Energistics Unit Symbol Grammar Specification

Version 1.0

Energistics Unit of Measure Standard	A set of resources that defines a standard unit of measure (UOM) dictionary to promote consistent usage, data exchange, and unit conversions. The set includes the base Energistics Unit of Measure Dictionary and related documentation for creating, implementing, and maintaining a UOM dictionary that is patterned after the Energistics dictionary.
Version	**1.0**
Abstract	Defines the Energistics unit symbol grammar, which uses an administered dictionary and algebraic operations to specify, combine, and constrain the unit components from the dictionary to produce unit symbols that represent desired units of measure. Conforms to the intent of RP66 V2.0.
Prepared by	Energistics
Date published	June 2014
Document type	Specification
Keywords:	standards, energy, data, information, process, units of measure

energistics®
The Energy Standards Resource Centre

Document Information	
DOCUMENT VERSION	1.0
Date	June 2014
Language	U.S. English

Acknowledgements

Energistics wishes to thank its member community, the Professional Petroleum Data Management (PPDM™) Association, and the Society of Exploration Geophysicists (SEG) for their contributions to this standard.

Amendment History			
Version	**Date**	**Comment**	**By**
1.0	June 2014	First publication of the standard and the document.	Energistics

Table of Contents

1 Introduction

Accurate use, exchange, and conversion of units of measure (UOM) in upstream oil and gas software are crucial. Errors in units of measure can cause serious problems for the accuracy and integrity of earth and reservoir models and the decisions that are based on those models.

The *Energistics Unit of Measure Standard* (the *UOM Standard*) represents the collaborative work of Energistics, its member community, the Professional Petroleum Data Management (PPDM™) Association, and the Society of Exploration Geophysicists (SEG) to update previous standards and best practices for use of units of measure used in the upstream oil and gas industry. (For a list of previous standards and specifications included, see the *Energistics Unit of Measure Usage Guide*.)

The goal of the *UOM Standard* is to provide the necessary components and guidelines to improve accuracy and consistency of implementation, usage, exchange, and conversion of units of measure related to all upstream activities.

1.1 Audience, Purpose and Scope

This document is intended for IT professionals (e.g., software developers, information architects) who want to implement and maintain in their software a units of measure dictionary patterned after the *Energistics Unit of Measure Dictionary*, the key component of the *UOM Standard*. Software that implements Energistics data-exchange standards (e.g., WITSML™, PRODML™ or RESQML™) MUST implement the *UOM Standard*.

Implementation of the *UOM Standard* includes:

- Conformance to the *Energistics Unit Symbol Grammar Specification* (this document).
- Optional, but strongly recommended use of Energistics quantity classes and unit dimensions, vital information that adds usage context to what would otherwise be a bunch of symbols.

1.1.1 Grammar Overview

A key component of the *UOM Standard*, this specification defines a grammar that constrains how unit components from an administered dictionary are combined and scaled to produce a unit symbol that represents a desired unit of measure.

The grammar defines how to combine the components using the algebraic operations multiplication, division, and exponentiation. The resulting combination may then be scaled by a numeric multiplier, which is expressed in terms of multiplication, division, and exponentiation of integer and decimal numbers.

NOTE: This grammar conforms to the *intent* of the Recommended Practices for Exploration and Production Data Digital Interchange (RP66 V2.0) Unit Model with exceptions as defined in Appendix B, page 13. The RP66 grammar was derived from recommendations from the International System of Units (SI). The RP66 term “symbol” represents the same concept as a “component” in this specification.

1.2 Resource Set

All resources that comprise the *Energistics Unit of Measure Standard* can be downloaded from the Energistics website at http://www.energistics.org/asset-data-management/unit-of-measure-standard. This resource set includes:

Resource	**Description**
Energistics Unit of Measure Dictionary	XML file that is the normative dictionary that developers will implement. Also, an "information-only" spreadsheet version of the normative XML file is provided for as an easier-to-read resource for people. **NOTE**: If any discrepancies exist between the XML and the spreadsheet, the XML is the normative source.
Energistics Unit Symbol Grammar Specification (this document)	Defines the Energistics unit symbol grammar, which uses an administered dictionary and algebraic operations to specify, combine, and constrain the unit components from the dictionary to produce unit symbols that represent desired units of measure. Conforms to the intent of RP66 V2.0.
Energistics Unit of Measure Usage Guide	Intended for IT professionals, provides usage requirements and guidelines for implementing and maintaining a units of measure dictionary patterned after the *Energistics Unit of Measure Dictionary*.
Readme.txt	A text file that explains contents of other supporting files and information contained in the *UOM Standard* download.
Integer Codes	In association with this standard, a set of integer codes has been defined for use in special cases where only binary values can be used. The Society of Exploration Geophysicists uses these codes as part of its new SEGD Rev3.0 format.
Mapping Documents	To aid users and adopters, mappings for POSC v2.2, EPSG v8.1, Open Spirit Unit Dictionary v3.0 and RP66 v1 and v2 are included with this standard.

1.2.1 Additional Resources

It is recommended that anyone who is maintaining a dictionary also consult:

- The *Guide for the Use of the International System of Units (SI)* document (NIST Special Publication 811 – 2008 Edition), which is freely available online in PDF format.
- Recommended Practices for Exploration and Production Data Digital Interchange (RP66 v2) specification (http://w3.energistics.org/RP66/V2/Toc/main.html).

2 Grammar Specification

This chapter provides the details of the grammar.

2.1 Supported Components and Assumptions

The grammar defines these special components, which represent the building blocks for more complex derived symbols.

Component	Definition
atom	A string that should not be further subdivided by parsing.
prefixed-atom	Has one- or two-characters prefixed to the atom to represent a scaling of the underlying concept. Thus, if *s* is the atom for *seconds*, then *ms* is a prefixed atom that represents *milliseconds* (i.e., equivalent to *1e-3 s* where the *1e-3* is inherently defined by the prefix, see Appendix A (page 12).

This grammar specification makes the following assumptions:

- The unit components are not part of the grammar but rather must be defined in a separate dictionary that uses this grammar. For example, the letter "a" is an allowed character but only a specific dictionary can say whether the letter "a" is a supported symbol with a specific meaning.
- The grammar cannot distinguish a character that is intended to represent a prefix from a character that happens to match a prefix, so the components must be explicitly defined by the dictionary. For example, "km" and "knot" both use "k", which is the prefix for "kilo". But the usage in "knot" does not represent a multiplier of 1000. Conversely, there is no way for the grammar to understand that in something like "km" the character "k" is *intended* to represent a multiplier of 1000.
- SI specifies that a prefix must not be used by itself. The person who administers the dictionary must enforce this rule because the grammar cannot. For example, "m" is an allowed prefix but the grammar cannot distinguish this from a symbol such as "m" which a dictionary asserts to represent a metre.
- Any dictionary that uses this grammar must enforce the constraint that any prefixed-atoms must use the allowed prefixes.
- The SI concepts of "base units" or "special units" are represented by atoms. This grammar makes no other assumptions about these SI concepts.

2.2 Unit Symbol Construction Grammar

The following Backus-Naur Form (BNF) grammar defines the syntax for expressing a unit symbol. An example below shows how this grammar is implemented in XML.

- This *::=* means that the token on the left must be replaced with the definition on the right.
- A token definition is terminated by a period token.
- A choice is indicated by the vertical bar token, |.
- A grouping is indicated by parenthesis tokens, ( ).
- Optional items are enclosed in square bracket tokens, [ ].
- Items that may repeat zero or more times are enclosed in curly bracket tokens, { }.
- Any character not enclosed in single quotes is presumed to be a token.
- Whitespace is ignored, unless it is enclosed in single quotes.

```
UnitSymbol ::= [ Multiplier ' ' ] FactorExpression .
FactorExpression ::=
      OneOrMore |
      ( '1' | OneOrMore | Division ) '/' ( Divisor | Division ) .
Division ::= '(' OneOrMore '/' Divisor ')' .
OneOrMore ::= Factor | Factors .
Divisor ::= Factor | '(' Factors ')' .
Factors ::= Factor '.' Factor { '.' Factor } .
Factor ::= UnitComponent [ Exponent ] .
UnitComponent ::= PrefixedAtom | Atom | SpecialAtom [ Qualifier ] .
PrefixedAtom ::= ( SIPrefix | BinaryPrefix ) Atom .
Atom ::= Letter { Letter } [ Qualifier ] .
SpecialAtom ::= '%' | 'inH2O' | 'cmH2O' .
Qualifier ::= '[' [ AT ] QualPart { COMMA QualPart } ']' .
AT ::= '@' .
COMMA ::= ',' .
QualPart ::= LetterOrDigit { LetterOrDigit } .
LetterOrDigit ::= Letter | Digit .
Letter ::= E | LTTR .
LTTR ::=
      'A' | 'B' | 'C' | 'D' |       'F' | 'G' | 'H' | 'I' | 'J' | 'K' |
      'L' | 'M' | 'N' | 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' | 'U' | 'V' |
      'W' | 'X' | 'Y' | 'Z' |
      'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' | 'k' |
      'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r' | 's' | 't' | 'u' | 'v' |
      'w' | 'x' | 'y' | 'z' .
Exponent ::= GtOneDigit |
      '(' ( NonZeroInt '.' FractionalPart | '0' '.' FractionalPart ) ')' .
Multiplier ::= '1' E PowerOfTen [ '/' GtOneInt ] |
               '1' '/' GtOneInt |
                Number [ E PowerOfTen ][ '/' GtOneInt ] .
E ::= 'E' .
PowerOfTen ::= [ '-' ] GtOneInt .
Number ::= GtOneInt |
           NonZeroInt '.' FractionalPart |
           '0' '.' FractionalPart .
GtOneInt ::= GtOneDigit | NonZeroDigit Digit { Digit } .
FractionalPart ::= { Digit } NonZeroDigit .
NonZeroInt ::= NonZeroDigit { Digit } .
Digit ::= '0' | NonZeroDigit .
GtOneDigit ::= '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' .
NonZeroDigit ::= '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' .
SIPrefix ::=
      'y' | 'z' | 'a' | 'f' | 'p' | 'n' | 'u' | 'm' | 'c' | 'd' |
      'da' | 'h' | 'k' | 'M' | 'G' | 'T' | 'P' | 'E' | 'Z' | 'Y' .
BinaryPrefix ::=
      'Ki' | 'Mi' | 'Gi' | 'Ti' | 'Pi' | 'Ei' | 'Zi' | 'Yi' .
```

2.2.1 Example Patterns

This section contains examples of both supported patterns and unsupported/invalid patterns.
Assumption: *a, b,* and *c* are supported components.

2.2.1.1 Supported Patterns

Supported Patterns	Description
`a.b`	Multiplication
`a/b`	Division
`1/a`	Inverse
`1/(a.b)`	Inverse
`1E6 a`	Scientific notation multiplier
`1E-6 a`	Negative power of ten
`1/16 a`	Fractional multiplier
`0.01 a`	Decimal multiplier
`a2`	Squared
`a3`	Cubed
`a9`	Ninth power
`a(0.5)`	Square root
`a.b/c`	Single factor in divisor
`a/(b.c)`	Multiple factors in divisor
`(a/b)/(c/b)`	Maximum division nesting
`(a3.c/b2)/(c7/(a.b))`	Complex expression
`a[1]`	Atom with qualifier
`a[1]2`	Atom with qualifier squared
`a[@5b,9c]`	Atom with multiple conditions

2.2.1.2 Unsupported/Invalid Patterns

Pattern	Description
`a/b.c`	Missing parenthesis around factors in divisor
`a/b/c`	Missing parenthesis around divisor containing divisor
`1/(b)`	Parenthesis around single factor in denominator
`(1/(b))`	Unnecessary outer parenthesis
`(a)`	Parenthesis around factor(s)
`(a.b)/b`	Parenthesis around factor(s) in numerator
`9a`	Missing space after multiplier
`9`	Missing component

Pattern	Description
`9   a`	Multiple spaces after multiplier
`.5 a`	Missing zero to left of decimal point
`5.10 a`	Fractional part contains trailing zero
`5.0 a`	Fractional part is zero
`1E0 a`	Power of ten is zero
`1E1 a`	Power of ten is one
`1E-1 a`	Power of ten is negative one
`1E0.5 a`	Power of ten is non-integer
`2/1 a`	Numerator is one
`2/0 a`	Numerator is zero
`-10 a`	Multiplier is negative
`0 a`	Multiplier is zero
`1 a`	Multiplier is one
`a 9`	Multiplier at end
`a(2)`	Exponent within parenthesis is whole number
`a-2`	Exponent is negative
`a0`	Exponent is zero
`a1`	Exponent is one
`a10`	Exponent greater than nine
`(a.b)2`	Exponent outside of parenthesis (use '(a2.b2)')
`010 a`	Leading zero
`1E05 a`	Leading zero
`+10 a`	Plus sign
`1E+6 a`	Plus sign
`1e6 a`	Lower case E
`a[]`	Empty qualifier (invalid component)
`a[`	Unpaired bracket (invalid component)
`a[x,y]`	Punctuation within qualifier (invalid component)
`a[x y]`	Space within qualifier (invalid component)
`a[b@c]`	@ symbol not at beginning of qualifier
`a[b,,c]`	Consecutive commas
`a[,b]`	Comma at beginning

2.3 Guidelines for Constructing Unit Symbols

The following guidelines are suggested but not required.

- For multipliers between *1E3* and *0* (inclusive), use an integer number instead of an exponential multiplier. For example, use *100 ft* not *1E2 ft*.
- For multipliers between *1E-2* and *1E-3* (inclusive), use a decimal fraction instead of an exponential number. For example, use *0.01 ft* not *1E-2 ft*.
- For an irrational multiplier, use whole numbered ratios (where reasonable) instead of decimal fractions. For example, use *1/3 ft* not *0.33333333 ft*. For other fractions, use a decimal fraction such as *0.01 ft*.
- For numbers that would otherwise contain many zeros, use exponential format (e.g., *1E6* or *2.3456E-6*). That is, do not use values such as *0.00000023456* or *2345600000*.

Appendix A. Prefixes

The following table lists the scale factor associated with each prefix. Note that the two-character prefixes represent binary multipliers (i.e., *2**N*), which are not allowed in SI-compliant unit symbols. The common names are usable worldwide. Other names (such as billion) tend to have different meanings around the world and should be avoided.

Prefix	Name	Scale Factor	Common Name
y	yocto	1e-24	
z	zepto	1e-21	
a	atto	1e-18	
f	femto	1e-15	
p	pico	1e-12	
n	nano	1e-9	
u	micro	1e-6	millionth of
m	milli	1e-3	thousandth of
c	centi	1e-2	hundredth of
d	deci	1e-1	tenth of
da	deka	1e+1	ten
h	hecto	1e+2	hundred
k	kilo	1e+3	thousand
M	mega	1e+6	million
G	giga	1e+9	thousand million
T	tera	1e+12	million million
P	peta	1e+15	thousand million million
E	exa	1e+18	million million million
Z	zetta	1e+21	
Y	yotta	1e+24	
Ki	kibi	1024	
Mi	mebi	1048576	
Gi	gibi	1073741824	
Ti	tebi	1099511627776	
Pi	pebi	1125899906842624	
Ei	exbi	1152921504606846976	
Zi	zebi	1180591620717411303424	
Yi	yobi	1208925819614629174706176	

Appendix B. Differences From RP66

The grammar conforms to the *intent* of the Recommend Practice 66 (RP66) V2.0 Unit Model (http://w3.energistics.org/RP66/V2/Toc/main.html). In particular, the grammar represents a subset of the RP66 V2.0 grammar with limited extensions for atoms. The RP66 term of “symbol” represents the same concept as a “component” in this specification. This document also has these semantic changes:

- Extends the grammar to allow an atom to have a qualifier. For example, there are many variations of the concept of a yard (3 feet) depending on which definition of foot was assumed.
- Extends the grammar to allow the following special atoms: *inH2O* and *cmH2O*.
- Constrains the use of *%*. The percent sign can only be specified by itself or with a qualifier. That is, the percent sign inherently represents a special concept.
- Disallows a symbol that is only a space. The concept of unitless is more appropriate for a property.
- Constrains the multiplier:
  - Disallows a symbol that is only a multiplier. This presumes that an atom such as *Euc* represents *1* such that something like *99 Euc* would be possible.
  - Disallows a fractional denominator.
  - Disallows a denominator of *1*.
  - Disallows a power of ten of *1* or *-1*.
  - Disallows a value of *1*.
- Disallows a single factor within parenthesis.
- Disallows parenthesis around factors in a numerator.
- Constrains the level of nested divisions.
- Constrains a component exponent to a single digit between *2* and *9* or to a factional decimal value within parenthesis.
- Disallows a negative exponent for a component (redundant to divide and can be confused with subtraction).
- Disallows an offset. The offset is more appropriate as part of an expression that may be used to capture conversion information:

  ```
  DefinitionExpression ::= UnitSymbol [',' ' ' Offset] .
  Offset ::= ['-'] Multiplier .
  ```

- Implies that a prefix cannot be specified without an associated atom.