectra. (The **<units>** for the abscissa are given by ***DataSeries.AbscissaUnits***.)
||
|| These units *must* conform to the type of data (from table 3B(b)) indicated by ***DataSeries.PhysicalParameter***.

DataSeries.OrdinateFormat[(m)]_txt = "<FORTRAN format specification>"; || CONDITIONAL

|| REQUIRED whenever the ***DataSeries*** is provided, either here in this VTF file or at a ***URL*** as an ASCII file.
||
|| The Fortran-style format(s) for the ordinate portion(s) of the ***DataSeries*** listing matrix.
||

|| Where there are multiple-ordinate column vectors in the ordinate,
|| the format of each and every ordinate column (except for complex
|| values) is indicated by a separate ***(m)***. That is, there are no
|| repetition counts in these formats.
||
|| (The exception is that, because there must be two columns for
|| every complex-valued ordinate, it is important that each such
|| real/imaginary pair associate to a single ***(m)***. That is, each
|| format for a complex value is given as an concatenated pair of
|| formats in a single string, and that it must include the
|| requisite white space between the real and imaginary columns.)
||
|| In the case where both this VTF file and a ***URL*** to an ASCII
|| file are present, this single ***OrdinateFormat*** must apply to
|| both. That is, at present, the VTF does not have a way to
|| express different ***OrdinateFormat***s (or ***AbscissaFormat***s) in
|| local and remote ASCII ***DataSeries*** listings.
||
|| While the ***OrdinateFormat*** defined here might be expected to
|| include an entry for the abscissa column-vector's format,
|| that role is filled by ***DataSeries.AbscissaFormat***.
||
|| Most often, the ***DataSeries*** listing matrix will have exactly
|| one value per line, a single column of real values. This will
|| be true of all real, evenly sampled time series. Thus there
|| most often will be a ***DataSeries.OrdinateUnits(1)*** and a
|| ***DataSeries.OrdinateFormat(1)***.

Critical Notes on Formats

Authors are cautioned to provide enough text precision in every format to preserve the precision of the original recording, with no conversion or round-off errors. (They should, therefore, have a mantissa at least as precise as the analog-to-digital converter [ADC] and generally somewhat longer). For 24-bit records in cm/s/s, for example, try "F11.5". For related complex spectra try "E16.9 E16.9". Over-specifying is not recommended because it may imply greater precision than is available from the data, but under-specifying is worse, damaging the data.

In all *DataSeries* listing matrices it is essential that the *AbscissaFormat* and *OrdinateFormat*(s) you specify be wide enough to ensure (1) that there will always be at least one blank space between each value, and (2) that columns also will align Fortran style. This rule is necessary to ensure the coexistence of older languages like Fortran and newer ones like C, MatLab™, and Excel™. That is, the VTF requires both column alignment *and* white-space separation in the *DataSeries* listing matrix.

Response spectra generally will have both an abscissa column (a real-valued period or frequency) plus exactly as many real-valued ordinate column vectors as there are damping values in ***DataSeries.ResponseSpectrumDamping(n)***. Thus, there will be ***N+1*** columns in the ***DataSeries*** listing matrix, while both ***DataSeries.OrdinateFormat(n)*** and ***DataSeries.OrdinateUnits(n)*** must have the same dimensions and order as ***ResponseSpectrumDamping(n)***.

Tags for remote ***DataSeries*** (those external to this VTF file):

```
DataSeries.Remote.RealDataSeriesURL_txt = "<Web address>";                          || CONDITIONAL
DataSeries.Remote.ComplexDataSeriesURL_txt = "<Web address>";                       || CONDITIONAL
DataSeries.Remote.DataType_txt = [ "IEEE float" | "XDR IEEE float" | "ASCII" ];     || CONDITIONAL
            || These tags give the URL and the form of file containing an
            || "original" copy of the DataSeries, generally maintained by
            || the Agency providing of the data. We do not yet support
            || integer DataSeries or nonstandard floating-point formats.
            ||
            || The aims are to support, compactly, (1) a "dataless" redistribution
            || of updated metadata for a file and (2) dual reference to the DataSeries,
            || both stored within this VTF file and at the URL. While we allow dataless
            || VTF files, we strongly recommend that the full DataSeries listing also
            || be provided here every time (that is, by using the
            || "DataSeries.DataSeriesValues_txt = { … };" DataSeries listing matrix).
            || We urge provision of either (1) a complete VTF file with DataSeries
            || or (2) that complete VTF file plus a URL.
            ||
```

|| One of the first two tags is REQUIRED if (1) this is not a “dataless”
|| file and (2) the ***DataSeries*** is *not* provided below, in this file (by
|| “***DataSeries.DataSeriesValues_txt*** *= { … };”*, the ***DataSeries*** listing
|| matrix). Otherwise, these tags are Optional and may be provided
|| in addition to the ***DataSeries*** listing within this file, as desired.
||
|| Again, we do not recommend either dataless or URL-only usage since
|| a key goal of the VTF is to present, in s single file, a complete
|| ***DataSeries*** with all its supporting metadata.
||
|| The third tag, ***DataSeries.Remote.DataType***, is REQUIRED whenever one
|| of the first two tags is given. ***“IEEE float”*** implies that the data
|| found at that URL will be in the provider’s local flavor of IEEE
|| double or IEEE complex double (whichever is implied by this VTF file).
|| ***“XDR IEEE float”*** indicates that the data are in the “XDR” (eXternal
|| Data Representation) version of those formats (cf. below).
||
|| Other, machine dependent, floating-point formats and binary-integer
|| formats are not supported at this time.
||
|| ***“ASCII”*** indicates that the data are text in exactly the same format
|| used for the ***DataSeries*** listing matrix, just as presented in this VTF
|| file. That is, it must use exactly the same ***OrdinateFormat*** and
|| ***AbscissaFormat*** as in this VTF file. There is currently no provision
|| for differing ASCII formats between this file and any remote “***ASCII***”
|| file.
||
|| XDR encoding of IEEE floating-point values is strongly recommended
|| (that is, ***DataSeries.Remote.DataType_txt*** **= “XDR IEEE float”;**), simply
|| because XDR-binary format is well-established and fully portable
|| between computers of differing architectures and, therefore, is easily
|| translated between them. XDR input and output subroutines also exist
|| for every architecture supporting NFS file systems, which includes
|| all versions of UNIX and, therefore, every major computer architecture
|| in use).

DataSeries.Remote.BytesToSkip_int = <bytes>; || CONDITIONAL
DataSeries.Remote.LinesToSkip_int = <lines>; || CONDITIONAL

|| These tags describe how far to skip into the file to get beyond
|| any preamble (for example, a header) in that remote file. (This
|| skip distance can be described in bytes for IEEE-float files or
|| in lines for ASCII files). Since the goal is to use the VTF header
|| given above, header information in the remote file must be skipped
|| (there is no provision to read remote header information from VTF
|| software). Further, a concatenated remote file can be handled by
|| defining the correct number of samples and how far into the
|| remote file to skip before reading.
||
|| (Note that we do not recommend concatenating files for various
|| ***DataSeries*** anywhere, since these can be difficult for users to
|| parse or sort. There is no longer any reason to save a few
|| bytes by such concatenation.)
||
|| Whichever of these two tags fits the type of file at the URL, that
|| tag is REQUIRED whenever the first byte of that remote file is not
|| the first character of the ***DataSeries***. (So, as mentioned, use
|| ***BytesToSkip*** for binary files and ***LinesToSkip*** for ASCII files.)
||
|| Both default to zero if not given.

DataSeries.Checksum_int = <value>; || Recommended

|| Optional checksum for the ***DataSeries*** listing matrix only, that
|| listed in "***DataSeries.DataSeriesValues_txt*** *= { … };*".
||
|| This checksum is computed as follows: sum all the mantissa and
|| exponent digits, if any, (first subtracting five from each digit
|| so that the values summed range from –5 to +4); add to this sum
|| each sign, valued as +1 for each positive and as -1 for each
|| negative (these without first subtracting five, so values of
|| either ±1); to this sum, add nothing for each NaN.
||
|| (Note that this value must be recomputed every time any change
|| is made to the ***DataSeries*** listing matrix, including to any of
|| its formats.)

Table 4. The Time Series or Spectrum (the *DataSeries Values*)

```
DataSeries.DataSeriesValues_txt = {                                  || RECOMMENDED
      <Value 1,1>[ <Value 1,2>[ <Value 1,3> <…>]]                    || CONDITIONAL
      …                                                              || …
      <Value N,1>[ <Value N,2>[ <Value N,3> <…>]]                    || CONDITIONAL
      };                                                             || CONDITIONAL
```

where "N" is the number of lines in the ***DataSeries*** and, therefore, must equal to ***DataSeries.NumberOfSamples***. The "***{***" and "***};***" are ***DataSeries*** listing "matrix" (or "vector" for a single, real value) delimiters.

Although ***DataSeries.DataSeriesValues*** indicates type **"_txt"**, the values will be parsed, as appropriate, into ***"_int"***, ***"_dbl"***, ***"_cpx"***, or a mix of these.

We recommend that the ***DataSeries*** be provided here in the VTF file in addition to any URL pointer to an ***Agency*** original and even when only the metadata have changed and one might be tempted to distribute a "dataless" file.

If the first line is present, the entire ***DataSeries*** listing matrix is REQUIRED, as well as the closing line, "***};***". Avoid blank lines within the ***DataSeries*** listing matrix, though they are allowed.

Notwithstanding this recommendation, the ***DataSeries*** listing matrix is allowed be provided *only* by way of a URL. Similarly, a VTF file is allowed to be "dataless" (that is, containing only a header and neither a ***DataSeriesValues*** nor a URL pointing to the original ***DataSeries***); in this case, this VTF file is intended only for use in the distribution of revised header metadata applicable to a ***DataSeries*** previously disseminated with the same ***ThisFile.DataSeriesCOSMOSidentifier*** (analogous to a "dataless SEED" file as used within the CISN, for example).

The one or more columns of the ***DataSeries*** value listing matrix go between "{\n" and "\n};", to the exclusion of all comments and preferably to the exclusion of blank lines. (Here "\n" means "newline" or "line break", not printable characters in the ***DataSeries*** listing matrix.)

Missing values must be replaced by "NaN" ("Not a Number") which parses in most systems into the equivalent IEEE value "NaN". In rare systems "NaN" will have to be replaced by some extremal value that is then announced to the user reading in the ***DataSeries***. Default NaN value replacements may be suggested to the parser with ***ThisFile.NullComplexValue***, ***ThisFile.NullRealValue***, or ***ThisFile.NullIntValue***.

Common forms of ***DataSeries*** listing matrix include, but are not limited to the following:

(1) A real-valued time series will form a single-column matrix (a vector):

```
<amplitude>
```

where time begins at ***DataSeries.FirstSampleTime.DateTime*** and constantly increases, either in steps of (1 / ***DataSeries.SamplesPerSecond***) or ***DataSeries.SampleInterval***, or as given in an explicit abscissa, hence

```
<time>  <amplitude>
```

where **<time>** is seconds from ***DataSeries.FirstSampleTime.DateTime***, and the value of the first element of the ***<time>*** vector usually will be zero.

(3) Regularly-binned spectra (for example, a DFT or PSD spectrum) without an explicit abscissa will form either one or two columns (for real or complex spectral values):

```
<amplitude>
```

or

```
<real>  <imaginary>
```

where the abscissa is taken to be equal-width frequency or period bins (the latter rare) with the first centered at ***DataSeries.FirstBinFrequency*** or ***DataSeries.FirstBinPeriod*** and proceeding to positive abscissa values in steps of ***DataSeries.FrequencyBin*** or ***DataSeries.PeriodBin***. The longest-period bin, while it could be "**Inf s**" period, need not correspond to 0-Hz, nor to the first finite frequency of a DFT, nor any other particular value. It is worth noting that a spectrum regularly sampled by period will not have an abscissa that is the inverse of a DFT's frequencies, since the latter tend to be spaced evenly in frequency, so hyperbolically in period.

For irregularly-binned spectra (or whenever an explicit abscissa is desired) the ***DataSeries*** matrix listing will form two or three columns (assuming the abscissa is real-valued), either:

```
[ <frequency> | <period> ]  <amplitude>
```

or

```
[ <frequency> | <period> ]  <real>  <imaginary>
```

where **<frequency>** and **<period>** refer to bin centers and must be in ascending order of the abscissa.

(5) Response spectra (for example, S_a) typically will be irregularly sampled and therefore form two or more columns (assuming a real-valued abscissa):

```
[ <frequency> | <period> ] <amplitude(Damping 1)> [ <amplitude(Damping 2)> [...]]
```

where **<frequency>** and **<period>** refer to (structural) resonator natural frequencies or periods and where exactly one of the response spectral amplitudes must be given for each damping value given in the tag array ***DataSeries.ResponseSpectrumDamping(n)***—that is, there would be ***N+1*** columns.

References

Aki, K., and Richards, P. G. 1980. *Quantitative Seismology, Theory and Methods*: San Francisco, W. H. Freeman and Company.

Boore, D.M., and Akkar, S., 2003. Effect of causal and acausal filters on elastic and inelastic response spectra, *Earthquake Eng. Struct. Dyn.*, **32**, 1729–1748.

Boore, D.M., Joyner, W.B., and Fumal, T.E., 1993. Estimation of response spectra and peak accelerations from western North American earthquakes: an interim report, U.S. Geological Survey Open-File Report, **93-509**, 72 p. [http://pubs.er.usgs.gov/usgspubs/ofr/ofr93509] (last accessed 14 October 2008).

Boore, D.M., Joyner, W.B., and Fumal, T.E., 1994. Estimation of response spectra and peak accelerations from western North American earthquakes: an interim report, part 2, U.S. Geological Survey Open-File Report, **94-127**, 40 p. [http://pubs.er.usgs.gov/usgspubs/ofr/ofr94127] (last accessed 14 October 2008).

Boore, D.M., Joyner, W.B., and Fumal, T.E., 1997. Equations for Estimating Horizontal Response Spectra and Peak Accelerations from Western North American Earthquakes: A Summary of Recent Work, *Seism. Res. Lett.*, **68**, no. 1, 128–153.

Basöz, N., and Kiremidjian, A.S., 1995. Prioritization of Bridges for Seismic Retrofitting, *Technical Report* **NCEER-95-0007**, Multidisciplinary Center for Earthquake Engineering Research, Buffalo, New York.

Basöz, N., and Kiremidjian, A.S., 1996. Prioritization of Bridges for Seismic Retrofitting, *Technical Report* **No. 118**, John A. Blume Earthquake Engineering Center, Civil Engineering Department, Stanford University, Stanford, California.

Bray, J.D. and Rodriguez-Marek, A., Geotechnical site categories, *Proc. First PEER-PG&E Workshop on Seismic Reliability of Utility Lifelines*, San Francisco, CA, August, 1997.

California Governor's Office of Emergency Services, 2004, Data standardization guidelines for loss estimation–Populating inventory databases for HAZUS 99, 114 p.

Campbell, K.W., 1997. Empirical near-source attenuation relationships for horizontal and vertical components of peak ground acceleration, peak ground velocity, and pseudo-absolute acceleration response spectra, *Seism. Res. Lett.*, **68**, no. 1, 154–179.

Choy, G.L., and Boatwright, J.L., Global patterns of radiated seismic energy and apparent stress, *J. Geophys. Res.*, **100**, no. B9, 18,205–18,228, 1995.

Dana, P.H., 1997. http://www.colorado.edu/geography/gcraft/notes/datum/dlist.html, University of Texas, Austin. (last accessed 07 Oct 2008).

ISO 8601:2004, Data elements and interchange formats—Information interchange—Representation of dates and times, http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=26780. (Last accessed 08 October 2008.)

Hanks and Kanamori, 1979. A moment magnitude scale, *J. Geophys. Res.*, **84**, no. B5, 2348–2350.

Gutenberg, B., and Richter, C.F., 1956. Earthquake magnitude, intensity, energy and acceleration, *Bull. Seis. Soc. Am.*, **46**, 105–145.

IASPEI, 2005. Summary of magnitude working group recommendations on standard procedures for determining earthquake magnitudes from digital data, http://www.iaspei.org/commissions/CSOI.html. (last accessed 07 Oct 2008)

Lee, W.H.K., Bennet, R.E. and Meaghu, K.L., 1972. A method of estimating magnitude of local earthquakes from signal duration, U.S. Geological Survey Open File Report 72-223, p. 28. [http://pubs.er.usgs.gov/usgspubs/ofr/ofr72223] (last accessed 14 October 2008).

Richter, C.F., 1935. An earthquake magnitude scale, *Bull. Seis. Soc. Am.*, **25**, no. 1, 1–32.

Nuttli, O.W., 1973. Seismic wave attenuation and magnitude relations for Eastern North America, *J. Geophys. Res.*, **78**, no. 5, p. 876-885.

NIBS, 2002. Earthquake loss estimation methodology, HAZUS-99 service release 2 (SR2) Technical Manual, prepared by the National Institute of Building Sciences for the Federal Emergency Management Agency, Washington, D.C.

Shakal, A.F., Huang, M.–J., Rojahn, C., and Poland, C., 2002. Current building instrumentation programs and guidelines, *in* Proceedings Invited Workshop on Strong-Motion Instrumentation of Buildings, Consortium of Organizations for Strong-Motion Observation Systems (COSMOS), 14-15 November, 2001, Emeryville, California, 5–14.)

Wills, C.J., Petersen, M.D., Bryant, W.A., Reichle, M.S., Saucedo, G.J., Tan, S.S., Taylor, G.C., and Treiman, J.A., 2000. A site-conditions map for California based on geology and shear wave velocity, *Bull. Seism. Soc. Am.*, **90**, S187–S208.

Index of Tags

DAU

Event

EventGeoLocation

GeoLocation

GeoLocationRupture

Private

Processing

RawSeries

Sensor

S (cont.)

T

ThisFile

V