# ConversionCalc Library

An easy-to-implement function library providing generic numeric conversions in a wide array of possible scenarios for engineering and practical applications.

{Line}

**New!**: CSS Measurement Units and event-based external value resolution!

See **Source/ConversionCalcExample/Program.cs** methods *Run*, *converter\_ResolveBaseToValue*, and *converter\_ResolveValueToBase* for examples of how to use relative CSS unit values.

{Line}

## Table of Contents

You can jump to any section of this page from the following list.

* [25 Years of .NET](#_25_Years_of).
* [Yet Another Value Converter](#_Yet_Another_Value).
* [A Dictionary-Driven Converter](#_A_Dictionary-Driven_Converter).
* [Installation](#_Installation).
* [Usage Notes](#_Usage_Notes).
* [Example 1 - Teaspoons to Cups](#_Example_1_-).
* [Example 2 - Intercategorical Conversions](#_Example_2_-).
* [Appendix - Built-In Measurement Categories](#_Appendix_-_Built-In).
* [Appendix - JSON Catalog Format](#_Appendix_-_JSON).

{Line}

## 25 Years of .NET

Although .NET doesn't officially turn 25 until February 13, 2027, I'm starting the celebration a little early.

To commemorate 25 years since the public release of the .NET framework, I'm open sourcing this and several other of my long-lived libraries and applications. Most of these have only previously been used privately in our own internal company productivity during the early 21st century but I hope they might find a number of new uses to complete the next 25 years.

I have every intention of keeping these libraries and applications maintained, so if you happen to run into anything you would like to see added, changed, or repaired, just let me know in the Issues section and I'll get it done for you as time permits.

{Line}

Sincerely,

**Daniel Patterson, MCSD (danielanywhere)**

{Line}

## Yet Another Value Converter

The main use of this conversion library has always been to allow the user to enter any kind of value in a unit of measurement that makes sense to them, then to successfully respond with real world results in a different unit of measurement. A couple of great examples of this kind of conversion are:

* If the user asks for an appointment "3 weeks from next Tuesday", I could easily calculate that would be September 5.
* If the user knows that they are making 15 batches of cupcakes and that the single recipe for cupcakes calls for 1.5 cups of flour, then two inter-categorical conversions can quickly determine that they will need more than 5lb of flour before starting.

In fact, this library serves to be more of a universal engineering converter for practical everyday applications than a computing converter in the literal sense of the word, which I believe is where I deviated from the norm.

I absolutely hate re-inventing the wheel all the time, so as usual, when I began on this library many years ago, I had definitely already searched high and low for generic value converters. Out of the closest libraries I could find, most were completely hard-coded, but if not, they would probably utilize some strange look-up table, not be extensible in any way, be mainly concerned with the conversion of binary data from one datatype to another, or some other unique makeup that I either didn't want or need.

However, as I mentioned, this library has also been under active maintenance for several years - this code has existed in one form or another since 1997 to be exact, starting as VBA clear back in the days of Microsoft Office Automation, so there weren't too many generic, non-proprietary things along these lines that even existed in that time.

Let's be clear, though. Contrary to the modern IT way of thinking (mainly proliferated by people who don't ever directly develop technology), software doesn't ever age - it just either works perfectly or it doesn't. I maintain that this converter won't seem old or outdated to you in any way. You are likely to find only the most up-to-date methods and techniques, and somewhat efficient use of your definitions throughout each conversion. Even though I admit there's always room for improvement, you can be sure not to experience any of the clunkiness in this library that so many projects seem to carry as constant baggage. This package has already been around the block more than a few times and should be ready for anything you may need to throw at it.

Whatever the library might lack in forward momentum or future vision, it makes up for by far in consistency, accuracy, and outright extensibility. If you have a set of specialized conversions to implement in your any-format-accepted textbox and don't mind pulling out a couple of regular expressions once in a while to tend to inter-category complex equational traffic management, then this library is probably for you.

I've made sure to provide a few different kinds of examples, too, that might help you decide how you might want to use the functionality. Just start an Issue in the toolbar above if you need to see a few more examples or if you run into a problem where something isn't working as expected.

{Line}

## A Dictionary-Driven Converter

In most cases, you might never need to make any changes to the base code. This library allows you to extend or even entirely change the types of calculations that can be performed by maintaining your own JSON unit catalog. Although you can either add to or replace the calculation definitions at will, the built-in calculations are meant to be helpful in nearly every practical scenario. I'll likely be adding more all the time to work with our own projects.

{Line}

See the [JSON Catalog Format](#_JSON_Catalog_Format) appendix at the bottom of this page for more information on editing the dictionary data or the file Data/DefaultData.json, in this project, for practical examples of how the conversion definitions are structured.

{Line}

## Installation

You can include this library in any .NET project using any supported programming language or target system. This library compiles as **.NET Standard 2.0** and is available in **NuGet** as

{Center}{Bold}{Big}Dan's ConversionCalc Library{/Big}{/Bold}{/Center}

{Line}

**Instructions For Installation**

In **Visual Studio Community** or **Visual Studio Professional** editions:

* Right-click your project name in **Solution Explorer**.
* From the context menu, select **Manage NuGet Packages**.
* Click **Browse**.
* In the **Search** textbox, type **Dan's ConversionCalc Library**.
* Accept the license agreement.
* In your code add the header line **using ConversionCalc;**

{Line}

In **Visual Studio Code**:

* Run the command **NuGet: Add NuGet Package** (typically [Ctrl][Shift][P]).
* If there are multiple projects in the solution, select the open project to which the package will be assigned.
* In the **Search** textbox, type **Dan's ConversionCalc Library**.
* Select the package.
* Select the version you wish to apply.

{Line}

## Usage Notes

For the full documentation of this library, you can access the API in HTML format at the associated GitHub user page library <https://danielanywhere.github.io/ConversionCalc>.

The main object in this library is instantiated from the Converter class. In general, create a new instance of Converter and call its various methods, as in the following example.

{CodeBlock,cs,ConverterUsage.cs}

{Line}

## Example 1 - Teaspoons to Cups

This easy example shows you how to convert 25,479 teaspoons to cups, using the implicit domain lookup feature.

{CodeBlock,cs,Example1.cs}

{Line}

## Example 2 - Intercategorical Conversions

In this next admittedly preconceived example, while slightly more involved than the previous one, I show you how to consistently handle multiple conversions for different domains based upon the single user input of 1000 km/sec to mi/hr.

To make it interesting, instead of using the built-in conversions that could solve this query in one pass, the distance units are going to be separated from the time units, and different conversions will be performed in each domain. Notice that in the second conversion, we are flipping the order of the source and target units in the call to **converter.Convert** because the values in the second calculation are denominators of a division (ie miles/hr as opposed to miles\*hr).

{CodeBlock,cs,Example2.cs}

{Line}

## Appendix - Built-In Measurement Categories

Several units of measure have been included by default to provide normal conversions for many of life's daily demands.

The individual unit measurements are partitioned into the following major measurement categories, or domains, to which they apply.

[Bracketed] values next to a unit name indicate other names by which that unit is equally recognized.

* [Angles](#_Angles). Converting angles between radians and degrees.
* [Binary Data](#_Binary_Data). Working with binary-oriented conversions.
* [Circles](#_Circles). Converting between the characteristics of a circle.
* [Count](#_Count). Common methods for counting and enumerating objects, like dozen, pair, etc.
* [Density](#_Distance). The ratio of a material's volume to its mass.
* [Distance](#_Distance). Various conversions of length ranging from nanometers to light years.
* [Power](#_Power). Conversions in energy consumption.
* [Pressure](#_Pressure). Conversions for pressures in different contexts and settings.
* [Temperature](#_Temperature). Conversions of temperature on various scales.
* [Time](#_Time). Conversions and comparisons of time, from the billionth of a second to the billions of years.
* [Volume](#_Volume). Conversions of space occupied by a solid, liquid, or gas.
* [Weight](#_Weight). General conversions of weight and mass.

{Line}

### Angles

The angles category is sparse in this version with only a conversion path between radians and degrees.

{Line}

#### Units

The following units are currently available in this domain.

* **degree**. [degrees, deg]. With its roots in ancient civilizations, like the Sumerians and Babylonians, who used a base-60 numbering system, a degree is 1/360 of a full revolution. A circle divided into 360 degrees not only aligns it generally with the number of days in a year but offers a greater than usual number of even divisors, making it convenient for people to easily divide a circle into one of several patterns of even parts. Most angles are expressed by humans in the form of degrees.
* **radian**. [radians, rad]. A radian, or one radius length along the circumference of a circle, is an ideal way by which to perform angular calculations because of the radius' relationship to the circle through PI. Most trigonometric calculations are performed directly in radians.

{Line}

### Binary Data

This version of the binary data domain maintains the original processor-agnostic binary theory that the basic sizes of all units are universally permanent and static.

{Line}

In this version of that system, the following relations always hold true:

|  |  |
| --- | --- |
| Entity | Expression |
| BIT | 1 binary unit |
| BYTE | 8 BITS |
| WORD | 2 BYTES |
| DOUBLE WORD | 2 WORDS |
| QUAD WORD | 4 WORDS |

{Line}

#### Units

The following discrete units are available in this domain.

* **bit**. [b]. The smallest unit of binary information: 1 or 0.
* **bool**. [bitmask, Boolean]. True or False value. In Assembly Language, this data type can often be represented by a single bit, but in higher level languages, the data type is often represented by the size of a register, which depends on the processor architecture. In this version of the definition, *\*bool\** is the size of a byte.
* **byte**. [bytes, B, char, character, Int8, Integer8]. 8 bits functioning as a single accessible group. Although the size of a byte may seem to be arbitrary, it was chosen early on in the game both because of the balance it can strike between complexity and efficiency, and because 8 bits is perfectly compatible with the range of characters that had been defined by ASCII for the standardization of the teletype. All but the very first popular microprocessors have used some multiple of 8 bits in their processor architecture, namely 8-bit, 16-bit, 32-bit, and 64-bit, although that pattern in design also represents the effective doubling in size at every turn and indicating that the processor plateau will be at 128 bits.
* **datetime**. [timestamp]. A value representing a date and time. In many cases, a datetime is a floating-point value where the integer part represents the number of days since a specific epoch (like January 1, 1970), and the fractional part represents the time as a fraction of 24 hours. In other cases, datetime could be represented by a very large whole number expressing the total number of milliseconds elapsed since a notable epoch. In this version, the values of datetime and timestamp are 8 bytes, although it is sometimes possible for timestamp to use a much smaller value to capture a moment in immediate time.
* **double**. [double floating point]. The double floating-point number, often just referred to as a "double", is a way to represent very large or very small numbers with a lot of precision. This version defines a double's size at 8 bytes.
* **double word**. [DW]. Not to be confused with a double floating-point number, a double word is an integer value consisting of two words or 4 bytes. 32-bit numbers are often handwritten as two sets of four hexadecimal digits each, annotated with a DW to indicate the overall size.
* **gigabit**. [gigabits, gb]. Considered to be generally in the billions range, one gigabit is 1,073,741,824 bits or 134,217,728 bytes. The reason this figure is not represented as an even decimal number is that because the binary numbering system is ranked by powers of two, the "even" values in the system follow the general pattern of 1, 2, 4, 8, etc. In this chain, value 1024 is equal to 1K, the value 1024K is equal to 1M, and the value 1024M is equal to 1G.
* **gigabyte**. [gigabytes, gB]. Considered to be in the conceptual billions range, one gigabyte is 1,073,741,824 bytes or 8,589,934,592 bits. In 2024, the sizes of consumer-level RAM memories are in the low gigabyte range.
* **Integer16**. [Int16]. The 16-bit integer is rarely used today except in specialized cases, but in the days of the Intel 80286 processor, represented the native 'int' value for the architecture.
* **Integer32**. [int, Int32]. A 32-bit integer or whole number value. In many logic processing systems, this size of value still represents the native 'int' value, although most architectures have moved on to 64-bit register widths. The reasoning behind retaining the 32-bit natural 'int' could be that backward compatibility is promoted while optimizing the use of memory, being that practical working values are rarely expected to approach or overrun the 4,294,967,295 total range of a single Int32 value.
* **Integer64**. [Int64]. A 64-bit integer or whole number with a total range of 18,446,744,073,709,551,616 possible byte-oriented values, using either a signed or unsigned representation.
* **ip packet**. [tcp packet, udp packet]. Although TCP and UDP packets are significantly different in their data patterns and uses, they are typically both prepared and routed at a size of 1500 bytes to promote consistency in traffic.
* **kilobit**. [kilobits, kb]. A kilobit is 1024 bits or 128 bytes.
* **kilobyte**. [kilobytes, kB]. Conceptually occupying the thousands range, a kilobyte is 1024 bytes.
* **megabit**. [megabits, mb]. One megabit is 1024kb or 1,048,576 bits.
* **megabyte**. [megabytes, mB]. A megabyte represents the conceptual millions range, composed of 1,048,576 bytes.
* **nibble**. [nibbles]. Although a nibble is typically no longer referred to except in specific real-time identification references during troubleshooting or diagnostics, it maintains a permanent importance as a four-bit number that perfectly quantifies the value of a hexadecimal digit, having the range from 0-9 and A-F. Written in hexadecimal, one byte is represented by two hex digits.
* **quad word**. [quad words, QW]. A quad word is four words, 8 bytes, or 64 bits. When written by hand in hexadecimal, this value is normally represented by four groups of four hex digits and annotated with QW.
* **single**. [float]. The single floating-point number, otherwise known as a float, is similar in nature to a double floating-point value, but uses only 4 bytes for its definition, limiting the overall precision to less-than-astronomical values.
* **terabit**. [terabits, tb]. One terabit is 1024g bits.
* **terabyte**. [terabytes, tB]. A terabyte, or 1024gB, corresponds to sizes in trillions of bytes. In 2024, the sizes of consumer-level hard drives are in the low terabyte range.
* **word**. [words, W]. A system-agnostic word is two bytes or 16 bits. This size is often seen represented in the typical groupings of four hexadecimal characters per expression.

{Line}

### Circles

A circle is a fascinating geometric shape with several key parts that are all interconnected. The radius is the distance from the center of the circle to any point on its edge. The diameter is twice the length of the radius, stretching from one edge of the circle to the opposite edge, passing through the center. The circumference is the total distance around the circle, which can be calculated using the formula C=2𝜋r, where r is the radius. The area of the circle, representing the space enclosed within its boundary, is given by the formula A=𝜋r2. Each of these parts plays a crucial role in defining the properties and measurements of a circle.

{Line}

#### Units

Here are the units in this version of the Circles category.

* **area**. The area of a circle is 𝜋r2.
* **circumference**. The circumference, or length of border, of a circle is 2𝜋r.
* **diameter**. The diameter of a circle is 2r.
* **radius**. The radius of a circle is the distance from its center to the edge.

{Line}

### Count

The Count domain translates different general methods for counting objects. Certain counting terms that have no discrete basis, like *set* and *fleet* have been omitted.

{Line}

#### Units

The following discrete units have been built-in to the count domain.

* **deca**. A prefix indicating a multiple of ten.
* **dozen**. [doz]. 12.
* **each**. [ea]. Standard terminology for a single item.
* **giga**. A prefix indicating a multiple of one billion.
* **gross**. 144.
* **hundred**. Multiple of 100.
* **kilo**. [k]. A prefix indicating a multiple of one thousand.
* **mega**. [m]. A prefix indicating a multiple of one million.
* **package**. [pkg]. Multiple items grouped together.
* **pair**. [pr]. Two matching items.
* **peta**. A prefix indicating a multiple of one quadrillion.
* **ream**. 500.
* **score**. 20.
* **tera**. A prefix indicating a multiple of one trillion.
* **thousand**. A multiple of 1000.

{Line}

### Density

The density of a material is the ratio of its mass to the volume it occupies.

Density is influenced by the mass of the atoms, their dimensions, and their physical arrangement.

Density is calculated by dividing the mass of the material by its volume, represented by the formula: D=m/v.

{Line}

#### Units

The following default units of measure have been included in this domain.

* **g/cc**. Grams per cubic centimeter.
* **g/cm3**. Grams per cubic meter.
* **g/l**. Grams per liter.
* **kg/m3**. Kilograms per cubic meter.

{Line}

### Distance

Distance can be measured in various units, from nanometers (nm) for tiny objects to light years for astronomical distances. A millimeter is useful for small-scale measurements, while a light year, about 9.46 trillion kilometers, measures the vast distances light travels in a year. These units help us describe everything from the microscopic to the cosmic scale.

{Line}

#### Units

The distance domain includes the following units of measure. The concrete CSS units pc, pt, and px are included on this list. For an example of how to implement the relative CSS units ch, em, ex, rem, vh, vmax, vmin, and vw, please see the methods 'Run', 'converter\_ResolveBaseToValue' and 'converter\_ResolveValueToBase' in Source/ConversionCalcExample/Program.cs.

* **AU**. Astronomical Units.
* **AU/day**. Astronomical Units per day.
* **AU/hr**. Astronomical Units per hour.
* **AU/min**. Astronomical Units per minute.
* **AU/sec**. Astronomical Units per second.
* **AU/week**. Astronomical Units per week.
* **AU/year**. [AU/yr]. Astronomical Units per year.
* **C**. [C/sec] The universal constant representing the speed of light in one second, in meters.
* **C/day**. The distance light travels in one day, in meters.
* **C/hr**. The distance light travels in one hour, in meters.
* **C/min**. The distance light travels in one minute, in meters.
* **C/wk**. [C/week]. The distance light travels in one week, in meters.
* **C/yr**. [C/year]. The distance light travels in one year, in meters.
* **cm**. [centimeter, centimeters]. Centimeter.
* **cm/hr**. [centimeter/hr, centimeters/hr]. Centimeters per hour.
* **cm/min**. [centimeter/min, centimeters/min]. Centimeters per minute.
* **cm/sec**. [centimeter/sec, centimeters/sec]. Centimeters per second.
* **ft**. [foot, feet]. Feet.
* **ft/hr**. [foot/hr, feet/hr]. Feet per hour.
* **ft/min**. [foot/min, feet/min]. Feet per minute.
* **ft/sec**. [foot/sec, feet/sec]. Feet per second.
* **in**. [inch, inches]. Inches.
* **in/hr**. [inch/hr, inches/hr]. Inches per hour.
* **in/min**. [inch/min, inches/min]. Inches per minute.
* **in/sec**. [inch/sec, inches/sec]. Inches per second.
* **km**. [kilometer, kilometers]. Kilometers.
* **km/hr**. [kilometer/hr, kilometers/hr]. Kilometers per hour.
* **km/min**. [kilometer/min, kilometers/min]. Kilometers per minute.
* **km/sec**. [kilometer/sec, kilometers/sec]. Kilometers per second.
* **m**. [meter, meters]. Meters.
* **m/hr**. [meter/hr, meters/hr]. Meters per hour.
* **m/min**. [meter/min, meters/min]. Meters per minute.
* **m/sec**. [meter/sec, meters/sec]. Meters per second.
* **mi**. [mile, miles]. Miles.
* **mi/hr**. [mile/hr, miles/hr, mph, MPH]. Miles per hour.
* **mi/min**. [mile/min, miles/min]. Miles per minute.
* **mi/sec**. [mile/sec, miles/sec]. Miles per second.
* **mm**. [millimeter, millimeters]. Millimeters.
* **mm/hr**. [millimeter/hr, millimeters/hr]. Millimeters per hour.
* **mm/min**. [millimeter/min, millimeters/min]. Millimeters per minute.
* **mm/sec**. [millimeter/sec, millimeters/sec]. Millimeters per second.
* **nm**. [nanometer, nanometers]. Nanometers.
* **nm/sec**. [nanometer/sec, nanometers/sec]. Nanometers per second.
* **pc**. [pica, picas]. CSS measurement unit. 1 pica = 12 points.
* **pt**. [point, points]. CSS measurement unit. 1 point = 1/72 inch.
* **px**. [pixel, pixels]. CSS measurement unit. 1 pixel = 1/96 inch on a general purpose LCD display.

{Line}

### Power

Power measurements help us understand the rate at which energy is used or transferred.

{Line}

#### Units

The following power units are defined in the built-in catalog.

* **horsepower**. [hp]. Horsepower.
* **joule/sec**. [j/sec]. Joules per second.
* **n/sec**. [newton/sec, newtons/sec]. Newtons per second.
* **newton-meter/sec**. [newton-meters/sec]. Newton-meters per second.
* **watt**. [watts, w, W]. Watts.

{Line}

### Pressure

Pressure measurements are essential in various fields, from engineering to meteorology. These units help us quantify and compare pressures in different contexts, ensuring accurate and consistent measurements across various industries.

{Line}

#### Units

Following are the predefined units available in the pressure category.

* **atm**. Atmospheres.
* **bar**. An obsolete unit of measurement, introduced by Norwegian meteorologist Vilhelm Bjerknes in 1909, now replaced by SI units.
* **pa**. [pascal, pascals]. Named after famed French mathematician Blaise Pascal, the pascal is a standard based equal to one newton per square meter.
* **psi**. [psia]. Pounds per square inch, absolute. This is the absolute pressure relative to a perfect vacuum.
* **psig**. Pounds per square inch, gauge. This is the measured pressure relative to local air pressure.
* **joule**. [j]. The unit used to measure energy or work representing the energy transferred when a force of one newton acts through a distance of one meter, in SI units.

{Line}

### Temperature

Temperature measurements are crucial in various fields, using different scales like Fahrenheit (°F), Celsius (°C), and Kelvin (K). These scales help us accurately measure and communicate temperature in diverse applications, from weather forecasting to scientific research.

{Line}

#### Units

Here are the default temperature units built-in on this version.

* **Celsius**. [Centigrade, C].
* **Fahrenheit**. [F].
* **Kelvin**. [K].

{Line}

### Time

Time can be measured in various units, each suited for different scales. For instance, a nanosecond (ns) is one-billionth of a second, often used in high-speed computing and electronics. These units allow us to measure and understand time from the incredibly brief to the vast expanses of history.

{Line}

#### Units

These units have been provided by default in the time domain.

* **day**. [days, d]. The period of time it takes the Earth to rotate once upon its axis, relative to the Sun.
* **hr**. [hour, hours, h]. Originally, this value was equal to the time it takes the Earth to rotate 15 degrees on its axis, relative to the Sun, and corresponded exactly to the lines of longitude on a globe, as well as roughly to time-zones established for the railroads during the Industrial Age. The modern definition of an hour, however, is defined as being the range from 3599 to 3601 seconds to accommodate the need for occasional leap second, because, you know, people don't screw up the most otherwise consistent static references. Seriously, for all practical intents and purposes, one hour is always 60 minutes or 3600 seconds.
* **hz**. [hertz]. Number of cycles in one second, the hertz was named after Heinrich Hertz, the German physicist who, in 1886, was able to prove the existence of the electromagnetic waves that had earlier been predicted by James Maxwell.
* **rpm**. [revolutions per minute, rotations per minute]. Revolutions per minute.
* **sec**. [second, seconds, s]. The second, originally a 1/60 derivative of one minute, is now the basis of the measurement of time, based upon the resonant frequency at which the element Cesium is able to absorb microwave energy, which is 9,192,631,770 times per second. In other words, one second today is equal to 9,192,631,770 Cesium absorption cycles, which is accurate to 1/3,000,000 of one second per year, but which has no dependency on the position of the Earth to the Sun.
* **min**. [minute, minutes, m]. The original value of a minute corresponded directly to the amount of time it took Earth to rotate 0.25 degrees on its axis, relative to the Sun, and was tied intrinsically both to the measurement of a second and to 1/60 of an hour. Today, one minute is equal to 60 seconds, regardless of the position of the Earth.
* **mo**. [month, months, M]. The story of the month is a long and convoluted one. Originally based upon the synodic orbits of the Moon around the Earth, each of which take approximately 29.5 Earth days, the "moons" served as important milestones in the year, each representing different times to accomplish important tasks like planting, harvesting, tending, migrating, and even buckling down for cold winter months in certain places. Since the Moon is not in perfect synchronization with the Earth's orbit around the Sun, however, it eventually lost the favor of Roman emperors who thought they were gods. They ended up demoting it permanently to the role of *random inconsequential ball orbiting the Earth* replacing it with their own modern month of 1/10 of an Earth year, which supported their contemporary metric system. Who even knows how you could justify 36.425 days of a month as being a nice, round number? Eventually, under constant pressure from special interest groups, they reluctantly extended the monthly calendar to include 12 months in an Earth year to align a little more closely with the lunar cycle; while insisting it was lunacy to revert to that old pattern. Although humans still use the month unit of measurement, it might be more because of exhibiting hoarding behaviors or trying to hold onto some past golden age that never existed than any practical purpose it actually serves. Humans now work all the time performing the exact same tasks every day, all summer, all winter, and all of the seasons in between.
* **ms**. [millisecond, milliseconds]. 1/1000 of one second.
* **ns**. [nanosecond, nanoseconds]. One billionth of one second.
* **ps**. [picosecond, picoseconds]. One trillionth of one second.
* **us**. [microsecond, microseconds]. One millionth of a second. In this version, the small letter u is used to represent µ, the symbol for lowercase Greek letter mu.
* **wk**. [week, weeks, w]. Oddly, although it means absolutely nothing in astronomical or geometrical terms, the period of one week is a handy and consistent tool for calculations in time that span from the day to the year. Because a week is always 7 days and because there are always 52 weeks in a year, conversions spanning this range can often skip the inconsistencies found in the month range.
* **yr**. [year, years, y]. One year is the period of time it takes for the Earth to orbit the Sun once, which is about 365.2422 days, because the rotational speed of the Earth doesn't synchronize perfectly with the orbital speed of the Earth around its local star and why would it, when you think about it for 9,192,631,770 Cesium absorption cycles or so?

{Line}

### Volume

Volume can be measured using various units depending on the context. These units help us quantify and compare volumes in different settings, from everyday cooking to industrial applications.

{Line}

#### Units

These units of measurement have been preset on the built-in data catalog.

* **cc**. [cm3, cubic centimeter, cubic centimeters]. Cubic area one centimeter in length along each of its X, Y, and Z dimensions.
* **cc/hr**. [cm3/hr, cubic centimeter/hr, cubic centimeters/hr]. Cubic centimeters per hour.
* **cc/min**. [cm3/min, cubic centimeter/min, cubic centimeters/min]. Cubic centimeters per minute.
* **cc/sec**. [cm3/sec, cubic centimeter/sec, cubic centimeters/sec]. Cubic centimeters per second.
* **cf**. [cubic foot, cubic feet]. Cubic area of one foot in length along each of X, Y, and Z dimensions.
* **cf/hr**. [cubic foot/hr, cubic feet/hr]. Cubic feet per hour.
* **cf/min**. [cubic foot/min, cubic feet/min]. Cubic feet per minute.
* **cf/sec**. [cubic foot/sec, cubic feet/sec]. Cubic feet per second.
* **cu/in**. [cu.in., cubic inch, cubic inches]. Cubic area of one inch in length along each of X, Y, and Z dimensions.
* **cu/in/hr**. [cu.in./hr, cubic inch/hr, cubic inches/hr]. Cubic inches per hour.
* **cu/in/min**. [cu.in./min, cubic inch/min, cubic inches/min]. Cubic inches per minute.
* **cu/in/sec**. [cu.in./sec, cubic inch/sec, cubic inches/sec]. Cubic inches per second.
* **cup**. [cups]. 8 fluid ounces.
* **cup/sec**. [cups/sec]. Cups per second.
* **fl.oz.**. [oz, ounce, ounces, fluid ounce, fluid ounces]. A unit of liquid volume exactly equal to 29.5735295625 milliliters.
* **fl.oz./hr**. [oz/hr, ounce/hr, ounces/hr, fluid ounce/hr, fluid ounces/hr]. Fluid ounces per hour.
* **fl.oz./min**. [oz/min, ounce/min, ounces/min, fluid ounce/min, fluid ounces/min]. Fluid ounces per minute.
* **fl.oz./sec**. [oz/sec, ounce/sec, ounces/sec, fluid ounce/sec, fluid ounces/sec]. Fluid ounces per second.
* **gal**. [gallon, gallons]. A volume of exactly 3.78541 liters.
* **gal/sec**. [gallon/sec, gallons/sec]. Gallons per second.
* **l**. [liter, liters]. Otherwise known as 1 cubic decimeter, the liter is a metric standard unit equal to 1000 milliliters. From 1901 to 1964, the liter was associated directly with 1 kilogram of pure water at its maximum density at a temperature of 3.98 degrees C. However, when it was discovered that water also had variabilities according to pressure, purity and isotopic uniformity, all meaningful references were discarded forever so nobody in the future would understand how different things can be directly related to each other in a practical, hands-on world.
* **l/hr**. [liter/hr, liters/hr]. Liters per hour.
* **l/min**. [liter/min, liters/min]. Liters per minute.
* **l/sec**. [liter/sec, liters/sec]. Liters per second.
* **m3**. [cubic meter, cubic meters]. Cubic area of one meter in length along each of its X, Y, and Z dimensions.
* **m3/hr**. [cubic meter/hr, cubic meters/hr]. Cubic meters per hour.
* **m3/min**. [cubic meter/min, cubic meters/min]. Cubic meters per minute.
* **m3/sec**. [cubic meter/sec, cubic meters/sec]. Cubic meters per second.
* **ml**. [milliliter, milliliters]. A milliliter is a unit of area previously equal to exactly one gram of pure water in its maximum density at 3.98 degrees C.
* **ml/hr**. [milliliter/hr, milliliters/hr]. Milliliters per hour.
* **ml/min**. [milliliter/min, milliliters/min]. Milliliters per minute.
* **ml/sec**. [milliliter/sec, milliliters/sec]. Milliliters per second.
* **scfm**. [standard cubic feet per minute]. Standard Cubic Feet per Minute is a measurement of gas flow rate, expressed as the volume of gas at a standardized temperature and pressure, often defined as 60 degrees F (15.5 degrees C) and 14.7 pounds per square inch absolute (psia), representing a fixed number of gas moles regardless of actual flow conditions. The expected basis values may change among industries.
* **tbsp**. [tablespoon, tablespoons]. One tablespoon is equal to 15 milliliters.
* **tbsp/sec**. [tablespoon/sec, tablespoons/sec]. Tablespoons per second.
* **tsp**. [teaspoon, teaspoons]. One teaspoon is equal to 5 milliliters.
* **tsp/sec**. [teaspoon/sec, teaspoons/sec]. Teaspoons per second.

{Line}

### Weight

Weight can be measured using various units depending on the context. These units help to accurately measure and compare weights in various applications, from cooking to industrial use."

{Line}

#### Units

The following list contains units of weight-based measure included in the built-in conversion dictionary.

* **g**. [gram, grams]. A gram is formerly equal to the weight of one milliliter of pure water in its maximum density at 3.98 degrees C.
* **g/hr**. [gram/hr, grams/hr]. Grams per hour.
* **g/min**. [gram/min, grams/min]. Grams per minute.
* **g/sec**. [gram/sec, grams/sec]. Grams per second.
* **kg**. [kilogram, kilograms, kilo, kilos]. 1000 grams, which was formerly equal to the weight of one liter of pure water in its maximum density at 3.98 degrees C.
* **kg/hr**. [kilogram/hr, kilograms/hr, kilo/hr, kilos/hr]. Kilograms per hour.
* **kg/min**. [kilogram/min, kilograms/min, kilo/min, kilos/min]. Kilograms per minute.
* **kg/sec**. [kilogram/sec, kilograms/sec, kilo/sec, kilos/sec]. Kilograms per second.
* **lb**. [lbs, pound, pounds]. The pound, or International Aviodupois Pound, is defined as being exactly 0.45359237 kilograms.
* **lb/hr**. [lbs/hr, pound/hr, pounds/hr]. Pounds per hour.
* **lb/min**. [lbs/min, pound/min, pounds/min]. Pounds per minute.
* **lb/sec**. [lbs/sec, pound/sec, pounds/sec]. Pounds per second.
* **mg**. [milligram, milligrams]. 1/1000 of one gram.
* **mg/sec**. [milligram/sec, milligrams/sec]. Milligrams per second.
* **oz**. [ounce, ounces]. The modern standardized ounce, or International Aviodupois Ounce, is defined as being exactly 28.349523125 grams.
* **oz/hr**. [ounce/hr, ounces/hr]. Ounces per hour.
* **oz/min**. [ounce/min, ounces/min]. Ounces per minute.
* **oz/sec**. [ounce/sec, ounces/sec]. Ounces per second.

{Line}

## Appendix - JSON Catalog Format

As alluded to above, each unit-to-unit calculation within a domain is defined by a series of unidirectional unit definitions that occupy what is being referred to as a *domain*.

In all cases, a single conversion entry represents what it will take to convert the incoming value to the representative base unit, and in most cases, a conversion entry can be completed using one primitive mathematical operation.

The general schema for conversion catalog definitions follows. The main groupings in the pattern are the collection of domains, each domain's collection of conversions, and a conversion's procedural steps in the case that the conversion entry is of a procedural type.

{Line}

* **Remarks**: StringList. Space-extended multiline string array to contain any remarks you would like to present or retain for the loaded dictionary.
* **Domains**: ConversionDomainCollection. Collection of domains constituting this catalog.
  + **DomainName**: String. Name of the calculation domain (ie "Binary Data" or "Pressure", etc.).
  + **Remarks**: StringList. Multiline string.
  + **Conversions**: ConversionDefinitionCollection.
    - **EntryType**: String.
    - **Name**: String.
    - **Aliases**: StringList.
    - **Operation**: String.
    - **RefName**: String.
    - **Value**: Double.
    - **Steps**: ConversionStepCollection.
      * **RefName**: String.
      * **Operation**: String.
      * **Value**: Double.

{Line}

In the above structure, note the following:

* Remarks and Aliases can always be empty or omitted.
* EntryType is one of: "None", "Base", "Conversion", "Procedure".
  + Base indicates that this is the base unit for the domain.
  + Conversion will solve the conversion named in RefName before continuing to the next step. This is mostly useful in the procedural context of a Steps collection.
  + Procedure will sequentially solve the steps found in the item's Steps collection.
* Name is the name of the current unit of measure being solved for.
* Aliases is a list of alternate names by which this unit can be called.
* Operation is one of "None", "Add", "Convert", "Divide", "Multiply", "Subtract", "Reciprocal". If Operation is omitted on either the conversion or the step, the operation will default to "Multiply".
* RefName is ignored except when Operation = "Convert", in which case it will be expected to contain the name of the measurement unit to look up.
* Value is the literal amount to supply in the conversion calculation. Value is not used when Operation = "Convert" or "Reciprocal".
* Steps is ignored except when EntryType = "Procedural".

{Line}

The following empty JSON template exemplifies the general shape of the catalog. Keep in mind that Remarks are optional, the default value for Operation is "Multiply" when omitted, and the Steps collection only appears when EntryType = "Procedure".

{Line}

{CodeBlock,javascript,JsonCatalogFormat.json}

{Line}

**NOTE**: Also see the file {code}Data/DefaultData.json{/code}, in this repository, for examples of how the base conversions have been prepared.

{Line}