UnitConvert (v1.0)

1. Description

Converts numbers between various units.

This plugin lets you specify a number and two units, and it will return a number that is the same number converted from the one unit to the other.
For example, if you say you want to convert 1 meter to centimeters, it will return 100.

The units that can be converted between are in the configuration.
The default configuration includes various units for temperature, distance, weight, volume and more.

Note that there is a limit on the precision of the conversions, as doubles are used internally for calculation and conversions often take several steps of calculations resulting in amplification of errors. Taking several steps has been done to greatly simplify the configuration and code.

2. Detailed description/explanation of how it works

This section can be skipped if you don't care how it works, and just want to know how to use it.

This plugin internally keeps a list of units, referenced by an ID that is also used to select units to convert between. Each unit has a set of fields, specifying its name, its base unit, and how to get to the base unit.

The relationship between a unit and a base unit causes the units to be organized in trees - one tree for each type of unit, that is, units that can be converted between eachother. This means there are separate trees for e.g. distance and volume, while all units that specify volume are in a single tree, with a single root. There is one and only one root unit in a tree. This unit specifies itself as its base unit.

An example will probably make this much clearer. Let's take a look at some of the distance units.
We'll start with the meter as our root unit, and add a few more.

```
ID="m", name="meters", base unit="m", formula="*1";
ID="cm", name="centimeters", base unit="m", formula="/100";
ID="mm", name="millimeters", base unit="m", formula="/1000";
ID="in", name="inches", base unit="mm", formula="*25.4";
```

Now, to get from cm to m, we see by the formula that we need to take the cm value and divide by 100 to get the meter value.
To get from cm to mm, we know that we can just multiply by 10, but the code doesn't really know this - it can only look at the configuration, which doesn't explicitly list a conversion between the two.

It would technically have been possible to specify all the relationships between the units in the configuration file, but this would get very verbose and hard to keep track of, and would make it relatively difficult to add new units. Because of this, this tree structure has been chosen, which is much easier to work with in the configuration, and which only needs a little more code.

Since a unit are structured in a tree, not a map, converting from one unit to another usually requires more than one conversion. The algorithm used starts by finding the closest common ancestor (which in some cases can be one of the given units). Then it starts with the unit being converted from, and follows the path to that common ancestor, converting the given value to each unit it passes on the way, so it ends up at the common ancestor with the unit of that ancestor. It then follows the path from there to the desired unit, still converting as it goes.

To continue the earlier example, going from cm to mm, it will first convert the cm value to meters, dividing by 100, and then go from meters to mm, which is done by using the inverse of the mm-to-meters formula - multiplying by 1000.

In this case, the numbers are straightforward, and the chain short (just two conversions) - but in some cases you need long chains and the formulas use numbers that aren't. This is when inaccuracies appear. Note that even relatively large inaccuracies usually give values that are only wrong in the last few digits(of long numbers - shorter ones are usually accurate, or at least very close), and it is assumed that the users don't need highly accurate data. This is why the errors are accepted.

The formula for going from one unit to the next, is in the configuration split into three attributes, which is believed to be enough to convert between any two relevant related units. The three attributes are preAdd, mult and postAdd. Each specify one number. (It could here be noted that as of yet, none of the units in the default configuration use postAdd.)
The formula for going to the base unit then becomes newValue=((oldValue+preAdd)*mult)+postAdd where oldValue is the value in this unit, and newValue the value in the base unit.
Inversely, to go the other way, the formula is oldValue=((newValue-postAdd)/mult)-preAdd

3. Configuration

The configuration file for this plugin is **system/UnitConvert/UnitConvertConfig.xml**

3.1. commandConvert

The command prefix for converting a number between units.
In other words, what to put before X-Y to make it react.
You can set the authLevel to control who is allowed to access the command. The output is used to define where the output from the command will go.

- **authLevel** - ANY, ADMIN, MASTER, TRUSTED, NONE
- **output** - CHANNEL, PM, NOTICE

```
        <commandConvert authLevel="none" output="channel">!</commandConvert>
```

3.2. commandListConvUnits

The command for listing the available units that can be converted between.
You can set the authLevel to control who is allowed to access the command. The output is used to define where the output from the command will go.

- **authLevel** - ANY, ADMIN, MASTER, TRUSTED, NONE
- **output** - CHANNEL, PM, NOTICE

```
        <commandListConvUnits authLevel="none" output="channel">!ConvUnits</commandListConvUnits>
```

3.3. commandShowConvPath

The command for displaying the conversion path to go from one unit to the other.
You can set the authLevel to control who is allowed to access the command. The output is used to define where the output from the command will go.

- **authLevel** - ANY, ADMIN, MASTER, TRUSTED, NONE
- **output** - CHANNEL, PM, NOTICE

```
        <commandShowConvPath authLevel="none" output="channel">!ConvPath</commandShowConvPath>
```

3.4. Unit

This group contains a list of Unit elements, as defined below, that define the available units.

This element defines a unit and its properties.

- **ID** - The ID of this unit, usually its symbol/abbreviation, is used for saying which unit to convert to/from and to link the unit trees.
- **name** - The name of this unit.
- **baseUnit** - The ID of the base unit for this unit.
- **preAdd** - Part of how to get to the base unit from this unit; the value to add before multiplication.
- **mult** - Part of how to get to the base unit from this unit; the multiplication factor.
- **postAdd** - Part of how to get to the base unit from this unit; the value to add after multiplication.

```
  <Units>
    <Unit ID="C" name="degrees Celsius" baseUnit="C" preAdd="0.0" mult="1.0" postAdd="0.0" />
    <Unit ID="K" name="degrees Kelvin" baseUnit="C" preAdd="-273.15" mult="1.0" postAdd="0.0" />
    <Unit ID="F" name="degrees Fahrenheit" baseUnit="C" preAdd="-32.0" mult="0.5555555555555556"
postAdd="0.0" />
  </Units>
```

4. Commands

4.1. !X-Y

Command: !X-Y
Description: Where X and Y are unit IDs, convert the following number from unit X to unit Y.
Auth Level: none
Where to give command: channel, pm
Outputs to: same as given
Example(s):

- !in-cm 4
- !F-C 70

4.2. !ConvUnits

Command: !ConvUnits
Description: This will return the list of available units, or if a unit ID is given, show some information on that unit.
Auth Level: none
Where to give command: channel, pm
Outputs to: same as given
Example(s):

- !ConvUnits
- !ConvUnits yd

4.3. !ConvPath

Command: !ConvPath
Description: This will show the path the code takes to get from one unit to the other.

Auth Level: none
Where to give command: channel, pm
Outputs to: same as given
Example(s):

- !ConvPath yd cm

5. Change History

v1.0	First Version by EdorFaus