Rainmeter 3.0 Manual

Contents

Reference

2013

IZEN.ORG

IZEN

Izendark07@gmail.com

[RAINMETER 3.0 MANUAL]

Manual [home]

Rainmeter displays customizable skins, like memory and battery charge, RSS feeds and weather forecasts, right on your desktop. Many skins are even functional: they can record your notes and to-do lists, launch your favorite applications, and control your media player - all in a clean, unobtrusive interface that you can rearrange and customize to your liking. Rainmeter is at once an application and a toolkit. You are limited only by your imagination and creativity.

Getting started with Rainmeter

After downloading and installing Rainmeter, the very next stop should be Getting Started.

Getting Started is a **new user** introduction to Rainmeter, covering both how to use the application, and starting guides and tutorials for creating and editing skins.

Using the rest of this documentation, and the experience of using and learning Rainmeter, will be much easier and more enjoyable with the overview Getting Started provides.

Using the Rainmeter application

The User Interface section of the manual describes how to use the application to manage Rainmeter on the desktop. Control things with the Manage interface and context menus, and monitor important information usingAbout.

It won't be long before you want to extend your library of skins beyond the default **illustro** suite included with Rainmeter, with others downloaded from the internet. The Installing Skins section explains how to install and load the thousands of skins available for Rainmeter.

Creating and editing skins

Rainmeter can be used as a straightforward skin manager, loading and using skins created by others. However, the real power and fun of Rainmeter can only be unlocked when the transition is made from "user" to "author". TheReference section of the manual contains everything needed to craft that perfect skin that does exactly what you require, and looks just right on your desktop.

Settings contains detailed information about the **Rainmeter.ini** file, which controls many important aspects of Rainmeter on the desktop.

The Skins section describes what a skin's .ini file contains and does. It also is a reference for several powerful overall skin features.

The next sections, Meters, Measures and Plugins, are the core information for creating skins. Measures and plugins gather information from the system or the internet, and meters display things, with many different meter types and practically unlimited control over style.

The following sections provide a reference for other powerful skin options and features. Bring skins to life withBangs and Mouse Actions, and extend Rainmeter's functionality using the Lua Scripting language.

One of the most exciting things about Rainmeter is the free and open sharing of skins in the community. SeePublishing Skins for everything needed to package and distribute suites and skins.

More information

Be sure to check the Tips & Tricks section for a wealth of guides and suggestions from the Rainmeter community.

The Snippets section contains a collection of useful "snippets" of Lua script code, from the Rainmeter Team and users.

Remember, there is an entire community of people willing and able to help when things get stuck. Be sure to use the Rainmeter Forum or jump on the web-based IRC channel when help is needed.

Note: Most option entries and header text in the manual can be selected with **CTRL-Click** to add an **#anchor** to the URL. This makes it easy to link to a specific part of a page. Try it below.

```
SampleOption
```

Getting Started [home]

If this is your first experience with Rainmeter, then you're in the right place! *Getting Started* is a guide designed to walk you through the basics of setting up Rainmeter, using, customizing, and ultimately creating your own skins.

Before you begin, here are some frequently-asked questions about what Rainmeter is and how it works.

What is Rainmeter?

Rainmeter is a free, open-source application for Windows PCs. It is a platform that enables **skins** to run on the desktop.

What is a "skin"?

Some Rainmeter skins. Each skin is a separate window, and can be moved around on the desktop by clicking and dragging. Rainmeter can run any number of skins at one time, even from different sources.

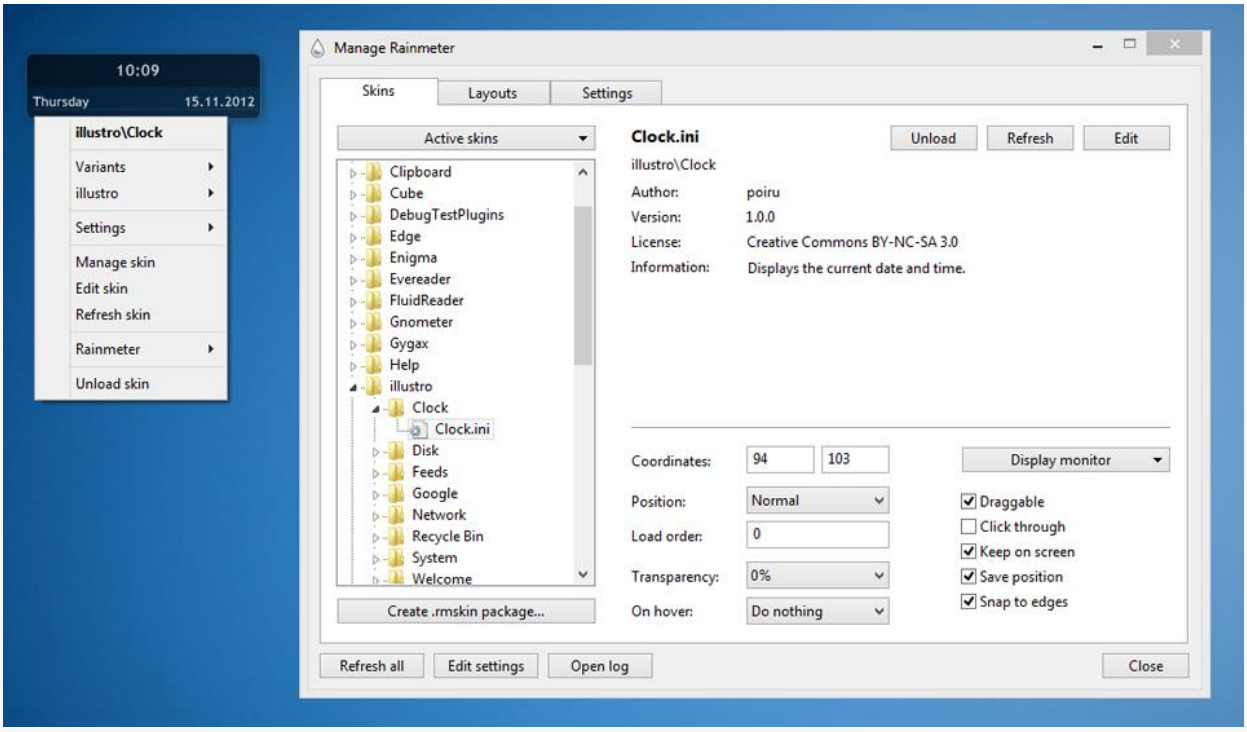
A skin can be many things. Some skins are very simple, single-purpose tools, like Windows desktop gadgets, or "widgets" on an Android device. Others are more complex, like miniature applications themselves. Some skins even come bundled in large "suites" and include their own tools for customizing their form and appearance, within or alongside Rainmeter's basic user interface. Every skin works differently, depending on the choices of that skin's individual author.

However, all skins are made from the same building blocks: measures, which gather information from your computer, a website, a text file, or some other source; and meters, which create visual elements in the skin's window, such as frames, borders, backgrounds, images, text, charts, or buttons.

Skins can interact with other skins and applications using special commands, called bangs, and they can be customized by changing short lines of text, called variables. All of these things are made possible by Rainmeter's unique code language, which allows a skin to access functions and resources built into the Rainmeter application. Every skin's code is completely open, and can be tweaked, modified or even completely rewritten using any text editing software.

How much technical skill do I need to use Rainmeter?

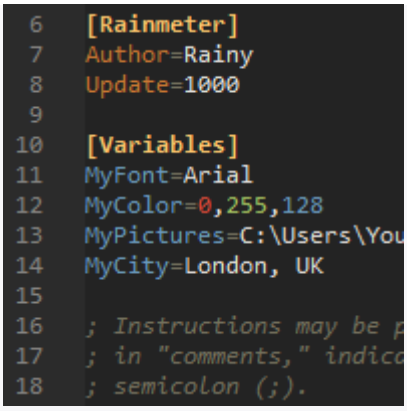
If you only want to **download** skins from the Internet and **use** them as-is, then the answer is "none." Rainmeter provides a basic user interface for managing your library of skins, saving and restoring layouts, and changing basic settings such as a skin's location, transparency, and "always on top" behavior.



Most Rainmeter features can be reached through the basic Manage window or context menus.

Some skin authors create their own controls for users to **customize** their skins. These controls may be included as a separate utility, or they may be created entirely within Rainmeter as another skin. If this is the case, then you will not need to know any code to customize these skins.

In other cases, you may need to change some variables in the skin code. This may be scary if you are not a programmer, but usually, these "variables" are clearly marked labeled, located near the beginning of the file so that you don't have to do any searching, and are accompanied by helpful instructions and comments.



An example of some Rainmeter code.

If you want to **create** skins, or **modify** someone else's skin beyond the customization options that the author has provided, then you will get some hands-on experience with Rainmeter's code language. You do not need to be professional programmer to become a proficient writer of Rainmeter skins—although those skills will certainly help you, and a full-powered scripting language is available for advanced users. But all of a skin's basic properties are written using a simple configuration language that is suitable for novice programmers. The difficulty level is similar to that of HTML or JavaScript.

What *isn't* Rainmeter?

Rainmeter is just one of many different tools that you can use to customize your Windows PC. It includes a powerful and flexible set of features, and we are continually surprised by the creative ways that those features are used. However, it is important to understand what Rainmeter does *not* do:

- Rainmeter does not change your Windows **visual style**. It cannot change the appearance of your taskbar, Start button, desktop icons, file explorer, or other built-in Windows components.
- Rainmeter is also not a **window manager**. It does not keep track of your open windows; it cannot maximize or minimize other application windows; and it does not enable "workspaces" or manage multi-monitor setups.
- Rainmeter does not **replace other applications** that it interacts with. For example, an "iTunes" skin may let you pause, play or skip to the next track in your iTunes media player. But iTunes must still be running in the background for the skin to work.

In short, you cannot usually download and apply someone else's amazing desktop transformation in one click. Most customizers are courteous enough to provide links to the myriad programs, plugins, icons, wallpapers and other materials that they have used.

Setting Up [home]

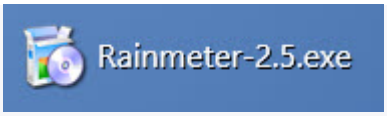
System Requirements

Rainmeter requires **Windows XP** or above. Some specific features require Windows Vista or above. The same installer may be used on either **32-bit or 64-bit** systems, and all skins and features are compatible with both architectures.

Rainmeter is not available for non-Windows systems, including Mac OS and Linux, and there are no plans to support these operating systems.

Download

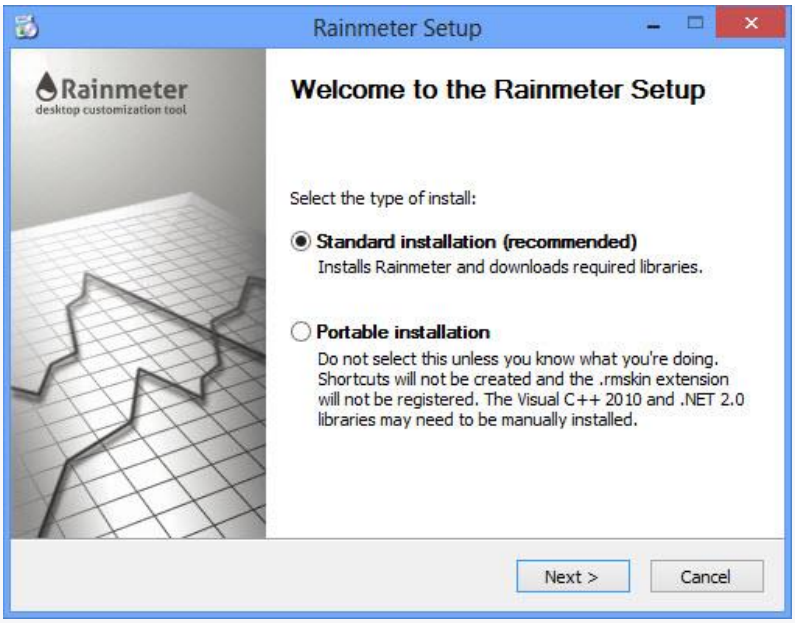
Start by downloading the Rainmeter installer from the Rainmeter.net.



The Rainmeter installer.

There are two versions of Rainmeter available: the **final** release of the current version, and the **beta** release of the next version. Betas are released weekly, and final versions are released every few months. Despite the "beta" label, every release of Rainmeter is stable and backwards-compatible with previous versions. The only difference is that beta features may (rarely) be changed or removed before the next version becomes final.

We recommend the beta version for most users who want to get the latest features and bug fixes. However, if you do not want to risk using a beta feature that may not work the same in a future release, use the final version instead.



Installing Rainmeter is mostly automatic.

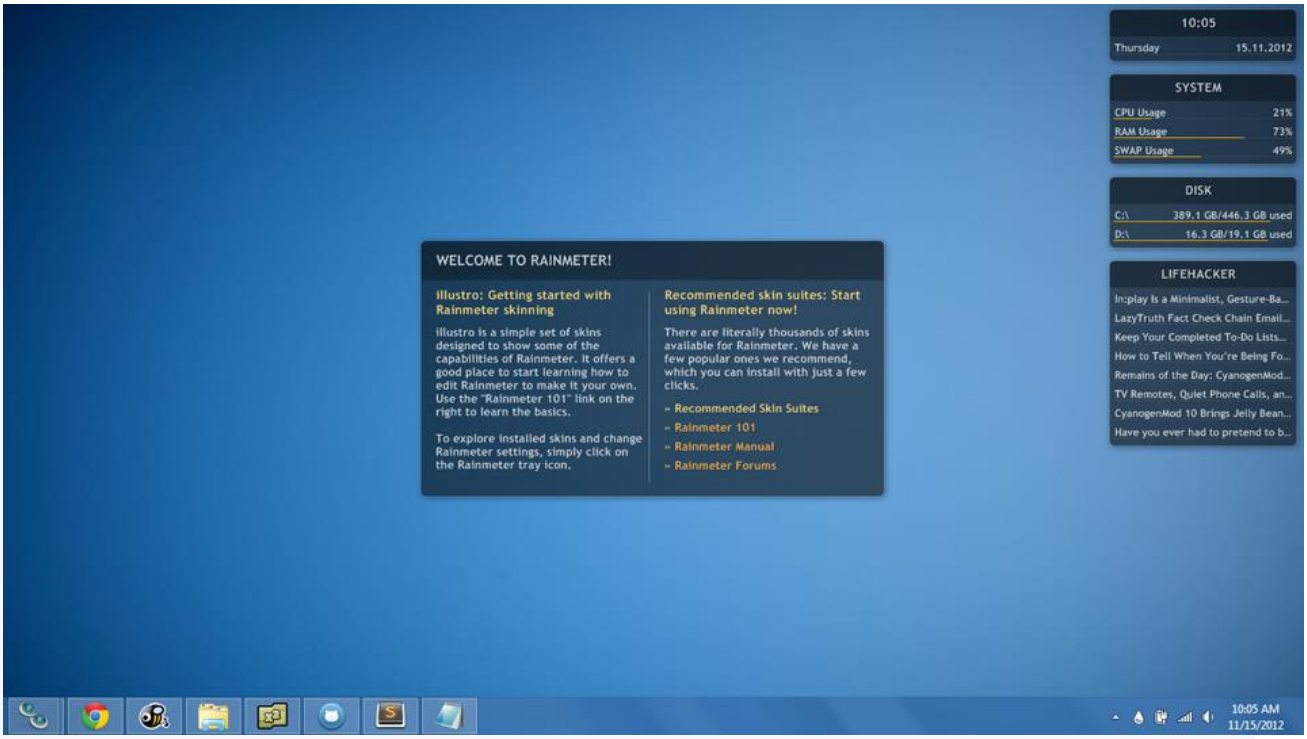
Install

To install Rainmeter, run the installer program that you downloaded, and follow the instructions. (See Installing Rainmeter if you need more detailed instructions, including steps for installing Rainmeter as a portable application.)

Rainmeter will run automatically after it has been installed.

Using Rainmeter [home]

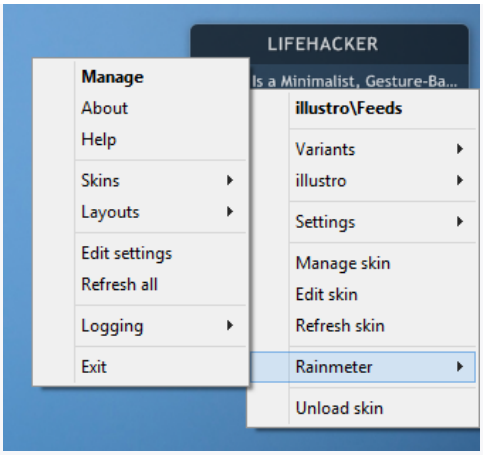
The first time you run Rainmeter, your desktop will look something like this:



***Illustro*, the standard "suite" that comes with Rainmeter.**

Each small window on the right side of your screen ("System," "Disk," and so on) is a skin. Rainmeter remembers each skin's location and settings independently. To move a skin, just click and drag it to a new location.

Context Menu



A typical skin's context menu.

The easiest way to interact with Rainmeter skins is through the context menu. A skin may have any number of tabs, buttons, menus, or other bells and whistles in its design—but no matter what, you can still access the context menu by right-clicking on the skin.

Some skins may assign some other action to right-clicking on the skin. In this case, you can still override this action and open the context menu by holding down the `Ctrl` *key when you right-click.*

All skins have the same basic context menu items that you can see in the screenshot on the right. Some skins may have custom items to their context menus, but these these will not replace the basic items. Instead, both types of items will appear alongside each other.

Loading and Unloading

You can use the context menu to load skins from your library. Right-click on any of the *illustro* skins, and select `illustro` → `Google` → `Google.ini`. The "Google" skin will appear in the top-left corner of your desktop. You can now drag it into place alongside your other skins.

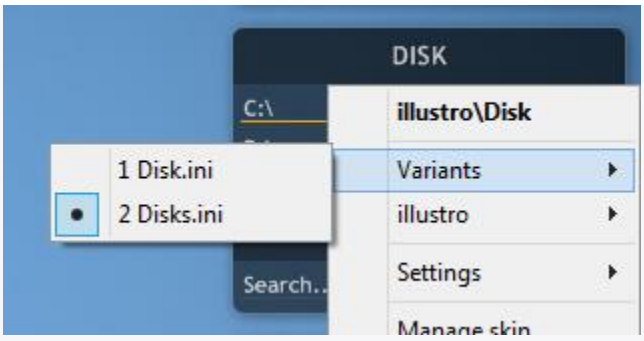
The Google skin has been loaded.

You can also unload a skin with the context menu. Right-click the new Google skin, and select `Unload skin`. You will see the skin fade away and disappear.

When a skin is unloaded, its location and settings are still saved. To see how this works, load the Google skin again. Notice that, instead of appearing in the top-left, it appears in the same location where you dragged it before.

Variants

When you loaded the Google skin, there was only one option under the "Google" menu, `Google.ini`. This is how most skins work: each skin is saved as a "`SkinName.ini`" file in a separate folder. What happens when there is more than one skin in the same folder? The answer is that these skins become variants of each other.



"Disk" has two variants.

To see an example of a skin with variants, right-click the "Disk" skin and select `Variants` in the context menu. You can see that the `2 Disks.ini` variant is already active. Click `1 Disk.ini` to

switch to that variant. Notice how the new variant replaces the old variant on the desktop. Variants share the same location and settings, and only one of a skin's variants may be loaded at one time.

Settings

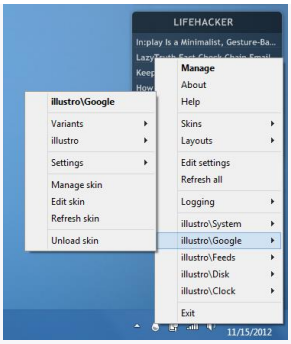
Rainmeter keeps track of a number of basic skin settings. These settings are created automatically for each skin. They include things like position (whether a skin stays on top of all windows or is pinned to the desktop), snap to edges (whether skins automatically align with other skins when they are dragged close together), and of course, the skin's coordinates (location on the desktop) and which variant is loaded.

"Welcome" is now transparent.

One of the most popular skin settings is transparency. Try changing the transparency of the "Welcome to Rainmeter!" skin in the center of your desktop. Right-click the skin to open the context menu, then select `Settings` → `Transparency` → `50%`. Now you can partially see through the skin and onto the desktop below.

Rainmeter does not manage any other "settings" beyond the basic ones found here. Extra customization options—like fonts, colors, images and passwords—are all stored and managed separately by the skins themselves, as variables. This means that saving and restoring your layout settings in Rainmeter will *not* affect your customizations for specific skins. (It also means that backing up your Rainmeter settings will not protect your customizations, so make sure to back up your skin files, as well.)

Tray Icon



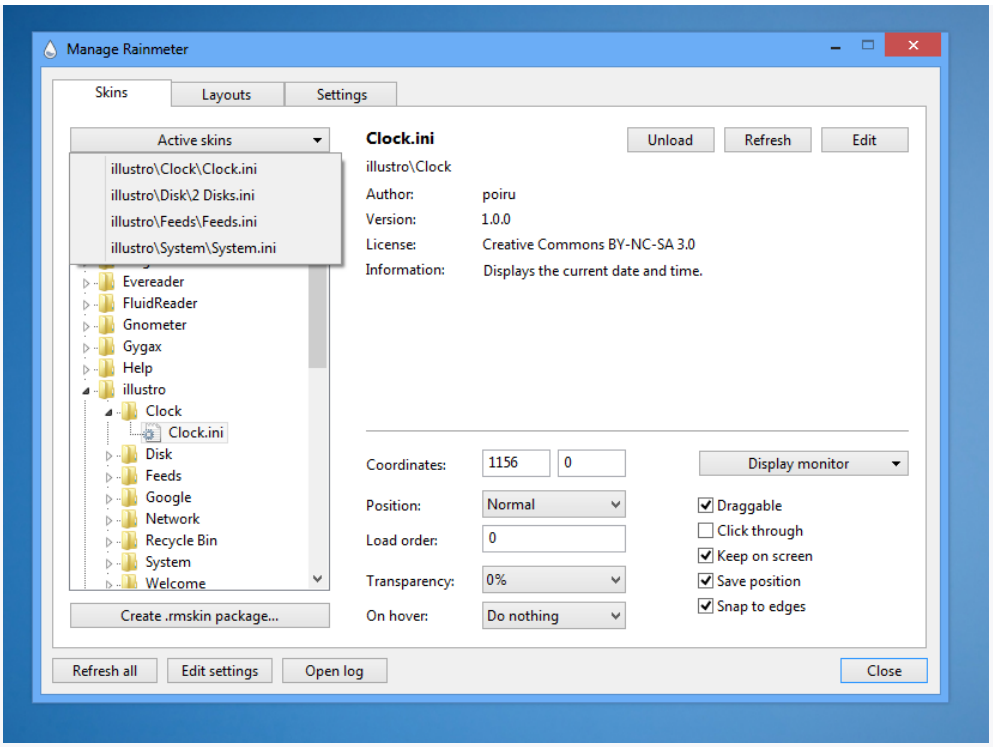
There are two ways to open the context menu.

Rainmeter has a notification area icon. You can reach the context menu for each of your loaded skins by right-clicking on the icon. This is a handy way to access a skin when you can't right-

click it for some reason (usually when the skin is hidden). It's also a good place to see a complete list of all the skins that you have loaded.

Manage

Aside from the context menus, the most important part of Rainmeter's basic user interface is the Manage window. Open Manage by selecting `Manage` from the context menu, or by left-clicking once on the tray icon.



The "Manage" window shows your whole library.

Skins

The Skins tab allows you to load, unload, or change settings for your entire library of skins. You can go to a specific skin's settings in the Manage window by choosing `Manage skin` from the skin's context menu, or choosing from the `Active skins` drop-down menu in Manage. You can also right-click to open any skin's folder in Windows Explorer, where you can see other resource files for that skin, such as images or scripts, or delete a skin's file, from your computer.

The Manage window does many of the same tasks as the context menu, but when you need to load or edit options for a number of skins at the same time, it's much easier to use the window.

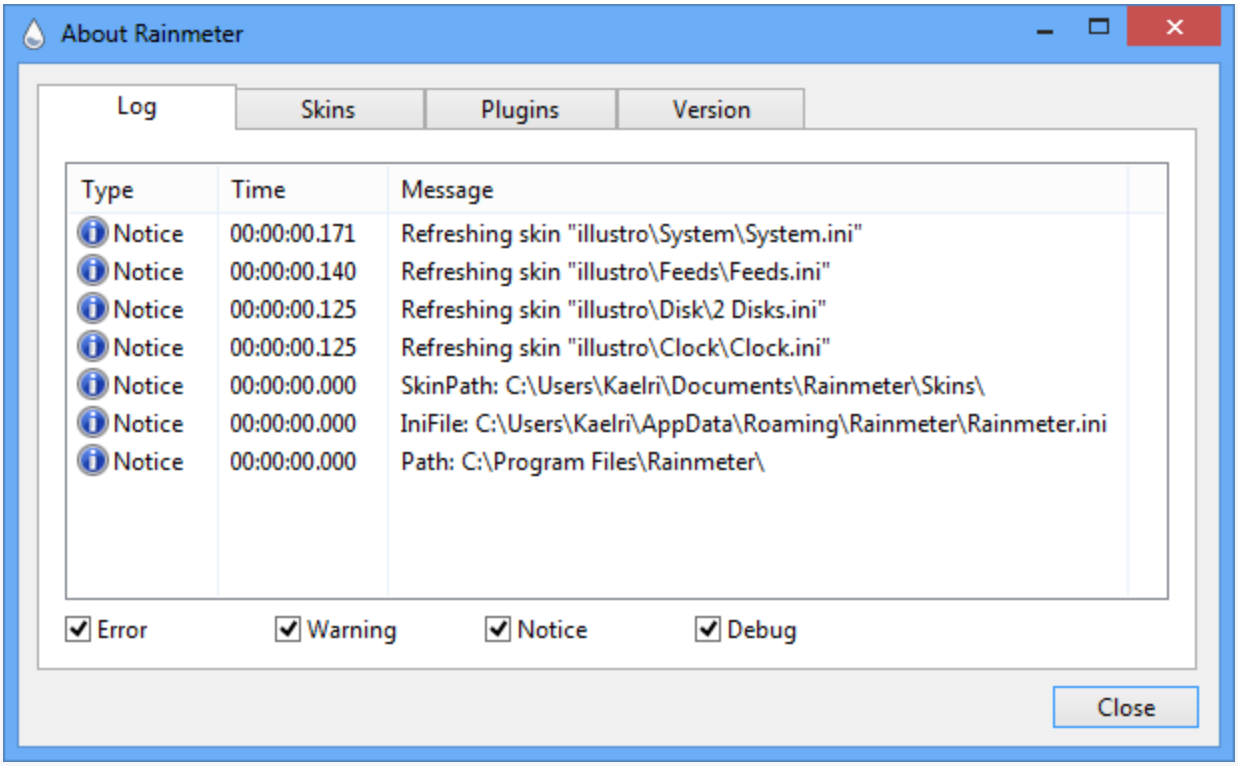
Layouts

Manage is also where you can load and save layouts. A layout is a permanent copy of your skin settings, including both loaded and unloaded skins. (Again, these settings do not account for changes to your skin files, which include any extra customization options. Loading a layout will also not recover a skin that has been deleted from your library.) When you save a layout, you

have the option of excluding settings for unloaded skins, which is a good way to "clean up" skins that you may no longer have or use. You can also associate your current wallpaper with your layout.

About

The last part of Rainmeter's interface is the About window. Open About by selecting `About` from the context menu, or by clicking `Open log` in Manage.



The "About" window shows you what Rainmeter is doing.

Log

The Log tab is where Rainmeter keeps a running record of what it is doing. This is also where Rainmeter reports errors related to a skin or the program itself. In addition, the Skins tab will show you the current values of all measures and variables in the skin. (You'll learn more about those later.) This makes the About window an invaluable assistant whenever you're writing, editing or troubleshooting a skin.

Version

The other tabs show information about the version of Rainmeter you are using, as well as the versions of pluginsthat you have installed. If you are ever having a technical issue and need to ask for support from the Rainmeter community, the About window is where you can find a wealth of information about your Rainmeter setup that may be helpful in solving your problem.

Exiting Rainmeter

Rainmeter saves all settings automatically. If you exit Rainmeter (by clicking `Exit` in the context menu), or even if the program crashes unexpectedly, all skins' settings are still saved, and your current layout will be loaded automatically the next time Rainmeter is launched.

Customizing [home]

Now that you are familiar with Rainmeter's basic user interface, you're ready to start customizing Rainmeter to your liking.

Finding Skins

There is no official, central repository of Rainmeter skins. Rainmeter is an open platform, and skins can be found all over the Internet, from large screenshot galleries to small personal blogs and websites. That said, there are a few major sites where the Rainmeter community tends to gather:

- **Rainmeter.net**
  The Rainmeter homepage highlights Featured Suites from veteran skin authors in the community.
- **Rainmeter Forum**
  Other members of the community like to share their creations on the official boards.
- **DeviantArt**
  Probably the biggest and best collection of skins, deviantArt not only has a special Rainmeter category, but also a dedicated Rainmeter Group, which is run by members of the community and features a curated stream of skins, screenshots, tutorials, interviews and more. All skins on deviantArt are checked for malware before they are accepted by the group.
- **Customize.org**
  One of the first gathering places for Rainmeter enthusiasts, Customize.org hosts one of the largest and oldestRainmeter skin collections.
- **Lifehacker**
  Not only is the Lifehacker community full of Rainmeter users, but Lifehacker's editors frequently post featured desktops and how-to guides for popular skins under the Rainmeter tag. Be sure to browse the Lifehacker Desktop Show & Tell pool on Flickr, as well.

Malware

A verified ".rmskin" package. Look for the green Rainmeter icon.

Rainmeter is built on an open software ecosystem. And like other open software, we occasionally have to deal with malware in our midst. Other than Rainmeter.net, the websites listed above allow anyone to upload and publish their own skins, and most sites have no policy of verifying either the identity of the uploader or the integrity of the files. **We rely on members of the community to be watchful for malware and report it when found.**

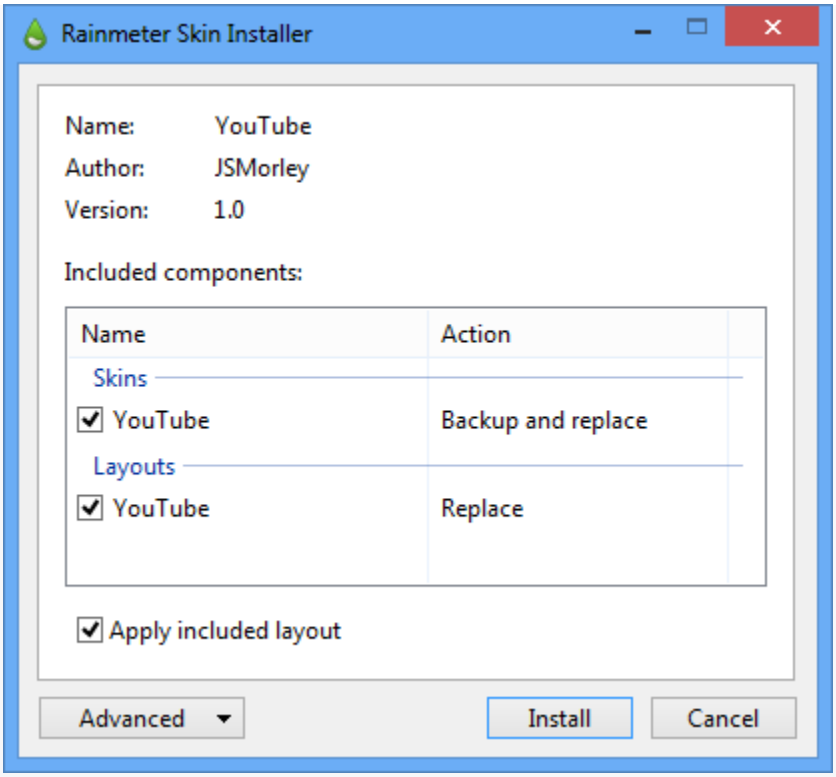
We recommend that you follow some simple, common-sense tips to make sure that your computer stays malware-free:

- Whenever possible, only download skins in the Rainmeter Skin Installer *(.rmskin)* format. Recent versions of Rainmeter require new skin packages to be created by Rainmeter's official Skin Packager, which helps reduce the risk of tampering. Many older skins may still

be published as *.zip* files that must be installed manually—these files are at greater risk of containing malware, and you should look for a newer version of the skin, if possible. You should never download a *.exe* file that claims to be a Rainmeter skin installer.

- Make sure the skin publisher is trustworthy. An account that is new, or has very few downloads, page views, or profile detals may be suspicious. Users that disable or delete comments on their submissions are also suspicious. In contrast, a long-standing author who has many comments and downloads is more likely to be safe.
- If comments are enabled on the submission, check them to see if other users have reported the skin as malware. Likewise, if you discover that a skin is dangerous, please take a moment to warn your fellow Rainmeter users, by posting a message in the comments and reporting the submission to the side administrators. Sites like deviantArt and Flickr include an easy "Report Spam" button on every page.
- If you are suspicious of a skin for any reason, upload the file to VirusTotal, which will scan the file with over 40 anti-malware services and report the results to you. (Note that false positives are common; only report skins that are flagged as dangerous by a significant number of tests.)

Adding Skins



Installing a typical Rainmeter skin.

Installing a skin in the Rainmeter Skin Installer *(.rmskin)* format is easy. Just double-click the file to open the package with Rainmeter's built-in Skin Installer, and follow the instructions like you did when you installed Rainmeter.

(Skins that are not in the .rmskin format, but instead come in an archive file like *.zip*, *.rar* or *.7z*, require a few more steps. See Installing Skins for more detailed instructions, including steps for installing skins on a portable version of Rainmeter.)

For an example of a Rainmeter skin package, try downloading and installing one of the Featured Suites from the Rainmeter homepage.

Editing Skins

You have already learned how to change basic skin settings in Rainmeter. Now, it's time to learn how to make edits to a skin itself.

Remember: every skin is different. Some skins provide more customization options, tools and instructions than others, and the processes for customizing two skins may be very different. In all cases, you should look to whatever documentation the author has provided, whether in the form of a "readme" file, an online support URL, or `;comments` *embedded in the skin code.*

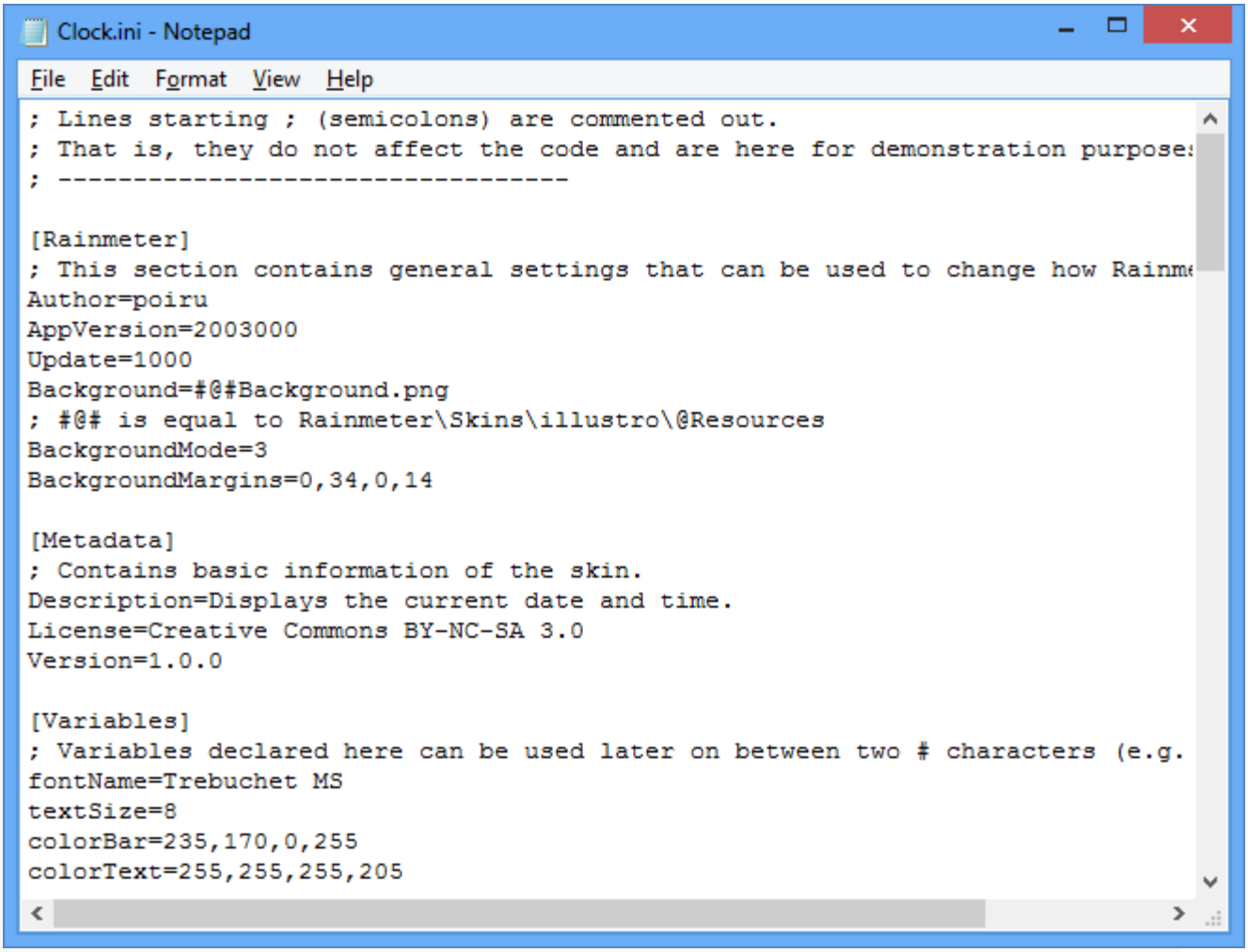
Controls

Built-in control tools for three featured suites.

If you're lucky, your skin will provide some kind of built-in graphical interface for changing its own code. In this case, all you need to do is follow the instructions provided, clicking buttons and typing or pasting new text values as needed. The skin may have control buttons embedded in the skin itself, or it may summon a separate skin or addon where you can change all customization options in one place.

This kind of sophisticated control system is more the exception than the rule, and is usually found only on large suites. More often, you will need to edit the skin code directly in order to change or add your own values.

Variables



What a skin looks like on the inside.

Most of the time, a skin's customization options—that is, options that the author intends and expects you, the user, to change around—are saved as variables. To see an example, right-click the illustro "Clock" skin on your desktop, and select `Edit skin`. The file "Clock.ini" will open in Notepad (or your default text editor).

This is what Rainmeter's configuration code looks like. (Click the image on the right to zoom in.)

Although it may look complex, everything in this file is actually one of three basic elements:

- A **section**. Section names are contained in brackets (`[]`) and are used to "declare" some property of the skin.
- A **key**. Keys are found at the beginning of each line in a section, followed by an equal sign (`=`).

- A **value**. The value is whatever meaning is assigned to a key—everything after the first equal sign (`=`) is the value. (Values must be constrained on a single line.) In Rainmeter, the pairing of a key and a value is called anoption.

```
[Section1]
Key1=Value
Key2=Value
Key3=Value

[Section2]
Key1=Value
Key2=Value
Key3=Value
```

Select all

When you're editing variables in a skin, the *values* are what you are going to change.

Scroll down in your text editor to the section called `[Variables]`. Here's what it looks like. (Code examples in this manual are highlighted to help you easily identify sections, keys and values.)

```
[Variables]
; Variables declared here can be used later on between two # characters (e.g. #MyVariable#).
fontName=Trebuchet MS
textSize=8
colorBar=235,170,0,255
colorText=255,255,255,205
```

Select all

Try changing one of these variable values. Go to the line that begins with the `fontName` key, and change`Trebuchet MS` to another font - say, Times New Roman:

`fontName=Times New Roman`

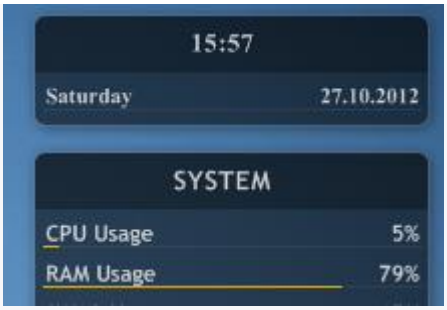
Then, save the file.

Refresh

The refresh action is important in Rainmeter. The effect of refreshing a skin is the same as if you had unloaded and then re-loaded the skin. When this happens, all dynamic variables and other options are reset according to their original values in the skin file, and all measures restart from

the beginning. Refreshing a skin is also necessary to apply any changes that have been made to a skin's files.

You can refresh any skin by selecting `Refresh skin` in the context menu, or browsing to the skin in Manage and clicking the `Refresh` button in the top-right.



It may be ugly, but it's different.

When skin files have been added, renamed or removed, you will need to refresh the entire Rainmeter application in the same way. You can do this by selecting `Refresh all` in the context menu or the bottom-left of the Manage window.

Try refreshing the Clock skin that you edited before. You will see the skin's font face instantly change.

You have now edited your first skin! This is the same basic process that you will use to customize, modify, and eventually create skins of your own.

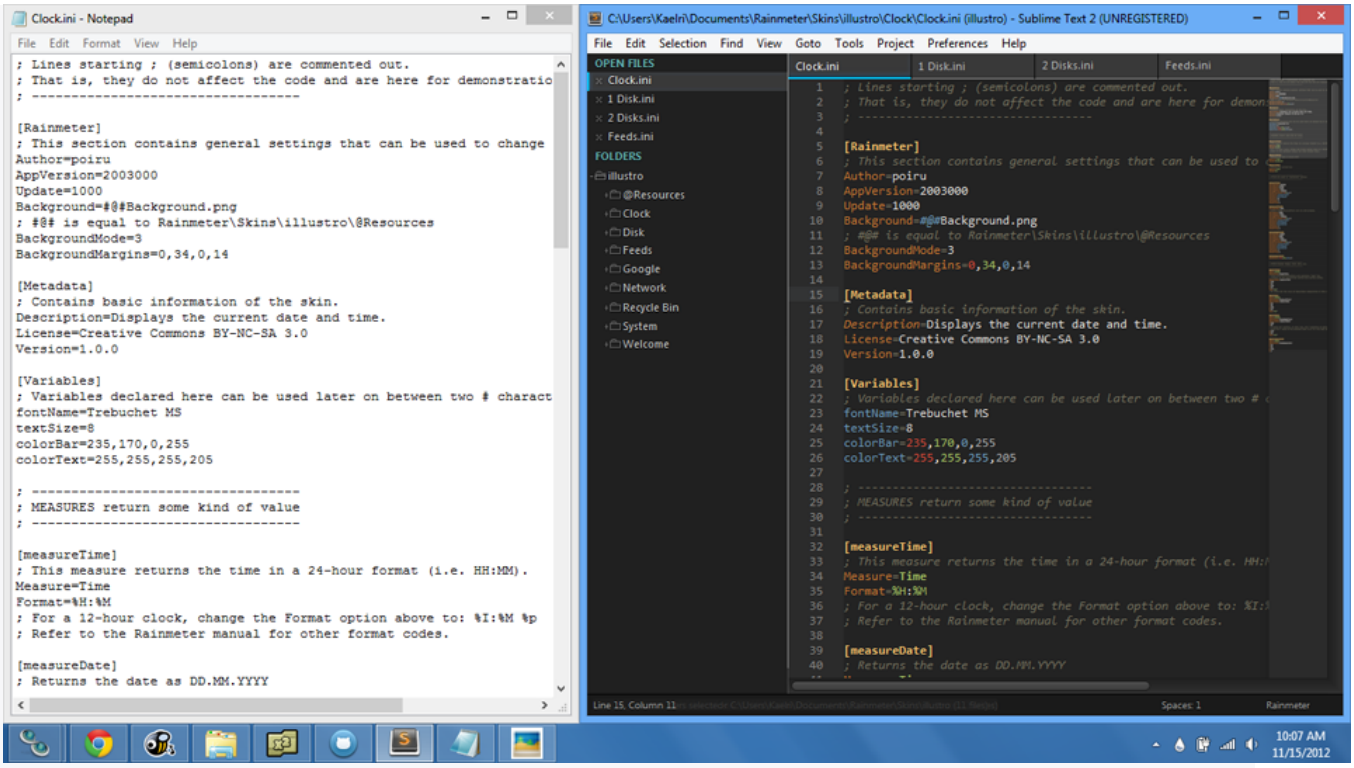
Going Further

If you don't find the option that you want to change in the `[Variables]` section, and the author has provided no other instructions, then your objective goes beyond merely "customizing" a skin: you will need to actually *modify*the skin away from the original author's design.

This is not only possible—we actually encourage it! Breaking down the pieces of someone else's creation and building something new with them is one of the best ways to learn Rainmeter. Modifying skins and creating new skins involve basically the same skills and require an understanding of the same core concepts. It is the tweakers, the experimenters, who get the most out of the tools that Rainmeter has to offer.

Creating Skins [home]

What You Need



An enhanced text editor with code highlighting makes a big difference.

Here's what you need to create Rainmeter skins:

1. Rainmeter.
2. A text editor.

And that's all. No extra software or materials of any kind are required.

Text Editor

That being said: while you *can* edit skins using Windows' built-in text editor, Notepad, we strongly recommend downloading an enhanced text editor, such as Notepad++ or Sublime Text. These applications come with powerful features like tabs, auto-completion, embedded file browsers and more. You can even download extensions that add Rainmeter-specific code highlighting, which makes it much faster and easier to read a skin's code and spot errors.

For more information, see Notepad Alternatives.

Image Editor

Depending on the kind of skin you want to make, you may also want to find a good piece of image editing software. Rainmeter can create text by itself, as well as simple shapes, like rectangles and circles, with solid colors or color gradients. But anything more complex will require a separately-created image file.

Adobe Photoshop is the usual gold standard for image editing, but there are other, less expensive alternatives worth considering, such as Paint.NET, GIMP, or Inkscape.

Configs

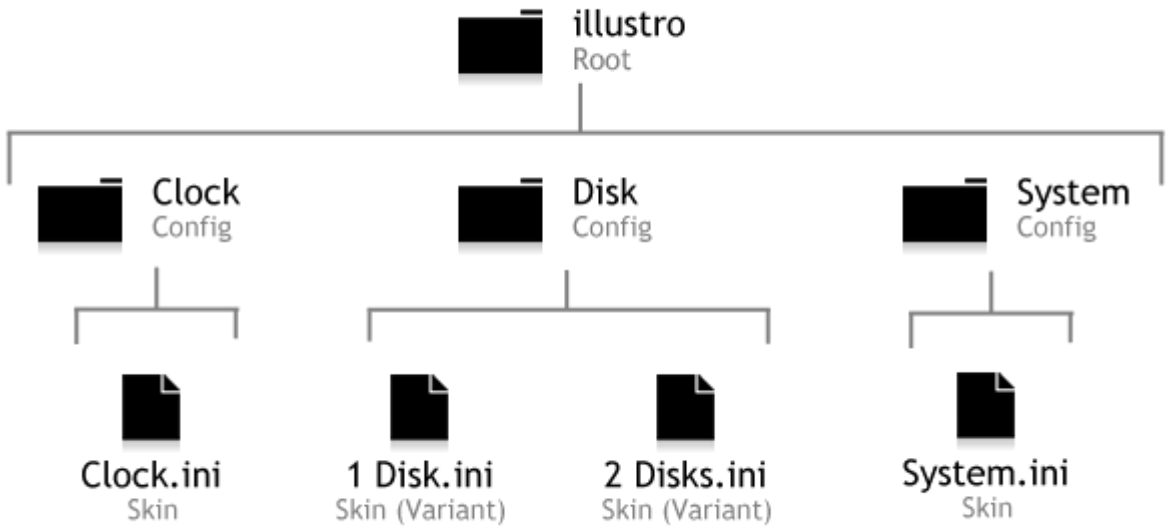
A skin can grow to become an enormously complex project by the time it's done. It may accumulate any number of images, icons, fonts, plugins, addons, scripts, and even included code that is strewn across multiple files and shared by other skins.

But at the core of every skin is a single *.ini* file. Named `SkinName.ini`—where "SkinName" is the name of the skin—this is a text file that contains the fundamental code that Rainmeter uses to create a working skin.

Because a skin may have any number of variants, skins are typically identified not by their file name, but by the folder where they are located. This is known as the skin's config name. To quickly find out a skin's config name, just check the context menu—the first item is the config name. For example, the illustro "Clock" skin's config name is `illustro\Clock`.

Each skin also has a root config folder. This refers to the one folder that contains *all* of the skins belonging to a "suite," such as illustro. When the skins in a suite are organized together in this way, they can be exported to apackage, and then installed on another system, as a single collection. They can also share fonts, images and other resources in a way that separate skins cannot. For a simple skin that is not part of such a "suite," the config and root config are the same.

Here's a quick reference chart to help you remember the relationship between skins, configs, variants and roots, using illustro as an example:

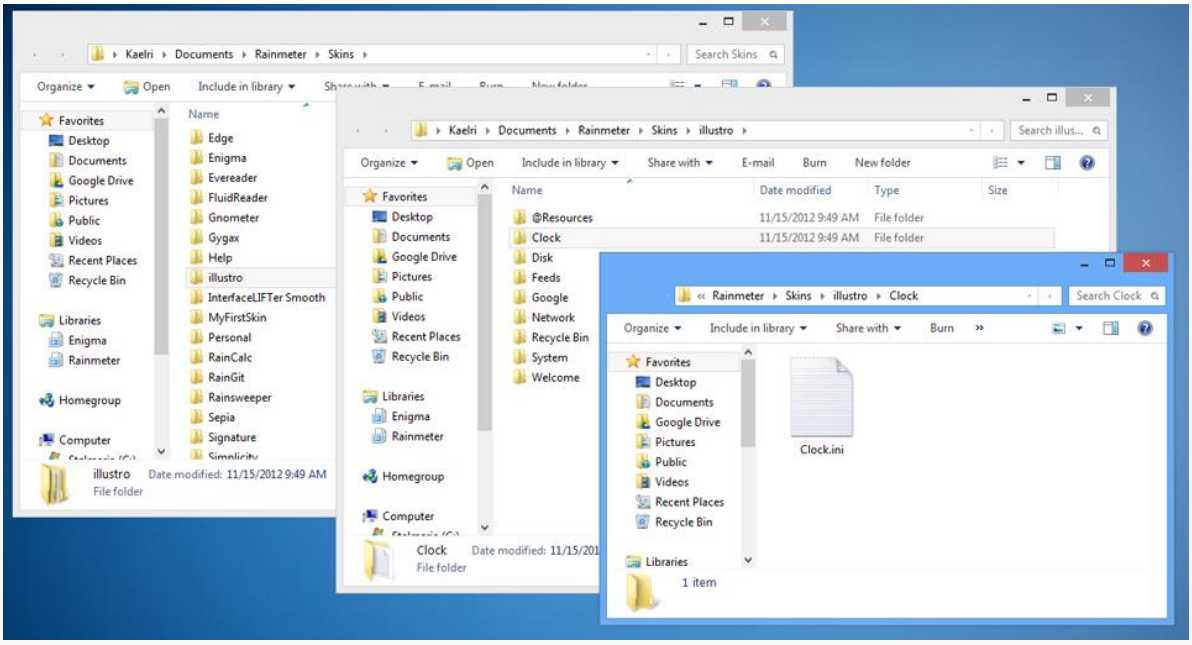


Root config folders are all organized in Rainmeter's main Skins folder:

`C:\Users\YourName\Documents\Rainmeter\Skins`[1]

Accordingly, you can identify any skin by its file path, according to a simple rule:

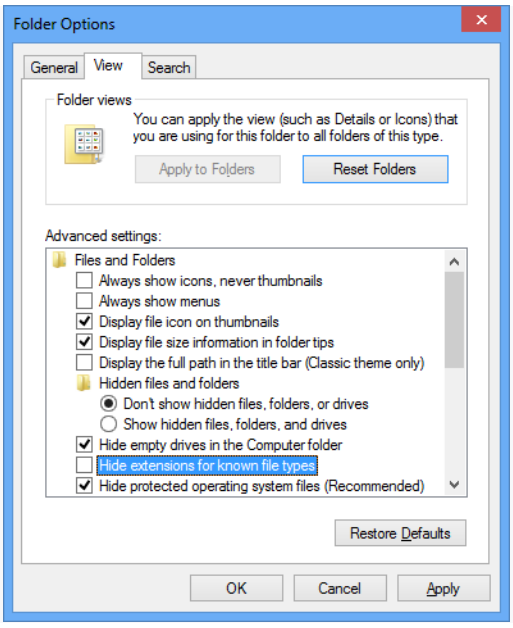
`C:\Users\YourName\Documents\Rainmeter\Skins\ConfigName\SkinName.ini`



The illustro "Clock" skin is organized in the Skins folder. Remember that skins are identified by their config name, e.g. "illustro\Clock".

1. For Windows Vista, 7 and 8. For default folder locations on Windows XP, see Installing Rainmeter.

Your First Skin



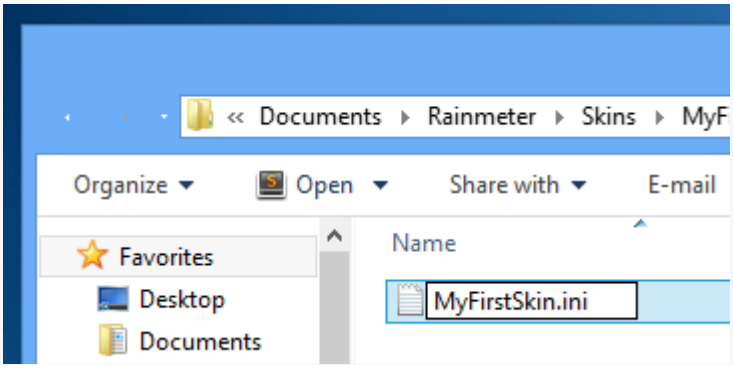
Until now, you have been working with the pre-made *illustro* skins that come with Rainmeter. Now, you're going to create a new skin from scratch. You will use this same skin throughout the tutorials that follow.

Open your Skins folder in Windows Explorer, and create a new folder. Call it `MyFirstSkin`. Next, you're going to create `MyFirstSkin.ini`.

Before you continue, there is a setting in Windows Explorer that you may want to change in order to make this process easier.

Press `Alt` to show the menu bar in Explorer, then select `Tools` →`Folder options`. Click the `View` tab, and uncheck the option labeled `Hide extensions for known file types`. Press `Ok` to apply the change.

You can now see the "extensions" that signify different file types in Windows Explorer, such as .txt *and* .ini. *This also allows you to change the extension when you rename a file.*



Open your new "MyFirstSkin" folder. Right-click inside the empty folder, and select `New` → `Text Document`. Windows will create the new file, and allow you to type in a new name.
Type `MyFirstSkin.ini`. Make sure to remove the ".txt" at the end of the file name. Windows will ask you if you're sure about changing the extension; click `Yes`.

Finally—open the Rainmeter context menu and click `Refresh all` to make Rainmeter see the new skin in your library. Then, open the skin in your text editor.

The "Hello, World!" Skin

The very first thing you're going to add to your skin is the [Rainmeter] section. This is a skin's "header" property, like the `<head>` tag in an HTML webpage. For now, your [Rainmeter] section will be mostly empty, except for one option:

```
[Rainmeter]

Update=1000
```

Select all

The Update option is what sets the length of the skin's update cycle. The length is given in milliseconds, or 1/1000ths of a second, so `Update=1000` means that the skin will update once per second. Updating is how the skin will react to changes in information. You'll see how this works in more detail later on.

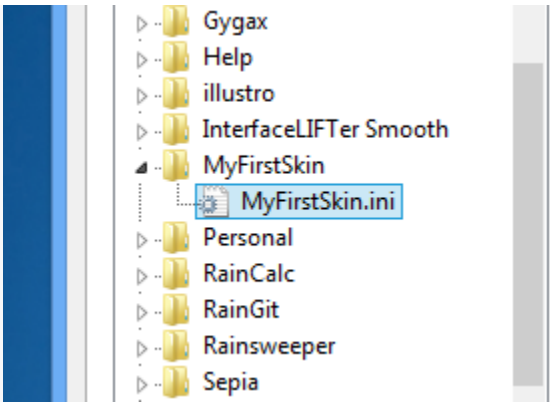
Now that you've given your skin a "head," it's time to give it a "body." You're going to create a string meter. This is one of the most common types of meters, and it is used to create **text**.

```
[Rainmeter]

Update=1000


[MyMeter]

Meter=String

Text=Hello, world!
```

Select all

The Meter option is required to tell Rainmeter that this section is, in fact, a meter. All meters have this option. The value of the option determines what *type* of meter it is.

The Text option, on the other hand, is unique to the string meter. As you might have guessed, this is where you provide a string of text for Rainmeter to display.

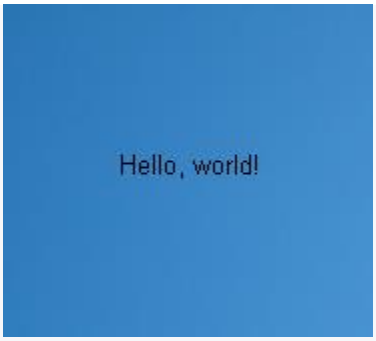


Believe it or not, what you have now is a complete, valid, working Rainmeter skin! Let's load it to see what it looks like. Load the skin using one of the methods that you learned before. You can either:

- Open the Manage window by left-clicking the Rainmeter tray icon, find `MyFirstSkin` in the skins list, then click the `Load` button in the upper-right.
- Open the context menu by right-clicking the tray icon, then select `Skins` → `MyFirstSkin` → `MyFirstSkin.ini`.

(As you get comfortable with Rainmeter's user interface, you'll decide whether you prefer working with the context menu or the Manage window.)

Now, look up in the top-left corner of your desktop. There's your skin!



Can you see me?

It's... not very big. Or pretty. Meters without any options tend to be very simple and unimpressive. So let's add some **formatting**.

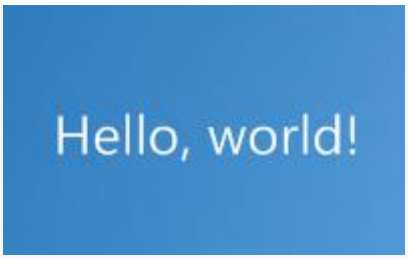
```
[MyMeter]

Meter=String

Text=Hello, world!

AntiAlias=1

FontColor=255,255,255

FontFace=Segoe UI

FontSize=20
```

Select all

Here's what we've added:

- AntiAlias
  A general meter option that smooths out the edges of a meter. This almost always improves the appearance of a string meter.
- FontColor
  A color option that changes the color of the text in this meter.
- FontFace
  An option that changes the font used for this meter. Rainmeter can use any font that you have installed in Windows, or another font in a skin's @Resources folder—but we'll get to that.
- FontSize
  The size of the font.

Now, let's apply these changes by refreshing the skin. Once again, you can either press `Refresh` in the Manage window, or `MyFirstSkin` → `Refresh skin` in the context menu.



Much nicer.

Congratulations! You have just created a new skin. You are now ready to move on to the Basic Tutorials. This series will guide you through the entire process of creating several example skins, while teaching you about the fundamental elements of a Rainmeter skin.

Basic Tutorials [home]

This is the first series of tutorials for Rainmeter. It is meant to follow Getting Started and assumes that you have gone through the steps described there. If you have not been through these steps, you should start here.

This series starts by creating a simple first skin, then covers the **anatomy of Rainmeter skins**, focusing on each of the basic skin properties in turn:

- **Meters**
  A skin's display elements. How to create objects that can be seen and clicked.
- **Measures**
  A skin's informational elements. How to pull data from your computer or the Internet.
- **Bangs**
  A skin's interactive elements. How to send commands to Rainmeter or other applications.
- **Updates**
  A skin's internal clock. How to control the timing and synchronicity of skin events.
- **Variables**
  A skin's data elements. How to manipulate independent strings that are used to store many kinds of information.

This series is in development. Check back soon for the first installment.

Launcher [home]

Creating a new skin

First, make sure you have gone through the guide on creating a simple "hello world" skin at Creating Skins so you understand how to create a .ini skin file in the Skins folder, and how to refresh Rainmeter so it sees your new entry.

So let's create a new folder under Skins, where we will store all of the skins we create in these tutorials. We want to keep them together in one root config folder, so we can later look at how to share some settings and resources between them.

In Windows Explorer, create a new folder called **Tutorials**. Under that folder, create another new folder called**Launcher**. In that **Tutorials\Launcher** folder, create a new empty text file. In Windows explorer, you can simply right-click in the folder and say "New / Text document". Give it the name **Launcher.ini** being sure that theextension is **.ini** and not **.ini.txt**.

Now, left-click the Rainmeter icon in the Windows notification area on your taskbar, to open the Manage dialog. Click on the **Refresh all** button on the bottom left, and you should see your new **Tutorials / Launcher** config in the list. Find **Launcher.ini** in the list, right-click it and say "Edit". This will open the new skin file in your default text editor. Don't load the skin just yet, we need to add some code first...

Building the Launcher skin

First, as we did in our earlier "hello world" skin, let's add the [Rainmeter] section to control the Update speed of the skin.

```
[Rainmeter]

Update=1000
```

Select all

Then, we can add our first meter to the skin.

```
[MeterLaunch1]

Meter=String

X=5

Y=5

FontFace=Trebuchet MS

FontSize=14

FontColor=255,255,255,255

StringStyle=Bold
```

```
AntiAlias=1

Text=Notepad

SolidColor=0,0,0,1

LeftMouseUpAction=["C:\Windows\System32\Notepad.exe"]
```
Select all

Note that the first two things we must do is create a [SectionName] for the meter, and tell Rainmeter what type of meter this is.

```
[MeterLaunch1]

Meter=String
```
Select all

This is a String meter, one of the most commonly used meters. It is used to display some text on the screen. Formatting control, like the position, size, color and font face can all be set with a combination of General meter options and the options specific to the String meter. Let's look at some we have used here.

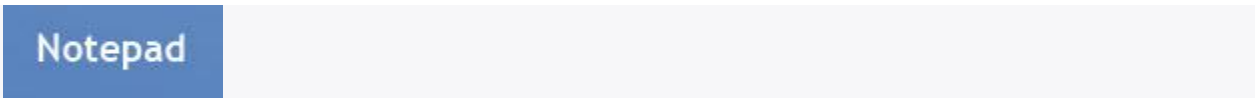
- X and Y: These control the position of the meter relative to the overall skin. So what we are saying here is that we want this first meter to be five pixels right of the left edge of the skin, and five pixels down from the top of the skin.
- FontFace: This determines which font you want to use when displaying the string.
- FontSize: The size in points for the font.
- FontColor: The color for the text.
- StringStyle: Contols some sytle options for the string. We are using a **Bold** style.
- AntiAlias: Does font smoothing on the text, to improve the display quality.

Then we are setting the Text option of the meter with `Text=Notepad`, defining the string of text we wish to display. String meters can also use the value of Measures as the text to display, we will go into that in more detail in a future tutorial.

The last two options in the meter are what really makes this into a "launcher", and not just some text on the screen.

- SolidColor=0,0,0,1: This is a little trick used to create a solid but invisible "box" behind the string meter, to make clicking on the text easier.
- LeftMouseUpAction=["C:\Windows\System32\Notepad.exe"]: This is a mouse action, telling Rainmeter to take the defined action option when the meter is clicked with the mouse. In this case, we are launching the application **Notepad.exe** found in the folder **C:\Windows\System32**. We will go into much more detail on different types of actions and Bangs in future tutorials.

So let's load our new skin and take a look at where we stand. From the Manage dialog, find the **Launcher.ini** entry in the list and click on the **Load** button on the upper right.



Drag the skin anywhere on the screen you like. Rainmeter will remember the position any time you load this skin in the future. You can also right-click the skin to change other skin options as desired.

There we have our first meter, and a fully functioning Rainmeter skin. Clicking on the **Notepad** text will launch the application. Congratulations! Take a short break.

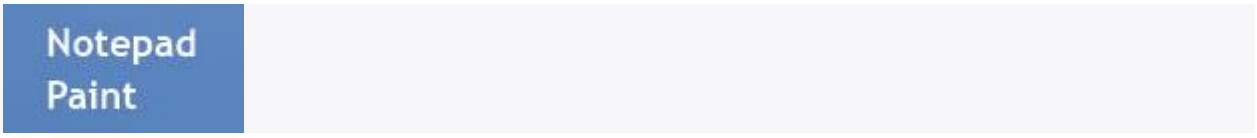
Adding another meter

Next, let's add another meter to launch another application. The format of the meter will be much the same as the earlier one, but we need to do some things to make sure the **position** of the meter is appropriate.

```
[MeterLaunch2]

Meter=String

X=0r

Y=2R

FontFace=Trebuchet MS

FontSize=14

FontColor=255,255,255,255

StringStyle=Bold

SolidColor=0,0,0,1

AntiAlias=1

Text=Paint

LeftMouseUpAction=["MSPaint.exe"]
```

Select all

The important changes here are in the X and Y options for the meter. Remember that **X** and **Y** set the position of a meter in the context of the overall skin. Note that we are using **relative positioning** to set the **X** option zero pixels relative to the left of the **previous meter**, (in effect the same **X** as before) and the **Y** option two pixels relative to the botton of the previous meter. We could also have specifically set the value of **X** and **Y** to some hard-coded value, but using relative positioning is often much easier when laying out a skin's meters.



We have changed the `Text` and `LeftMouseUpAction` options for our second application, and we can now use **Manage** and the **Refresh** button at the upper right to see our changes in action. Note that you can also right-click the skin on the screen and say **Refresh skin**.

A new meter type: The Image meter

Now we are going to shift gears a bit, and introduce a new meter type. What we want to do with the next meter is display an image, which we can click on to launch the application.

```
[MeterLaunch3Image]

Meter=Image

ImageName=#@#Images\Calc.png

W=32

H=32

X=0r

Y=2R

LeftMouseUpAction=["Calc.exe"]
```

Select all

We have created a new section that has the Meter option set to `Meter=Image`. We are going to use an image file we will include with our skin, so we have a couple of things to do to set that up correctly.

- First, we need to create a folder to hold all the images for our tutorial skins. We will use this folder not only for this **Launcher** skin, but other ones going forward.

  In Windows Explorer, create a new folder under **Skins\Tutorials** called **@Resources**. The @Resources folderis a special folder under the root config of a skin or suite of skins, used to hold images, sounds, fonts, include files and other shared resources for the skins.

- In this **Skins\Tutorials\@Resources** folder, create a new folder called **Images**. Download and copy the image below into that **Skins\Tutorials\@Resources\Images** folder.

Now let's go back to our skin code. We have set the ImageName option to use the built-in `#@#` shortcut for the**@Resources** folder, and the sub-folder of **Images** where we put our **calc.png** image file. The Image meter will load and display this image.

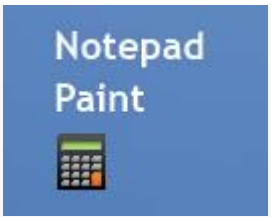
Next, we decided that the 128x128 pixels size of that original image file was just too large for our purposes. So we used the W and H (width and height) options to change the size of the image to 32x32 pixels when displayed. Just like the String meter, Image meters use both General meter options and options specific to the Image meter type.

Just as you did with the second String meter above, use relative positioning and the X and Y options to position the image below the previous meter.

```
ImageName=#@#Images\Calc.png

W=32

H=32

X=0r

Y=2R
```

Select all

Then, just as with the String meters above, set the **LeftMouseUpAction** option to `LeftMouseUpAction=["Calc.exe"]` so that the Windows Calculator is launched when the image is clicked.



Save your changes and refresh the skin. Click on the image to launch Windows Calculator.

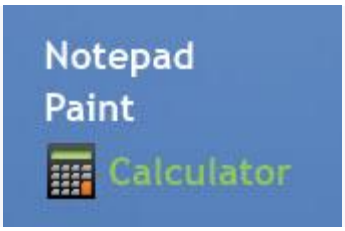
You could just use the image as is, but let's add a string label next to the image. For that, create a new String meter, much like the ones above.

```
[MeterLaunch3Text]

Meter=String

X=0R

Y=6r

FontFace=Trebuchet MS

FontSize=14

FontColor=146,197,94,255

StringStyle=Bold
```

```
SolidColor=0,0,0,1

AntiAlias=1

Text=Calculator

LeftMouseUpAction=["Calc.exe"]
```

Select all

Notice that we have used the X and Y options to set the position of the meter lined up just to the right of the previous meter (the Image meter), and six pixels below the top of it. That will align the meter more or less centered to the right of the image of the calculator. We add the same **LeftMouseAction** that the Image meter has, so you can click on either to launch the application. Just for fun, we also changed the **FontColor** of the string.



Save your changes and refresh the skin.

Having this skin just sorta float on the desktop is ok, but let's add a background meter to put a nice dark box behind the entire skin.

An important concept in Rainmeter is that fact that how meters that "overlap" in position display from the standpoint of which is in "front" and which is in "back" on the screen is determined by the position of the meter's code in the skin .ini file. Since we want our background meter to be "behind" all the other meters, we need to put the code for it before the other meters we want in front.

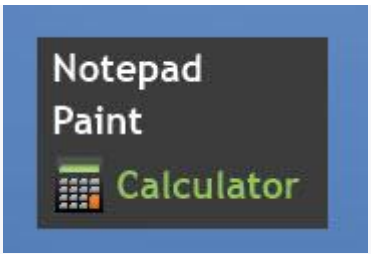
So, let's go back up to the top of the skin, just below the **[Rainmeter]** section, but before the first String meter**[MeterLaunch1]** and put in our new background Image meter.

```
[Rainmeter]

Update=1000


[MeterBackground]

Meter=Image

W=145

H=95

SolidColor=60,60,60,255
```

```
[MeterLaunch1]

Meter=String

X=5

Y=5

FontFace=Trebuchet MS

FontSize=14

FontColor=255,255,255,255

StringStyle=Bold

SolidColor=0,0,0,1

AntiAlias=1

Text=Notepad

LeftMouseUpAction=["C:\Windows\System32\Notepad.exe"]
```

Select all

We have set a specific W and H (width and height) so the meter is large enough to hold all the others, and no X and Y options. This is so the meter will be at the far left and top of the overall skin. We then set `SolidColor=60,60,60,255`, which in the absence of any `ImageName` on an Image meter will simply draw a square or rectangle based on the size of the **W** and **H** options.



Save your changes and refresh the skin.

Good job! You now have a pretty functional launcher skin that you can add items to, (just follow the patterns you used above) change string or image format options and positioning to modify to your tastes, and end up with something that is both functional and shows off your creativity.

Clock [home]

Creating your second tutorial skin

First, make sure you have gone through the guide on creating a simple "hello world" skin at Creating Skins so you understand how to create a .ini skin file in the Skins folder, and how to refresh Rainmeter so it sees your new entry.

In the previous tutorial, you should have already created a folder under Skins called **Tutorials**. We are going to add a new folder under that one to create our new skin.
Under **Skins\Tutorials** create a new folder called **Clock**.

In that **Tutorials\Clock** folder, create a new empty text file. In Windows explorer, you can simply right-click in the folder and say "New / Text document". Give it the name **Clock.ini** being sure that the extension is **.ini** and not**.ini.txt**.

Now, left-click the Rainmeter icon in the Windows notification area on your taskbar, to open the Manage dialog. Click on the **Refresh all** button on the bottom left, and you should see your new **Tutorials / Clock** config in the list. Find **Clock.ini** in the list, right-click it and say "Edit". This will open the new skin file in your default text editor. Don't load the skin just yet, we need to add some code first...

Building the Clock skin

This tutorial will introduce using Measures in a skin. Measures are used to obtain some information in Rainmeter, from your computer's system, text files, web sites, and other sources. In addition, we will be using some more features of the String meter and dip our toes a little deeper into using action options and Bangs in your skin.

First, as we did in our earlier tutorial, let's add the [Rainmeter] section to control the Update speed of the skin.

```
[Rainmeter]

Update=1000
```

Select all

Now let's add our first Measure, in this case a Time measure to retrieve information about the system time from your computer.

```
[MeasureTime]

Measure=Time

Format=%#I:%M
```

Select all

Do check out the manual entry for the Time measure to see how that Format option is used to obtain the time information you want. We are using a format of **"hour in 12 hour time:minutes"** for this measure.

So let's add a meter to display this value we obtained with our measure. First, we are going to use the MeterStyleoption by creating a **[TextStyle]** section to set up some common string formatting options. This way we won't have to repeat them in every meter we create. Then add the new **[MeterTime]** meter.

```
[TextStyle]

FontFace=Trebuchet MS

FontColor=255,245,207,255

SolidColor=0,0,0,1

StringStyle=Bold

StringAlign=Right

AntiAlias=1


[MeterTime]

Meter=String

MeterStyle=TextStyle

MeasureName=MeasureTime

X=165

Y=0

FontSize=40
```

Select all

The key to using measures and meters together is using the MeasureName option to "bind" the measure**[MeasureTime]** to the meter. What that means is that this meter will display the value returned by the measure on each Update.

So let's load our new skin and take a look at where we stand. From the Manage dialog, find the **Clock.ini** entry in the list and click on the **Load** button on the upper right.

Drag the skin anywhere on the screen you like. Rainmeter will remember the position any time you load this skin in the future. You can also right-click the skin to change other skin options as desired.

Adding more measures and meters

We are going to use various elements of the system time in different meters in our skin, so let's create some more measures to gather different types of information. Go back up to just under our [MeasureTime] section and add some new measures.

```
[MeasureSeconds]
Measure=Time
Format=%S

[MeasureAMPM]
Measure=Time
Format=%p

[MeasureMonthName]
Measure=Time
Format=%B

[MeasureDayOfMonth]
Measure=Time
Format=%#d

[MeasureYear]
Measure=Time
Format=%Y

[MeasureDayOfWeek]
Measure=Time
Format=%A
```

Select all

Now, start building meters to display all these different measure values. Let's start with the first one, so we can look at a new meter positioning concept. Head back down to the bottom of your skin and add this new meter.

```
[MeterSeconds]

Meter=String

MeterStyle=TextStyle

MeasureName=MeasureSeconds

X=204

Y=8

FontSize=18

FontColor=255,231,135,255
```

Select all

Note that we are using the `MeasureName=MeasureSeconds` option to bind this meter to the appropriate measure.

Now, if you look above, you will see that in the MeterStyle section **[TextStyle]** we have defined many format options for our string meters that we want to share with the meters rather than repeating them in each one. One of these is the StringAlign option, which will allow us to align the text in the meter based on the X and Y options for the meter. In this case, we are going to right-align all of our meters. You will see why as we continue with the layout. In the previous **[MeterTime]** meter, we aligned the meter so the right-most edge was at the **X** position of**165** in the skin. For this meter, we are aligning the right-most edge at the **X** position of **204**.

In addition, we want this meter to have a different color. So, we are "overriding" the`FontColor=255,245,207,255` option we set in the **[TextStyle]** MeterStyle, by specifically setting`FontColor=255,231,135,255` for this meter.

Save and refresh the skin to see the changes.

Continue creating the meters at the bottom of your skin, to display the information from the various measures.

```
[MeterAMPM]

Meter=String

MeterStyle=TextStyle
```

```
MeasureName=MeasureAMPM

X=204

Y=30

FontSize=16

FontColor=255,231,135,255


[MeterMonthDayYear]

Meter=String

MeterStyle=TextStyle

MeasureName=MeasureMonthName

MeasureName2=MeasureDayOfMonth

MeasureName3=MeasureYear

X=204

Y=0R

FontSize=13

Text=%1 %2, %3


[MeterDayOfWeek]

Meter=String

MeterStyle=TextStyle

MeasureName=MeasureDayOfWeek

X=204

Y=0R

FontSize=13
```

Select all

As you can see, there is a different approach to binding measures to meters in the **[MeterMonthDayYear]** meter above.

```
[MeterMonthDayYear]

Meter=String

MeterStyle=TextStyle
```

```
MeasureName=MeasureMonthName

MeasureName2=MeasureDayOfMonth

MeasureName3=MeasureYear

X=204

Y=0R

FontSize=13

Text=%1 %2, %3
```

Select all

Since we want to use the value of three different measures in this meter, we use the MeasureName option to define all three measures as being bound to this meter. Then we can use the Text option with **%1 %2, %3** to display each of the three measure values, including adding a hard coded comma for formatting.

Save and refresh the skin to see the changes.

Now you can see why we wanted to right-align our string meters. As the time changes and becomes longer and shorter, we don't want the position of the meters to change. We want to keep things lined up and tidy. The following image demonstrates how it looks when the time is much longer.

Let's add a background image like we did in the earlier tutorial, by going up above all the existing meters and adding a new Image meter.

```
[MeterBackground]

Meter=Image

W=210

H=107
```

```
SolidColor=60,60,60,255


[TextStyle]
FontFace=Trebuchet MS
FontColor=255,245,207,255
SolidColor=0,0,0,1
StringStyle=Bold
StringAlign=Right
AntiAlias=1
```

Select all

Save and refresh the skin to see the changes. Congratulations! Nice looking clock skin.

Some extra credit work

We are going to add a couple more features to our clock, so we can touch again on using Action options andBangs in your skins.

First, let's add the ability to toggle the display of the time between 12-hour and 24-hour when we hover the mouse over the **[MeterTime]** meter. Find that meter, and add a couple of lines.

```
[MeterTime]
Meter=String
MeterStyle=TextStyle
MeasureName=MeasureTime
X=165
Y=0
FontSize=40
MouseOverAction=[!SetOption MeasureTime Format "%H:%M"][!UpdateMeasure MeasureTime][!UpdateMeter *][!Redraw]
```

```
MouseLeaveAction=[!SetOption MeasureTime Format "%#I:%M"][!UpdateMeasure MeasureTime][!UpdateMeter *][!Redraw]
```

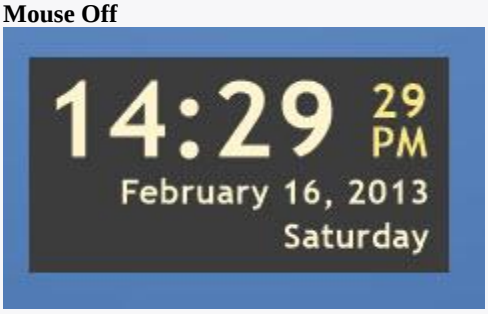
Select all

This is using a Mouse action and a handful of Bangs.

What we are doing when we move the mouse over the meter is to use the !SetOption bang to change the `Format`option of the **[MeasureTime]** measure to obtain the time from the system using the code for 24-hour time (%H) instead of 12-hour time (%I). Then we are using the !UpdateMeasure, !UpdateMeter, and !Redraw bangs to have the change take place as soon as we move the mouse over, and not wait for the next update of the skin.

When we move the mouse away from the meter, we are using the same combination of bangs to set the format back to 12-Hour time and update things.

Mouse Off



Mouse Over

"It tolls for thee"

Next, we can add some sound to our clock. What we will do is have the skin play a .wav file of a grandfather clock striking when the time is exactly on the hour.

First, we need get the sound file and put it in the right location in your skin folders. Right-click the link below and save the file **HourChime.wav** to your computer. Put it in a new **Sounds** folder in our **Skins\Tutorials\@Resources**folder we created before. So it will be **Skins\Tutorials\@Resources\Sounds\HourChime.wav**

Download HourChime.wav

Now we can add a new measure to control when the sound is played. Let's go up to the top of the skin, right after the **[MeasureTime]** measure we already have, and add a new one.

```
[MeasureTime]

Measure=Time

Format=%#I:%M


[MeasureChimeHour]

Measure=Time

Format=%M

IfEqualValue=0

IfEqualAction=[Play "#@#Sounds\HourChime.wav"]
```

Select all

This **[MeasureChimeHour]** measure is getting the number of **minutes** (%M) from the system time, and then using an IfAction to check the value of the measure on each update. If the value is equal to zero `IfEqualValue=0`, then the associated `IfEqualAction=[Play "#@#Sounds\HourChime.wav"]` is executed. This uses the Play bang to load and play the **HourChime.wav** file.

After making the changes, save and refresh the skin. Congratulations! You now have a beautiful, functional AND annoying skin!

We have jumped around a bit, so here is the entire completed skin so you can check that your code is the same.

```
[Rainmeter]

Update=1000


[MeasureTime]

Measure=Time

Format=%#I:%M


[MeasureChimeHour]

Measure=Time

Format=%M

IfEqualValue=0

IfEqualAction=[Play "#@#Sounds\HourChime.wav"]
```

```
[MeasureSeconds]
Measure=Time
Format=%S

[MeasureAMPM]
Measure=Time
Format=%p

[MeasureMonthName]
Measure=Time
Format=%B

[MeasureDayOfMonth]
Measure=Time
Format=%#d

[MeasureYear]
Measure=Time
Format=%Y

[MeasureDayOfWeek]
Measure=Time
Format=%A

[MeterBackground]
Meter=Image
W=210
H=107
SolidColor=60,60,60,255
```

```
[TextStyle]

FontFace=Trebuchet MS

FontColor=255,245,207,255

SolidColor=0,0,0,1

StringStyle=Bold

StringAlign=Right

AntiAlias=1


[MeterTime]

Meter=String

MeterStyle=TextStyle

MeasureName=MeasureTime

X=165

Y=0

FontSize=40

MouseOverAction=[!SetOption MeasureTime Format "%H:%M"][!UpdateMeasure MeasureTime][!UpdateMeter *][!Redraw]

MouseLeaveAction=[!SetOption MeasureTime Format "%#I:%M"][!UpdateMeasure MeasureTime][!UpdateMeter *][!Redraw]


[MeterSeconds]

Meter=String

MeterStyle=TextStyle

MeasureName=MeasureSeconds

X=204

Y=8

FontSize=18

FontColor=255,231,135,255


[MeterAMPM]

Meter=String
```

```
MeterStyle=TextStyle

MeasureName=MeasureAMPM

X=204

Y=30

FontSize=16

FontColor=255,231,135,255


[MeterMonthDayYear]

Meter=String

MeterStyle=TextStyle

MeasureName=MeasureMonthName

MeasureName2=MeasureDayOfMonth

MeasureName3=MeasureYear

X=204

Y=0R

FontSize=13

Text=%1 %2, %3


[MeterDayOfWeek]

Meter=String

MeterStyle=TextStyle

MeasureName=MeasureDayOfWeek

X=204

Y=0R

FontSize=13
```

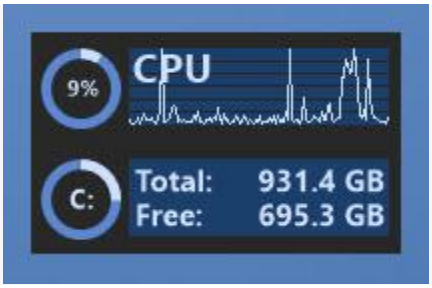
Select all

System [home]

Introduction

With this tutorial, we are going to cover several things. First, we are going to introduce two new measure and two new meter types, as well as some new options for controlling how they work. We will also touch on the use ofUpdateDivider on measures to control the performance of a skin, and will explore using Variables to set, use and change values shared by measures and meters.

There is a lot going on in this skin, so let's explore the end-result a bit so you can get a sense of where we are going as we work through it. First, let's take a look at the final skin.



What we are going to do is measure the percent usage of the CPU, and the space on a couple of hard drives. Then we are going to create some circular meters showing the usage with Roundline meters. We are going to monitor the CPU with a Line meter, and display some information about the space on your hard drives. You only see one hard drive in that image? We will get to that...

Building the System skin

First, as we did in our earlier tutorial, let's add the [Rainmeter] section to control the Update speed of the skin.

```
[Rainmeter]

Update=1000
```

Select all

The next thing we are going to do is define some variables for the skin. Variables are used to create a value, which can be used repeatedly in the skin by enclosing the variable name in ## (example: **#VarName#**). In order to change the value in all places it is used, you only need to change it one time in the [Variables] section of the skin.

```
[Variables]

DarkBlue=27,63,107,255

MediumBlue=92,135,209,255

LightBlue=207,224,255,255

LightRed=250,148,135,255

AlmostBlack=40,40,40,255
```

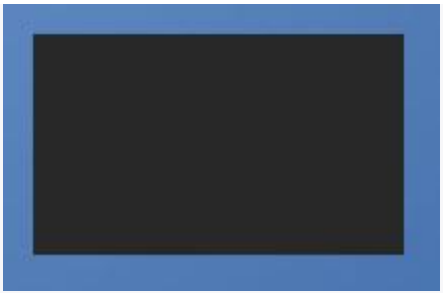
```
CurrentDrive=C:

Drive1=C:

Drive2=D:
```
Select all

Now we will cheat a bit. In earlier tutorials, we waited until the end and put a solid background on the skin using anImage meter. Let's go ahead and add that now, so we can demonstrate the first use of one of those **variables** we created above.

```
[MeterBackground]

Meter=Image

X=0

Y=0

W=185

H=110

SolidColor=#AlmostBlack#
```
Select all

Note that we have used the variable **#AlmostBlack#** in the SolidColor option for this meter. The meter will then use the value we defined as `AlmostBlack=40,40,40,255` in the **[Variables]** section earlier. Remember that if you define SolidColor on an image meter with **no MeasureName** option, it simply draws a square or rectangle based on the W and H (width and height) options.

So let's load our new skin and take a look at where we stand. From the Manage dialog, find the **System.ini** entry in the list and click on the **Load** button on the upper right.



Not much to look at just yet...

Next, let's create a measure to get the current amount of CPU usage, as a percentage, on each update.

```
[MeasureCPU]
```

```
Measure=CPU
```
Select all

Now it is time to introduce a new meter type. We are going to display the value of that **[MeasureCPU]** measure as filling a cirle on the screen using a Roundline meter. Create a new section with this code:

```
[MeterCPUCircle]

Meter=Roundline

MeasureName=MeasureCPU

X=5

Y=8

W=40

H=40

StartAngle=(Rad(270))

RotationAngle=(Rad(360))

LineStart=15

LineLength=20

Solid=1

LineColor=#LightBlue#

AntiAlias=1
```
Select all

Refresh the skin so we can see what it is doing as we walk through the options.

What the **Roundline** meter does is draw a line that rotates around the center of a circle defined by the W and Hoptions of the meter. In this case, we have defined the size of the meter (and thus the circle) as **40x40 pixels**. Keep in mind that this defines a circle that has a total width of 40 pixels, and a width from the **center** to the edge as 20 pixels.

Roundline - Working with angles

We first needed to define two options to control how the meter works. We need to tell the Roundline where the line should point when the value is **0%** (the starting point). This is done with the StartAngle option. We also need to tell the Roundline how much of the circle to use as **100%** (the distance to travel when the value is 100%). This is done with the RotationAngle option. As you can see, we have set the value of **StartAngle** to `StartAngle=(Rad(270))`, and the value of **RotationAngle** to `RotationAngle=(Rad(360))`. Let's talk about that for a minute.

The **angles** in a **Roundline** meter are defined in **Radians**. Radians are a unit of measure used to define degrees of distance around a circle, starting with **0°**, which is **the point directly to the right of the center of the circle**. This is important! Do not picture a **compass** in your head, which has **0°** (North) pointing "up". Radians are defined with**0°** pointing to the **right**.

I highly recommend taking a minute and reading through the explanation at Radians Guide.

So beginning with StartAngle, we are telling the Roundline that the starting point of the meter is a **distance** of **270° degrees** in radians from the 0° position of directly right. That will move the starting point around clockwise to the "top" of the meter, which is what we want.

Next, we are setting the RotationAngle option to **360° degrees** in radians, which tells the Roundline that the**distance** to travel around the circle from the starting point defined in StartAngle will be the full circle (a circle has 360 degrees). We want the entire circle to fill when the value of the measure is 100%.

*Note that we are using the Rad(x) function to make it easier to convert **degrees** (which are easy to picture in our heads) into **radians** (which are mathematically useful, but stupidly complicated).*

Roundline - The line options

By default, the **Roundline** meter will draw a single line from the center of the meter to the outside edge, pointing to the position represented by the value of the measure. We don't want a "pointer" in this case, but want to draw and fill a circle. There are few options we have used to control this.

- LineStart=15
  This tells the Roundline that we want the line to start 15 pixels from the center of the meter.
- LineLength=20
  This sets the overall length of the line to 20 pixels. Remember that our meter is 40 pixels wide in total, but only 20 pixels wide from the center to the outside edge.

  The result of these two options is a line that starts 15 pixels from the center, and extends to the full 20 pixels defined as the length from the center to the edge. So in effect, a 5 pixel line at the outside edge.

- Solid=1
  This option tells the Roundline that instead of a single line, we want the meter to "fill" from the position defined as 0% to the current value represented by the measure. So instead of a "pointer", we get a circle being filled as the value changes.

Then we are using one of the variables we defined in the **[Variables]** section at the beginning. We want to set the color of the circle to a nice light blue, which we defined as `LightBlue=207,224,255,255`. So we set the option in the Roundline meter to `LineColor=#LightBlue#`.

Good job! Roundline is a very useful and flexible meter type in Rainmeter, and once you play with the options and different behaviors you will find lots of creative ways to use it. Take a breather...

Continuing the skin

That Roundline meter looks a little funny by itself, so let's add another Roundline meter to serve as a "background" for it. In your code, move up **above** the **[MeterCPUCircle]** section we added before, and insert a new meter section. Remember, we want this meter to be "behind" **[MeterCPUCircle]**, so it needs to be "before" it in the actual .ini skin code.

```
[MeterCPUCircleBack]

Meter=Roundline

X=5

Y=8

W=40

H=40

StartAngle=(Rad(270))

RotationAngle=(Rad(360))

LineStart=15

LineLength=20

Solid=1

LineColor=#MediumBlue#

AntiAlias=1
```

Select all

What we are doing here is creating another Roundline meter, at the same X and Y position as the one that comes after, but behind it. All of the options in **[MeterCPUCircleBack]** are the same as those in **[MeterCPUCircle]**, with the exception that there is no MeasureName option to "bind" a measure to the meter. Roundline will by default use a value of **100%** in this case, in effect allowing us to draw a full circle that we can use as a background. Oh, we also used a different variable for the color of the line. Refresh the skin to see the change.

Your entire skin code should now look like this:

```
[Rainmeter]

Update=1000


[Variables]

DarkBlue=27,63,107,255

MediumBlue=92,135,209,255

LightBlue=207,224,255,255

LightRed=250,148,135,255

AlmostBlack=40,40,40,255

CurrentDrive=C:

Drive1=C:

Drive2=D:


[MeasureCPU]

Measure=CPU


[MeterBackground]

Meter=Image

X=0

Y=0

W=185

H=110

SolidColor=#AlmostBlack#
```

```
[MeterCPUCircleBack]
Meter=Roundline
X=5
Y=8
W=40
H=40
StartAngle=(Rad(270))
RotationAngle=(Rad(360))
LineStart=15
LineLength=20
Solid=1
LineColor=#MediumBlue#
AntiAlias=1

[MeterCPUCircle]
Meter=Roundline
MeasureName=MeasureCPU
X=5
Y=8
W=40
H=40
StartAngle=(Rad(270))
RotationAngle=(Rad(360))
LineStart=15
LineLength=20
Solid=1
LineColor=#LightBlue#
AntiAlias=1
```

Select all

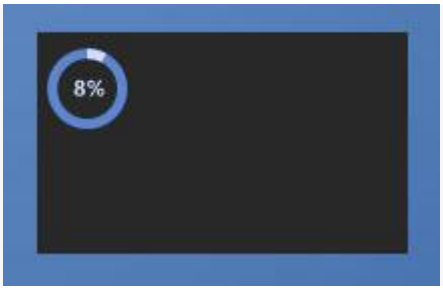
Next we want to create a String meter showing the actual value we are getting in the **[MeasureCPU]** measure, and center it inside our circle. We have talked about string meters

and positioning in earlier tutorials, so let's just add the following as a new section at the end of your skin code.

```
[MeterCPUPercent]

Meter=String

MeasureName=MeasureCPU

FontFace=Segoe UI

FontSize=8

FontColor=#LightBlue#

X=26

Y=28

StringAlign=CenterCenter

StringStyle=Bold

AntiAlias=1

Text=%1%
```

Select all

Save and refresh the skin to see your new meter.



You know, it would be nice if the color of that text changed when the CPU usage hits some value we define. What we can do is change the color to "red" if the value is at or above 25%, and back to "blue" if it is below that value.

Go back near the top of the code and find the the **[MeasureCPU]** measure. Let's add a few lines to it.

```
[MeasureCPU]

Measure=CPU

IfAboveValue=24

IfAboveAction=[!SetOption MeterCPUPercent FontColor #LightRed#][!UpdateMeter *][!Redraw]

IfBelowValue=25
```

```
IfBelowAction=[!SetOption MeterCPUPercent FontColor #LightBlue#][!UpdateMeter *][!Redraw]
```

Select all

This is using IfAction options to do the following:

- If the value of the measure moves above 24, use the !SetOption bang to change the **FontColor** option on the**[MeterCPUPercent]** meter to the value of the variable **#LightRed#** we defined earlier in **[Variables]**.
- If the value of the measure falls below 25, use the !SetOption bang to change the **FontColor** option on the meter to the value of the variable **#LightBlue#** we defined earlier in **[Variables]**, in effect changing it back to the original color.

Line meter

Next we are going to touch on another meter type in Rainmeter. The Line meter displays the percentage value of a measure (or more than one measure) as a series of points over time, connected to create lines on a graph.

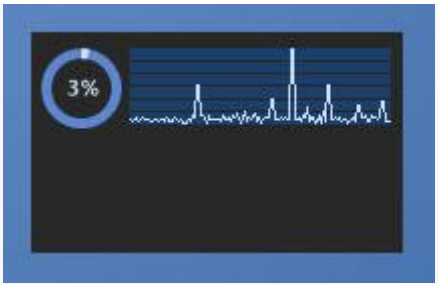
Add a new section to the skin.

```
[MeterCPULine]

Meter=Line

MeasureName=MeasureCPU

X=49

Y=8

W=130

H=38

LineCount=1

LineColor=#LightBlue#

LineWidth=1

HorizontalLines=1

HorizontalLineColor=#AlmostBlack#
```

```
SolidColor=#DarkBlue#
```

Select all

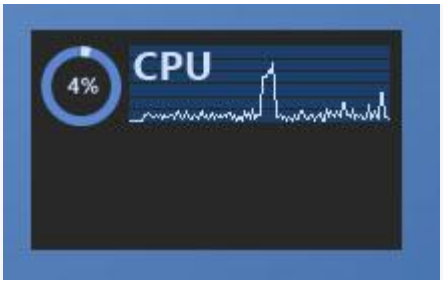
The Line meter is pretty self-explanatory, you "bind" one or more measures to the meter with MeasureName, and then use options like LineWidth and LineColor to control how the meter looks. The line is plotted over time on a graph that is defined by the W and H options of the meter. The additional HorizontalLines and HorizontalLineColoroptions draw the background lines on the graph, in the desired color. Finally, we set a SolidColor option on the meter to define a overall background color. Refresh the skin and have a look. Let it run for a while so you can see the movement of the line.



We should put a label on the Line meter, so it is clear what it is measuring. We have looked at the String meter previously, so let's just add one and take a look at the result.

```
[MeterCPUText]

Meter=String

FontFace=Segoe UI

FontSize=15

FontColor=#LightBlue#

X=48

Y=4

StringStyle=Bold

AntiAlias=1

Text=CPU
```

Select all



Measuring a hard drive

It is time to introduce a new measure type to our skin. We are going to be measuring the **total**, **used**, and **free**space on one of your hard drives. We will display the **used** space on a circular Roundline meter almost exactly as we did with the CPU measurement, then display the other information, **total** and **free** space, in string meters next to it.

First, let's set up the measures you will need. Back up in your skin code, right under the **[MeasureCPU]** we created in the beginning, and add some new measures.

```
[MeasureDriveTotal]

Measure=FreeDiskSpace

Drive=#CurrentDrive#

Total=1

IgnoreRemovable=0

DynamicVariables=1

UpdateDivider=-1


[MeasureDriveFree]

Measure=FreeDiskSpace

Drive=#CurrentDrive#

IgnoreRemovable=0

DynamicVariables=1

UpdateDivider=5


[MeasureDriveUsed]

Measure=FreeDiskSpace

Drive=#CurrentDrive#

InvertMeasure=1

IgnoreRemovable=0

DynamicVariables=1

UpdateDivider=5
```

Select all

The FreeDiskSpace measure obtains space information about a drive. There are several options we are using that should be explained.

- Drive=#CurrentDrive#
  This option tells the measure which drive to examine. In this case, we are using one of the variables we defined earlier, to set the value to **C:**.
- IgnoreRemovable=0
  By default, FreeDiskSpace will ignore all removable drives like USB or optical drives. Set **IgnoreRemovable=0**to enable measuring these kinds of drives.
- InvertMeasure=1
  By default, FreeDiskSpace will measure the free space on a drive. To measure the used space, you use theGeneral Measure Option **InvertMeasure=1** to reverse what is measured.

In addition, we are using two other options on the measures we have not looked at before.

DynamicVariables is used to tell a measure or meter that any variables used in the section should be re-evaluated on each update of the section. In order to have Rainmeter use as few resources as possible, variables are only evaluated when the measure or meter is created, and will not detect any dynamic changes to variables it uses, if this option is not set. It will become clear in a little bit why we want this option on our FreeDiskSpace measures.

UpdateDivider is used to control how often a measure or meter is updated. The overall Update of the skin is set in the **[Rainmeter]** section at the top of the skin, and defines how often in milliseconds the skin is updated. In our case, and by default, this is `Update=1000` or once a second.

The `UpdateDivider=5` option we are setting on these FreeDiskSpace measures tells Rainmeter to update these measures every 5 updates of the skin, or in our case every 5 seconds. UpdateDivider should be considered for any measures that don't need to be updated as often as the Update option alone, to reduce the amount of work that Rainmeter has to do. In the case of FreeDiskSpace, this is particularly useful as actually reading the drives once every second would be more resource hungry than is really needed.

See the guide at Update Guide for a lot more detail and explanation.

Ok, our measures are all set up, and providing the various kinds of space measurements for our drive in bytes. One thing that should be noted is that the FreeDiskSpace measure also automatically sets the MinValue andMaxValue of the measure, so the value can be used in meters that require a Percentage, like the **Roundline** meter we are about to create.

Add the following new meters to the skin.

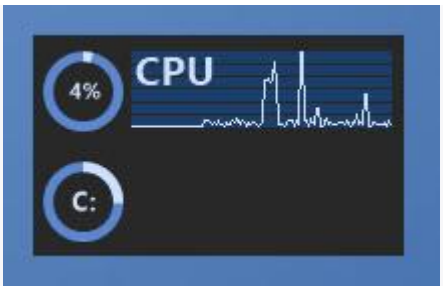
```
[MeterDriveCircleBack]

Meter=Roundline

X=5

Y=63

W=40
```

```
H=40
StartAngle=(Rad(270))
RotationAngle=(Rad(360))
LineStart=15
LineLength=20
Solid=1
LineColor=#MediumBlue#
AntiAlias=1

[MeterDriveCircle]
Meter=Roundline
MeasureName=MeasureDriveUsed
X=5
Y=63
W=40
H=40
StartAngle=(Rad(270))
RotationAngle=(Rad(360))
LineStart=15
LineLength=20
Solid=1
LineColor=#LightBlue#
AntiAlias=1

[MeterDriveCircleLabel]
Meter=String
FontFace=Segoe UI
FontSize=10
FontColor=#LightBlue#
X=26
```

```
Y=83

StringAlign=CenterCenter

StringStyle=Bold

Percentual=1

AntiAlias=1

DynamicVariables=1

Text=#CurrentDrive#
```

Select all

Note that these are pretty much identical to the meters we created earlier to show the CPU usage as a circularRoundline meter. We bind the active Roundline meter **[MeterDriveCircle]** to the measure **[MeasureDriveUsed]** to display the amount of "used" space, and instead of showing any measured value inside the circle in**[MeterDriveCircleLabel]**, we simply create a label with the value of the current drive letter we created in the variable **#CurrentDrive#**. We again set `DynamicVariables=1` on this label meter. We will see why shortly. Save and refresh the skin to see how we are doing.



Finally, we can create a nice background to match the one behind the Line meter we created in **[MeterCPULine]**, by simply creating an Image meter with a SolidColor and no MeasureName option, then some String meters to display the other drive space information, **total** and **free**. Add the following meters to the bottom of the skin. Save and refresh.

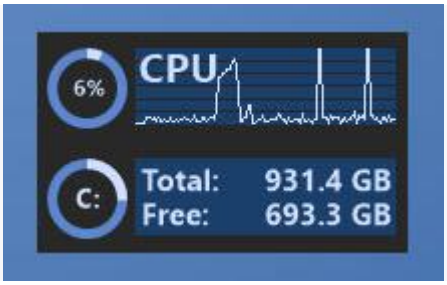
```
[MeterDriveBack]

Meter=Image

X=49

Y=63

W=130

H=38

SolidColor=#DarkBlue#


[MeterDriveTotalLabel]
```

```
Meter=String
FontFace=Segoe UI
FontSize=11
FontColor=#LightBlue#
X=50
Y=64
StringStyle=Bold
StringAlign=Left
AutoScale=1
AntiAlias=1
Text=Total:

[MeterDriveTotal]
Meter=String
MeasureName=MeasureDriveTotal
FontFace=Segoe UI
FontSize=11
FontColor=#LightBlue#
X=180
Y=64
StringStyle=Bold
StringAlign=Right
AutoScale=1
AntiAlias=1
Text=%1B

[MeterDriveFreeLabel]
Meter=String
FontFace=Segoe UI
FontSize=11
```

```
FontColor=#LightBlue#

X=50

Y=81

W=130

H=18

ClipString=1

StringStyle=Bold

AutoScale=1

AntiAlias=1

Text=Free:


[MeterDriveFree]

Meter=String

MeasureName=MeasureDriveFree

FontFace=Segoe UI

FontSize=11

FontColor=#LightBlue#

X=180

Y=81

StringStyle=Bold

StringAlign=Right

AutoScale=1

AntiAlias=1

Text=%1B
```

Select all

One new option we have used in two of these meters should be explained.
The FreeDiskSpace measure obtains the space values from the drive in **bytes**. For most of us, that is going to be a really huge and long number, (my one-terabyte C: drive has a total of 1000097181696 bytes for instance) and probably not one we want to display on a skin.
The AutoScale option on the string meter will automatically scale the number to **megabytes**, **gigabytes**,**terabytes** etc. and append the appropriate **M / G / T** label to the end. We have added an extra hard-coded **B** at the end of the text, so it will display as for instance: **931 GB**. Let's take a look.



What about those dynamic variables?

I promised we would get back to the reason why we added DynamicVariables=1 to a few measures and meters. Our last goal is to have the skin dynamically change the **Drive** that we measure and display when we move the mouse over the skin.

If you look back to the **[Variables]** section, you will see that we set the following:

```
CurrentDrive=C:

Drive1=C:

Drive2=D:
```

Select all

Then in our FreeDiskSpace measures **[MeasureDriveTotal]**, **[MeasureDriveFree]**, and **[MeasureDriveUsed]**, we set the Drive option to that variable **#CurrentDrive#**. So to start with, the measures are looking at the **C:** drive. What we want to do is set the value of **#CurrentDrive#** to be equal to the value of the variable **#Drive2#** when we move the mouse over the skin, and back to **#Drive1#** when we move the mouse away.

To accomplish this, we need to add new Mouse actions to the skin, and some Bangs that are executed by the mouse actions.

Let's go back up to our **[MeterBackground]** meter, the first one after the measures in the skin. Change it by adding our mouse action lines.

```
[MeterBackground]

Meter=Image

X=0

Y=0

W=185

H=110

SolidColor=#AlmostBlack#

MouseOverAction=[!SetVariable CurrentDrive #Drive2#][!UpdateMeasure *][!UpdateMeter *][!Redraw
]
```

```
MouseLeaveAction=[!SetVariable CurrentDrive #Drive1#][!UpdateMeasure *][!UpdateMeter *][!Redraw]
```

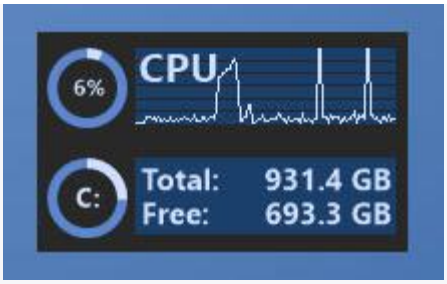
Select all

What this is saying is:

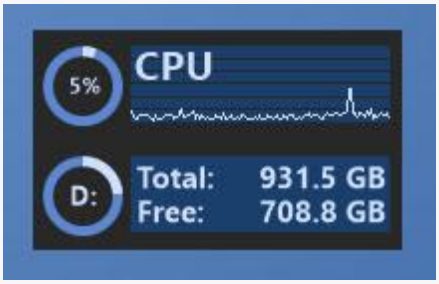
- If the mouse moves over the background meter, use the !SetVariable bang to change the value of the variable**CurrentDrive** to the value of the variable **Drive2**.
- If the mouse moves away from the background meter, use the !SetVariable bang to change the value of the variable **CurrentDrive** to the value of the variable **Drive1**, or in effect back to the original value.
- Then we are using the !UpdateMeasure, !UpdateMeter, and !Redraw bangs to have the change take place as soon as we move the mouse over or away, and not wait for the next update of the skin.

This functionality is why we added DynamicVariables=1 to the measures and meters which are using the**#CurrentDrive#** variable. That option allows the measures and meters to re-evaluate the variable on each update of the section, so when we change it with **!SetVariable** they react to the change.

So we are finally there... Great job! Here are two shots of your final skin, the first with the mouse away from the skin and the second with the mouse over the skin.



Mouse Off



Mouse Over

And here is the final code so you can compare to yours.

```
[Rainmeter]

Update=1000
```

```
[Variables]
DarkBlue=27,63,107,255
MediumBlue=92,135,209,255
LightBlue=207,224,255,255
LightRed=250,148,135,255
AlmostBlack=40,40,40,255
CurrentDrive=C:
Drive1=C:
Drive2=D:

[MeasureCPU]
Measure=CPU
IfAboveValue=24
IfAboveAction=[!SetOption MeterCPUPercent FontColor #LightRed#][!UpdateMeter *][!Redraw]
IfBelowValue=25
IfBelowAction=[!SetOption MeterCPUPercent FontColor #LightBlue#][!UpdateMeter *][!Redraw]

[MeasureDriveTotal]
Measure=FreeDiskSpace
Drive=#CurrentDrive#
Total=1
IgnoreRemovable=0
DynamicVariables=1
UpdateDivider=-1

[MeasureDriveFree]
Measure=FreeDiskSpace
Drive=#CurrentDrive#
IgnoreRemovable=0
DynamicVariables=1
```

```
UpdateDivider=5


[MeasureDriveUsed]
Measure=FreeDiskSpace
Drive=#CurrentDrive#
InvertMeasure=1
IgnoreRemovable=0
DynamicVariables=1
UpdateDivider=5


[MeterBackground]
Meter=Image
X=0
Y=0
W=185
H=110
SolidColor=#AlmostBlack#
MouseOverAction=[!SetVariable CurrentDrive #Drive2#][!UpdateMeasure *][!UpdateMeter *][!Redraw
]
MouseLeaveAction=[!SetVariable CurrentDrive #Drive1#][!UpdateMeasure *][!UpdateMeter *][!Redra
w]


[MeterCPUCircleBack]
Meter=Roundline
X=5
Y=8
W=40
H=40
StartAngle=(Rad(270))
RotationAngle=(Rad(360))
LineStart=15
```

```
LineLength=20
Solid=1
LineColor=#MediumBlue#
AntiAlias=1

[MeterCPUCircle]
Meter=Roundline
MeasureName=MeasureCPU
X=5
Y=8
W=40
H=40
StartAngle=(Rad(270))
RotationAngle=(Rad(360))
LineStart=15
LineLength=20
Solid=1
LineColor=#LightBlue#
AntiAlias=1

[MeterCPUPercent]
Meter=String
MeasureName=MeasureCPU
FontFace=Segoe UI
FontSize=8
FontColor=#LightBlue#
X=26
Y=28
StringAlign=CenterCenter
StringStyle=Bold
```

```
AntiAlias=1
Text=%1%

[MeterCPULine]
Meter=Line
MeasureName=MeasureCPU
X=49
Y=8
W=130
H=38
LineCount=1
LineColor=#LightBlue#
LineWidth=1
HorizontalLines=1
HorizontalLineColor=#AlmostBlack#
SolidColor=#DarkBlue#

[MeterCPUText]
Meter=String
FontFace=Segoe UI
FontSize=15
FontColor=#LightBlue#
X=48
Y=4
StringStyle=Bold
AntiAlias=1
Text=CPU

[MeterDriveCircleBack]
Meter=Roundline
```

```
X=5
Y=63
W=40
H=40
StartAngle=(Rad(270))
RotationAngle=(Rad(360))
LineStart=15
LineLength=20
Solid=1
LineColor=#MediumBlue#
AntiAlias=1

[MeterDriveCircle]
Meter=Roundline
MeasureName=MeasureDriveUsed
X=5
Y=63
W=40
H=40
StartAngle=(Rad(270))
RotationAngle=(Rad(360))
LineStart=15
LineLength=20
Solid=1
LineColor=#LightBlue#
AntiAlias=1

[MeterDriveCircleLabel]
Meter=String
FontFace=Segoe UI
```

```
FontSize=10
FontColor=#LightBlue#
X=26
Y=83
StringAlign=CenterCenter
StringStyle=Bold
Percentual=1
AntiAlias=1
DynamicVariables=1
Text=#CurrentDrive#

[MeterDriveBack]
Meter=Image
X=49
Y=63
W=130
H=38
SolidColor=#DarkBlue#

[MeterDriveTotalLabel]
Meter=String
FontFace=Segoe UI
FontSize=11
FontColor=#LightBlue#
X=50
Y=64
StringStyle=Bold
StringAlign=Left
AutoScale=1
AntiAlias=1
```

```
Text=Total:


[MeterDriveTotal]
Meter=String
MeasureName=MeasureDriveTotal
FontFace=Segoe UI
FontSize=11
FontColor=#LightBlue#
X=180
Y=64
StringStyle=Bold
StringAlign=Right
AutoScale=1
AntiAlias=1
Text=%1B


[MeterDriveFreeLabel]
Meter=String
FontFace=Segoe UI
FontSize=11
FontColor=#LightBlue#
X=50
Y=81
W=130
H=18
ClipString=1
StringStyle=Bold
AutoScale=1
AntiAlias=1
Text=Free:
```

```
[MeterDriveFree]

Meter=String

MeasureName=MeasureDriveFree

FontFace=Segoe UI

FontSize=11

FontColor=#LightBlue#

X=180

Y=81

StringStyle=Bold

StringAlign=Right

AutoScale=1

AntiAlias=1

Text=%1B
```

Select all

Installing Rainmeter [home]

To install Rainmeter, download the latest final or beta version from rainmeter.net, then follow the instructions below.

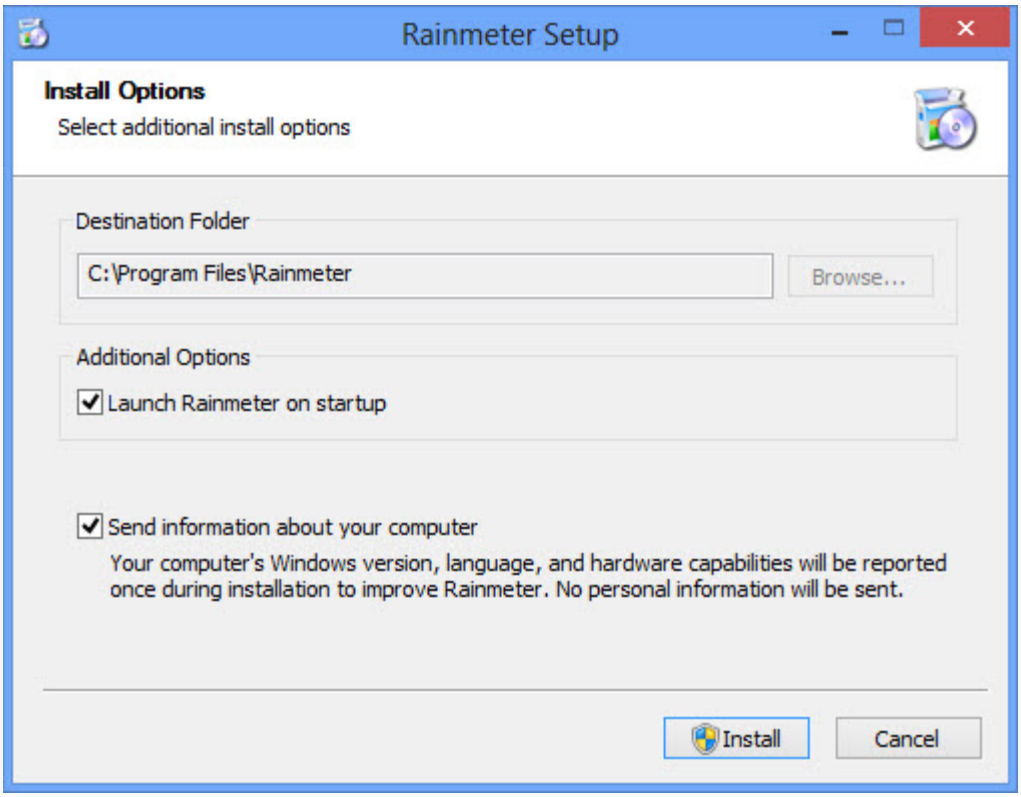
If you are reinstalling or updating Rainmeter, you do not need to uninstall your existing copy before continuing. Your settings, skins and plugins will be preserved.

Standard Installation

The standard installation is recommended for most users. Simply run the installer and follow the instructions.


Rainmeter Setup
Please select the installer language.
English - English
OK
Cancel


Rainmeter Setup
Rainmeter
desktop customization tool
Welcome to the Rainmeter Setup
Select the type of install:
Standard installation (recommended)
Installs Rainmeter and downloads required libraries.
Portable installation
Do not select this unless you know what you're doing. Shortcuts will not be created and the .rmskin extension will not be registered. The Visual C++ 2010 and .NET 2.0 libraries may need to be manually installed.
Next >
Cancel



The installation will do the following:

- Install the program to the default or selected **Destination folder**.
- Install required C++ and .NET runtime libraries if needed.
- Create a Windows file association for the .rmskin file extension and the Rainmeter Skin Installer.
- Create Windows Start menu item for Rainmeter.
- Create Windows Start menu item for Startup/Rainmeter, so Rainmeter starts with Windows. This may be disabled by unchecking **Launch Rainmeter on startup**.
- Launch Rainmeter at the end of the installation.

Skins and settings folders will be created in the default file locations when Rainmeter is first run.

Note: Rainmeter can be uninstalled using the **Add or Remove programs** function in Windows.

Portable Installation

To run Rainmeter from a single folder, so it can be copied to another computer or run directly from a removable drive, select **Portable installation** during the install process and browse to the desired installation folder. No changes to the Windows Registry or Start menu will be made. All program, skins and settings folders and files will remain in the selected program folder.

Note: The required C++ and .NET runtime libraries may need to be manually installed.

Default File Locations

Program folder:
`C:\Program Files\Rainmeter`

Skins folder:
Windows XP `C:\Documents and Settings\YourName\My Documents\Rainmeter\Skins` [1]
Windows 8/7/Vista `C:\Users\YourName\Documents\Rainmeter\Skins` [1]

Settings and Layouts folder:
Windows XP `C:\Documents and Settings\YourName\Application Data\Rainmeter` [1]
Windows 8/7/Vista `C:\Users\YourName\Appdata\Roaming\Rainmeter` [1]

1. "YourName" is an example.

Silent Installation (Advanced)

To install Rainmeter without user interaction, use the installer command line switches.

- **/S** - Must be specified to enable silent install.
- **/D=** - Install directory (do not enclose in quotes). This parameter is **required** for portable installs, but it's optional for standard installations. If not given, the install directory will be the previous (if found from registry) or default (%PROGRAMFILES%\Rainmeter) location.
- **/VERSION=** - Set to 64 for a 64bit install (optional).
- **/PORTABLE=** - Set to 1 for a portable install (optional).
- **/DESKTOPSHORTCUT=** - Set to 1 to create desktop shortcut (optional).
- **/STARTUP=** - Set to 1 to launch automatically with Windows (optional).
- **/ALLUSERS=** - Set to 1 to create shortcuts for all users (optional).

For example:

```
"Rainmeter-2.4-r1700-beta.exe" /S /DESKTOPSHORTCUT=1 /ALLUSERS=1
```

The return value (error-code) will be:

- **0** - Success!
- **1** - Unknown error.
- **2** - Unknown error.
- **3** - Version of Windows version not supported.
- **4** - Administrative rights not required.
- **5** - Failed to write to install directory.
- **6** - VC++ Redistributable 2010 is not installed.
- **7** - Unable to close Rainmeter/RainBrowser.

User Interface [home]

Rainmeter can be controlled using several user interface windows built into the application. Choose one of the items to see details on how they are used to operate the application, install, load and manage skins, monitor and debug during skin editing, and save and load the current skin layout and settings.

Manage

The primary management tool for Rainmeter. Load and unload skins, control settings for the application and skins, load and save layouts and more.

About

Much more than the usual simple About dialog in applications, the About window in Rainmeter provides powerful tools for monitoring the current status of the application and skins. Monitor the log file for information and errors, watch current values of variables and measures for running skins, and view the status and version of the application and plugins.

Context Menus

In addition to the Manage window, Rainmeter can be completely controlled from context menu items available by right-clicking the application's notification area (system tray) icon or an individual skin on the screen.

Manage [home]

The Manage window in Rainmeter is the primary means of controlling the application and skins. It consists of three main tabs:

- **Skins**: Displays a list of installed and loaded skins. This tab is used to view information about skins, manage skin settings, and control buttons to load/unload/refresh skins.
- **Layouts**: Used to save and load the current state of Rainmeter. Active and inactive skins, skin positions, and other Rainmeter settings.
- **Settings**: Controls high-level Rainmeter settings such as the interface language and logging.

Manage is accessed in several ways:

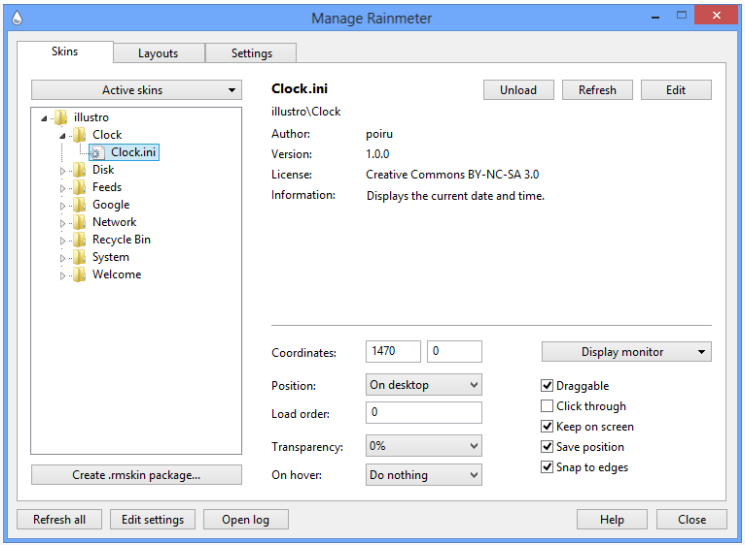
- **Left-Click the Rainmeter icon in the Windows Notification Area on the taskbar.**
- Right-Click the Rainmeter icon in the Windows Notification Area on the taskbar and select "Manage".
- Using the !Manage bang as an action in a skin or from the command line as a parameter to Rainmeter.exe.
- Right-Click any running skin on the desktop and select "Manage skin".

Buttons used to control Rainmeter:

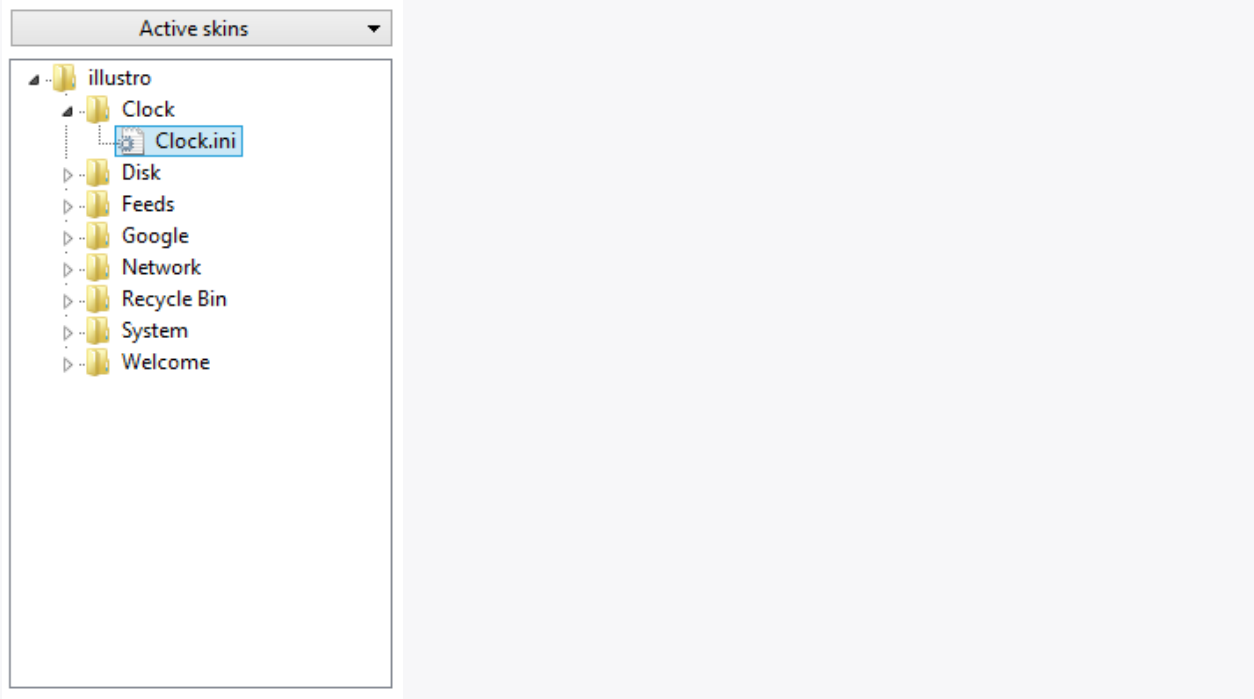
- **Refresh all**: Refresh the entire Rainmeter application, including all currently active skins.
- **Edit settings**: Manually edit the settings in Rainmeter.ini with the text editor associated with .ini files.
- **Open log**: View the Log tab of the Rainmeter About window.

- **Create .rmskin package**: Package skins(s) for distribution in the .rmskin format.

The Skins tab



There are four main areas in this tab.



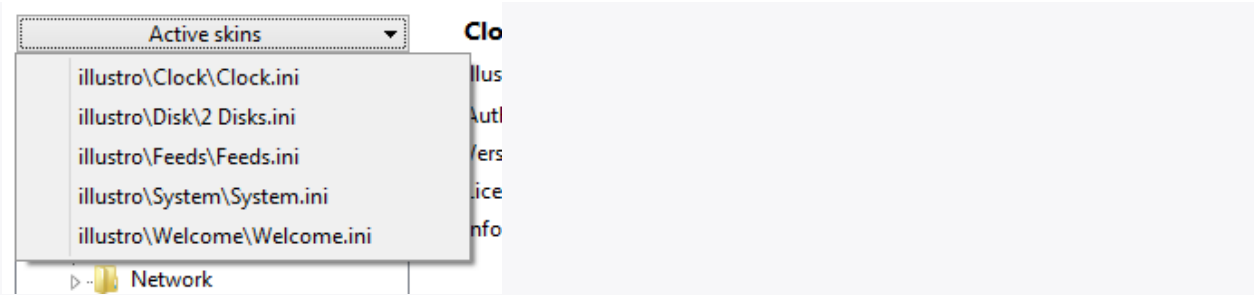
The skins list

List of currently installed skins. This contains all Skins found when Rainmeter is started or refreshed.

The list consists of the config folder for each skin, and the skin .ini files for each config.

- Clicking on a skin .ini file will make that skin active in the Manage tab.
- Double-clicking a skin .ini file will unload the skin if it is running, or load it if not.
- Right clicking on a config folder will allow opening the folder in Windows Explorer.
- Right clicking on a skin .ini file will allow loading, unloading or editing the skin.

The list is updated when Rainmeter is refreshed.



Active skins

This pull-down will contain a list of all currently loaded and active skins in Rainmeter.

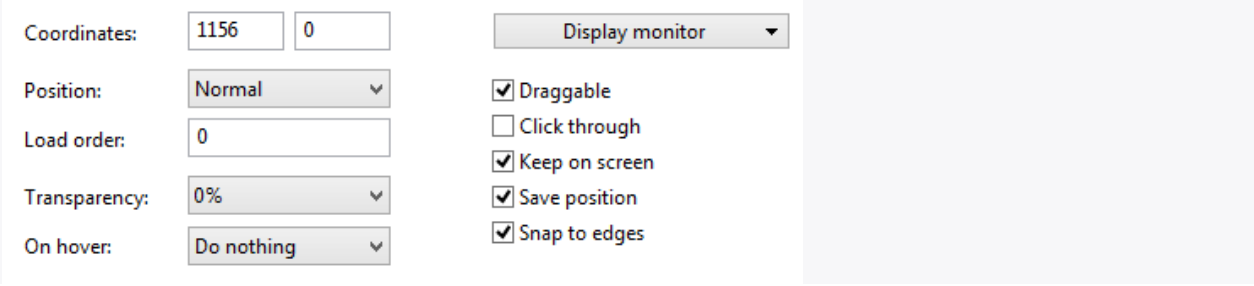
Clicking on a skin will make that skin active in the Manage tab.

Metadata

Displays the information in the [Metadata] section of the selected skin.

This includes **Name**, **Config**, **Author**, **Version**, **License**and **Information** fields.

If a skin does not contain a [Metadata] section, the **Add metadata** link in this area will add an empty section with all fields.



Skin Settings

For a selected active skin, shows the current values of various settings. Changes will immediately effect the skin on the desktop.

- **Coordinates**: The window X and Y location of the skin on the screen in pixels.
- **Position**: The z-position of the skin on the desktop relative to other windows.
- **Load order**: The loading order of the skin on the desktop relative to other skins.
- **Transparency**: The transparency of the skin.
- **On hover**: The hide on mouse over behavior of the skin.
- **Draggable**: The draggable setting for the skin.
- **Click through**: The click through setting for the skin.
- **Keep on screen**: The keep on screen setting for the skin.
- **Save position**: The save position setting for the skin.
- **Snap to edges**: The snap to edges setting for the skin.

- **Display monitor**: Settings for the monitor on which the skin is displayed.

  Use default: Primary monitor: Removes the `@N` directive from WindowX/WindowY settings.
  @0, @1, @2, ... , @32: Adds the specified monitor number to WindowX/WindowY settings. `@0` represents"The Virtual Screen".
  Auto-select based on window position: If checked, the WindowX/WindowY `@N` settings are made automatically based on the position of the meter's window. This setting will be unchecked when a specific monitor is selected.

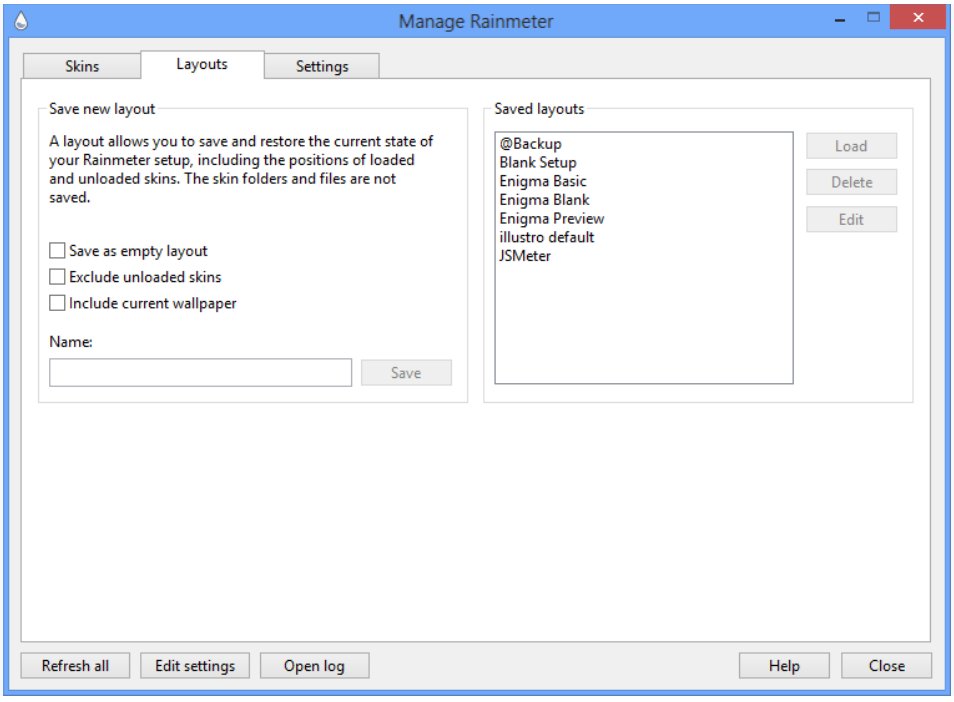
Buttons used to control skins:

- **Unload / Load**: Unload (make inactive) the selected skin if it is currently active, or load it if not.
- **Refresh**: Refresh the selected active skin.
- **Edit**: Edit the selected skin with the text editor associated with .ini files.

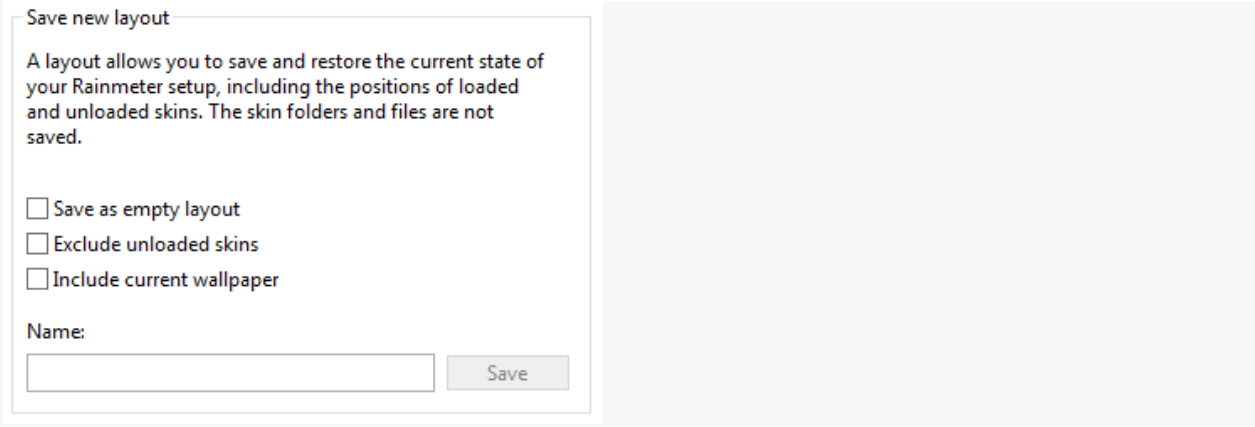
The Layouts tab

Layouts in Rainmeter are a way to save and load the current state of the Rainmeter settings. This saves the positions of currently active and inactive skins, as well as all other settings stored in the current Rainmeter.ini file. The layout can then be loaded to restore any saved state. Layouts are saved in the Rainmeter settings folder.

Note: The skin folders and files themselves are not saved with a layout.



There are two main areas in this tab.

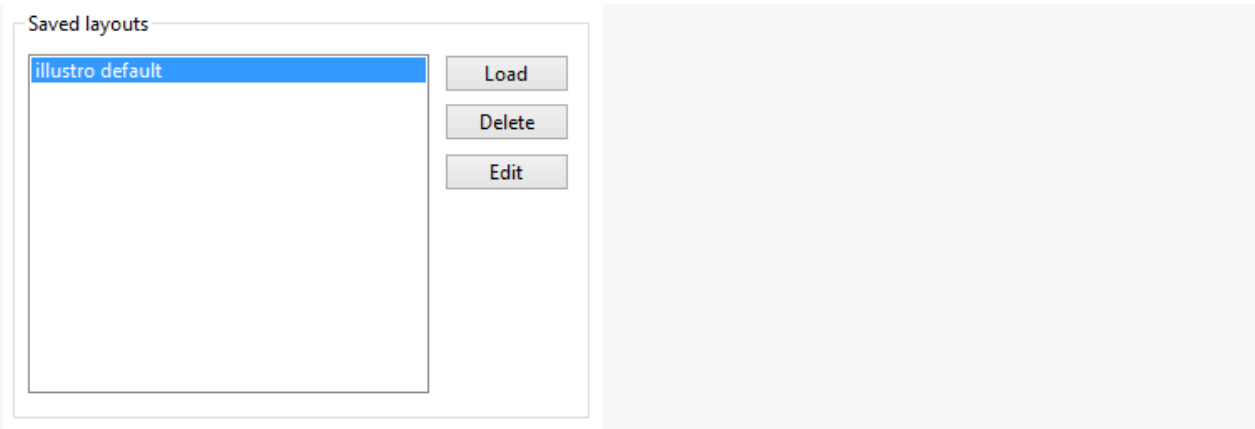


Save new layout

Enter the desired **Name:** and click **Save**.

- **Save as empty layout**
  Removes all [ConfigName] sections before saving.
- **Exclude unloaded skins**
  Removes all **inactive** [ConfigName] sections before saving.
- **Include current wallpaper**
  Saves the current Windows desktop wallpaper with the layout.

Note: If an existing layout is selected from the **Saved layouts** list or typed in, saving will replace the existing saved layout with the current state.



Saved layouts

Click on any layout name in the list.

- **Load**
  Loads the selected layout. If a Windows desktop wallpaper was saved with the layout, it will be applied to the desktop.
- **Delete**
  Permanently deletes the saved layout.
- **Edit**
  Edits the saved layout (Rainmeter.ini) file with the text editor associated with .ini files.

Global options under [Rainmeter] are not replaced when a layout is loaded, preserving local settings such as:

- ConfigEditor
- SkinPath
- DisableVersionCheck
- Language

When loading a layout, the current Rainmeter state will automatically be saved as a layout named **@Backup**.

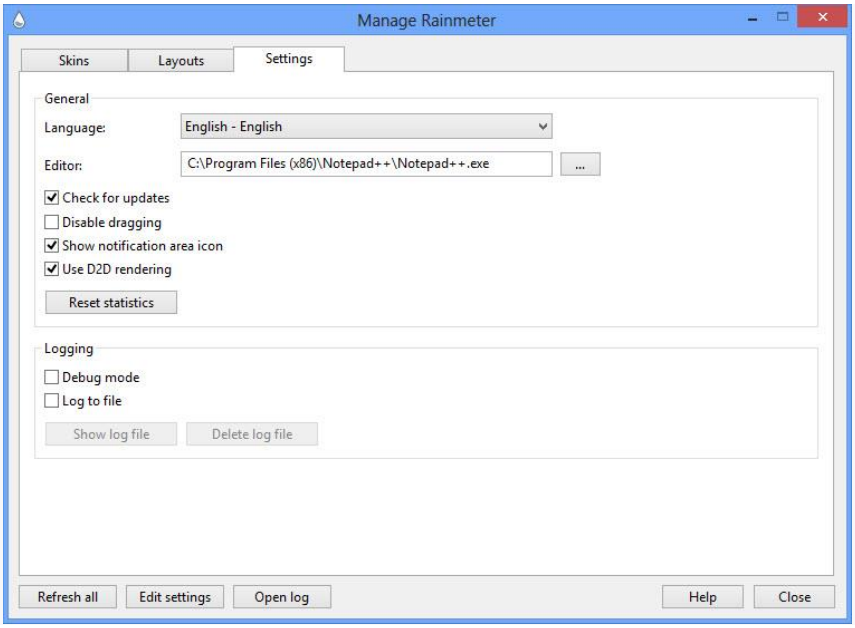
Hint: A layout can be loaded from the Windows command line using the !LoadLayout bang.

```
"C:\Program Files\Rainmeter\Rainmeter.exe" !LoadLayout "My Saved Layout"
```

The current Rainmeter state will be replaced with the named layout. If Rainmeter is not running, it will be started.

The Settings tab

This tab has some high level settings for the Rainmeter application, as well as controls for Rainmeter's logging capability.



General

- **Language:**
  Use the pull-down menu to select the desired language for all Rainmeter user interfaces. This does not have any effect on languages used in skins.
- **Editor:**
  Enter or browse to the text editor that will be used when "Edit skin" or "Edit settings" is selected.
- **Check for updates**
  If selected, Rainmeter will check online when started to see if the running version is the most recent release version, and will prompt the user to upgrade if not. This has no effect on beta versions of Rainmeter.
- **Disable dragging**
  If selected, automatically sets the **draggable** state of all active skins to prevent dragging skins with the mouse.
- **Show notification area icon**
  Shows or hides the Rainmeter icon in the Windows notification area (system tray).
- **Use D2D rendering**
  Use Direct2D rendering in Rainmeter. This requires Windows 8, or Windows 7 with the Platform Updateapplied.
- **Reset statistics**
  When clicked, clears all saved **network** and other statistics from the Rainmeter.stat file in the settings folder.

Logging

In addition to the real-time logging of errors, warnings and notices in the Log tab of the Rainmeter About window, Rainmeter can log activity to a **Rainmeter.log** text file, which will be created in the settings folder.

- **Debug mode**
  If selected, a more detailed logging mode will be used in the About window, and when **Log to file** is active. This should only be used when debugging a problem, as leaving this level of detailed logging on can impact Rainmeter performance.
- **Log to file**
  If selected, Rainmeter will append log entries to the Rainmeter.log file while running. Unchecking this item will turn off logging, but the Rainmeter.log file will be retained.
- **Show log file**
  If clicked, the Rainmeter.log file will be loaded in the text editor associated with .log files.
- **Delete log file**
  If clicked, the Rainmeter.log file will be deleted. If **Log to file** is currently active, it will automatically be turned off.

About [home]

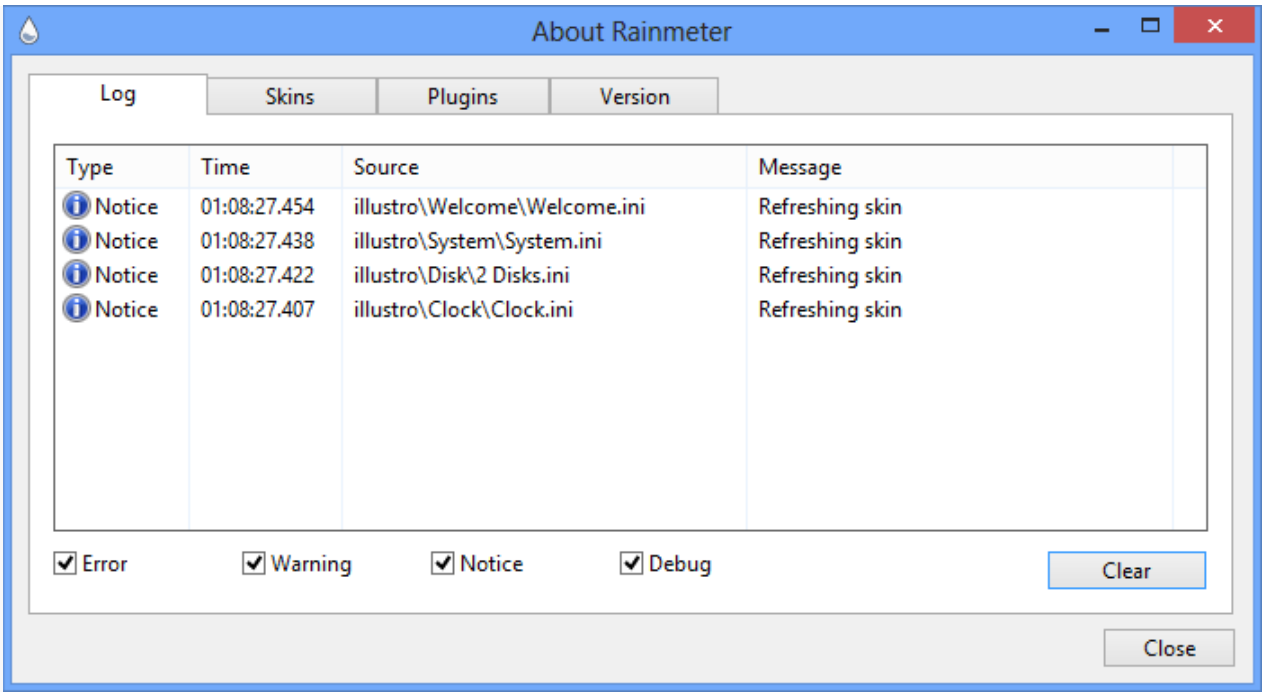
The About interface in Rainmeter is much more than the usual version number and link to a website. It is a powerful tool for finding and fixing problems with Rainmeter and skins. It consists of four main tabs:

- **Log**: Displays a running log of notifications and errors.
- **Skins**: Real-time display of important information about active skins.
- **Plugins**: Information about built-in and 3rd-party plugins installed in Rainmeter.
- **Version**: Version and installed path information.

About is accessed in several ways:

- **Right-Click the Rainmeter icon in the Windows Notification Area on the taskbar and select "About".**
- Click the **Open log** button in the Manage window.
- Using the !About bang as an action in a skin or from the command line as a parameter to Rainmeter.exe.
- Right-Click any running skin on the desktop and select "Rainmeter / About".

The Log tab



When opened, the list displays the last 20 messages logged by Rainmeter. While left open, the log continues to add new messages to the top of the list.

The log has three columns of information:

- **Type**: The category of message:

  Error - An error impacting the functionality of Rainmeter or a skin.
  Warning - An warning that some functionality is outside of normal parameters.
  Notice - Normal activity logged to provide information about the success of some action.
  Debug - Messages generated when Rainmeter or a skin is in Debug mode.

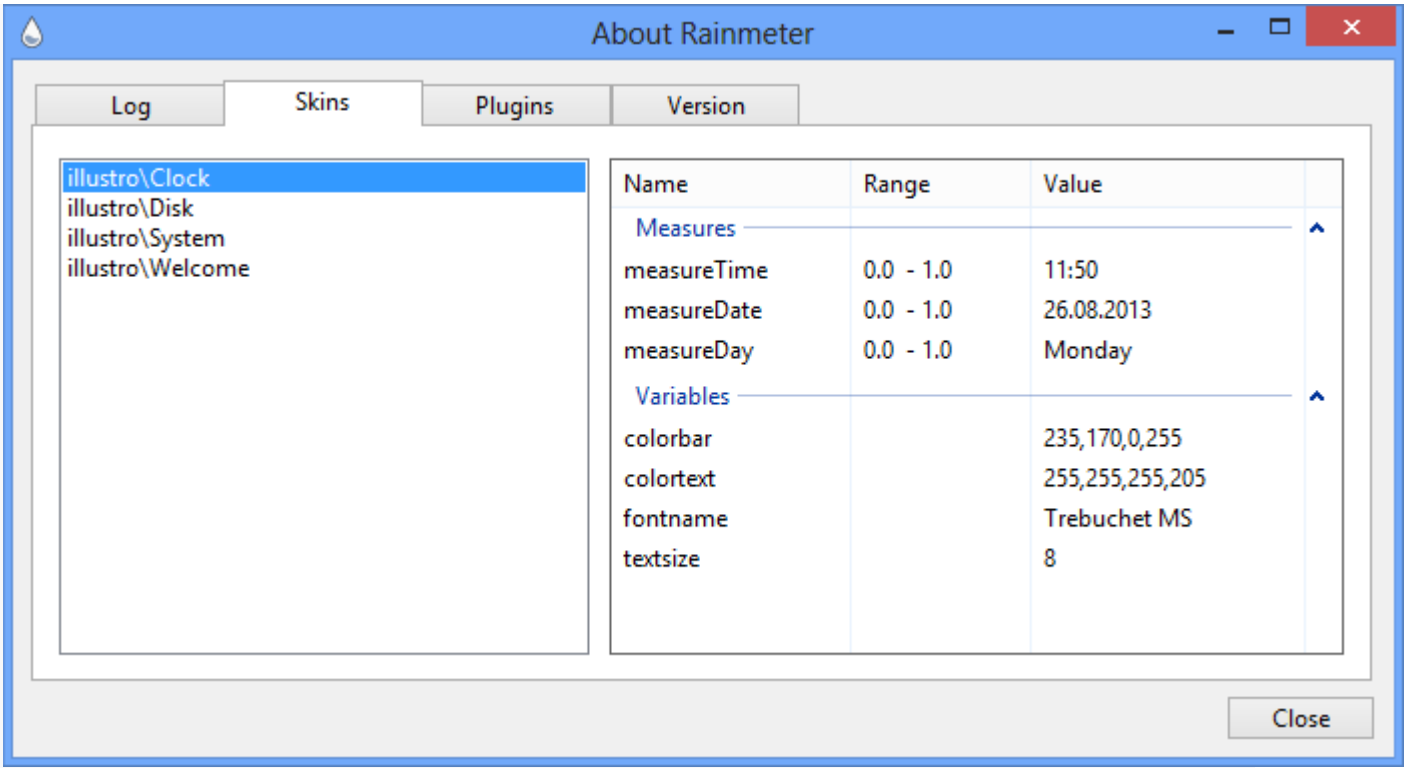
- **Time**: The time in Hours:Minutes:Seconds.Milliseconds since Rainmeter was started.
- **Message**: The text of the log message. Select a line and hit **CTRL-C** to copy the message to the Windows clipboard.

The types of messages that are displayed by the log can be controlled with the checkboxes at the bottom of the list. This can be used to filter the display to only certain kinds of new messages.

Click **Clear** to remove the current contents of the log panel.

Note: Custom log entries can be created using the !Log bang from a skin, or the print() function in the Lua scripting language. This can be very useful for debugging purposes.

The Skins tab



Extremely powerful tool displaying real-time information about currently active skins. The tab displays a list of currently active skin config names in the panel on the left. Selecting one from the list will update important information about the skin in real time, in the panel on the right.

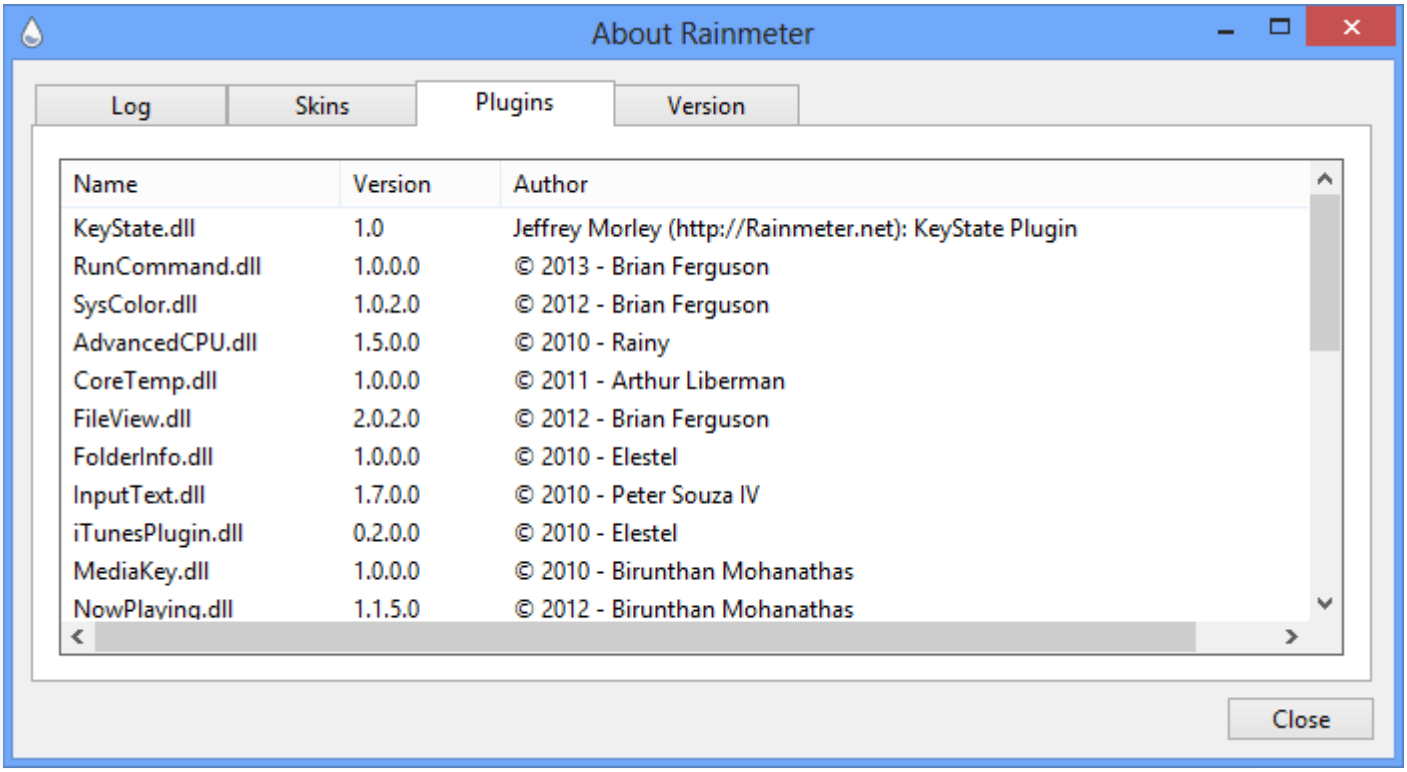
- **Measures**: Information about measures in the selected skin.

  Name - The name of the measure.
  Range - The current MinValue and MaxValue values for the measure, expressed as a range of values.
  Value - The current string value of the measure. Select a line and hit **CTRL-C** to copy the value to the Windows clipboard.

- **Variables**: Information about variables in the selected skin.

  Name - The name of the variable.
  Value - The current value of the variable. Select a line and hit **CTRL-C** to copy the value to the Windows clipboard.

The Plugins tab

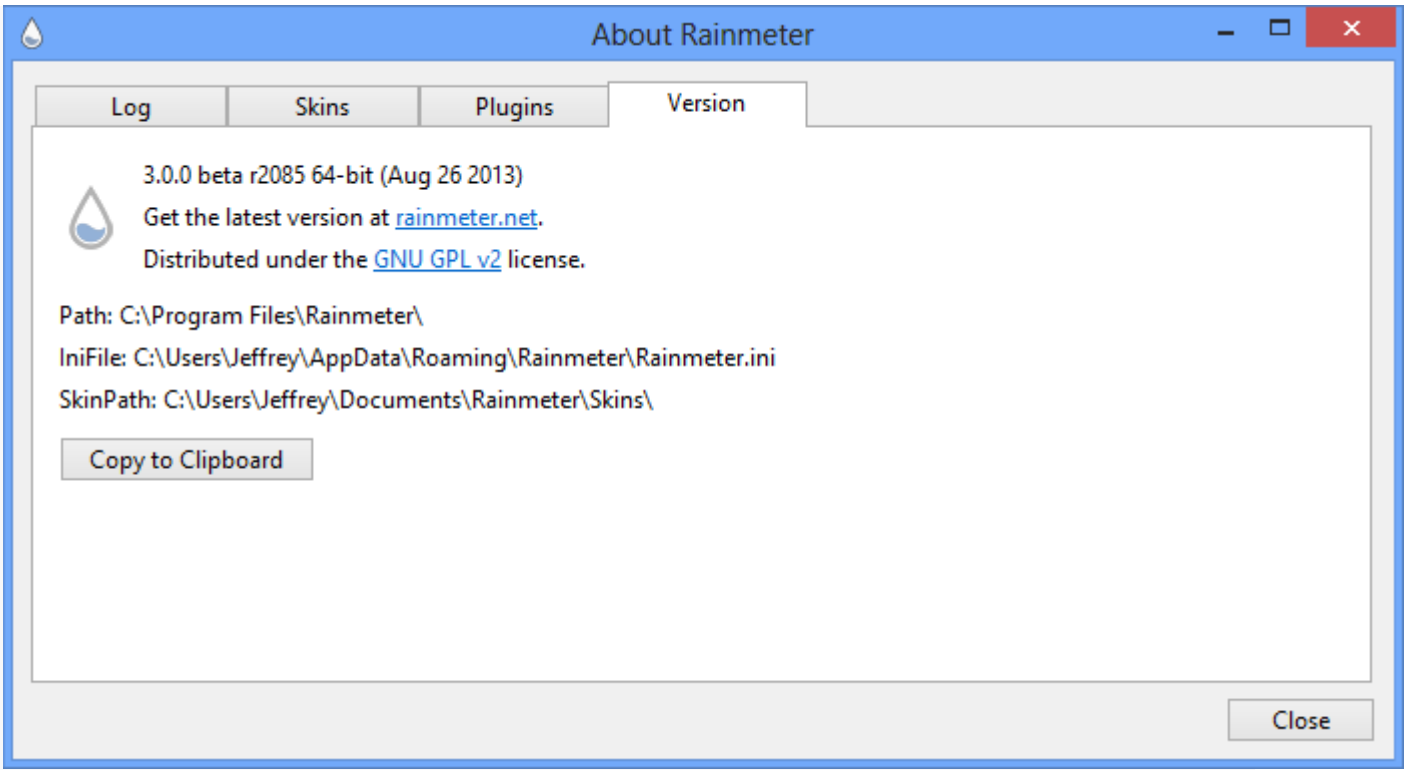


Information about the built-in and 3rd-party plugins installed in Rainmeter.

- **Name**: The name of the plugin .dll file.
- **Version**: The version number of the plugin .dll file.
- **Author**: The author name and other copyright information about the plugin.

Note: The list contains all plugins installed in Rainmeter. It does not reflect whether a particular plugin is actually loaded or running in a skin.

The Version tab



Information about the installed version of Rainmeter.

- **Version information**: The version, revision number, architecture and build date.
- **Path information**: The paths to the program, settings and skins folders.

Click **Copy to Clipboard** to copy the version and path information to the Windows clipboard.

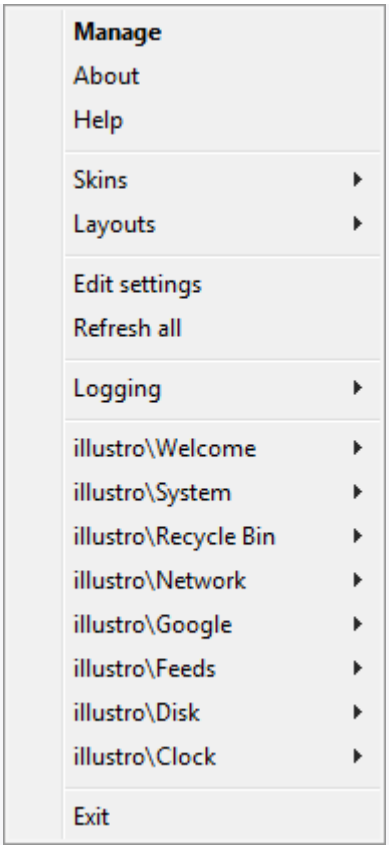
Context Menus [home]

In addition to the Manage window, the right-click context menus for Rainmeter can be used to control the application and skins.

There are two levels of context menus:

- **Rainmeter context menus**: Used to control the Rainmeter application and all skins. Accessed from the Rainmeter icon in the Windows Notification area.
- **Skin context menus**: Used to control an individual skin. Accessed by right-clicking any running skin on the desktop.

Rainmeter context menus



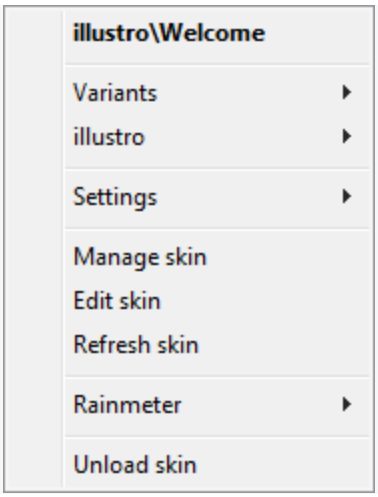
- **Manage**: Loads the Manage window.
- **About**: Loads the About window.
- **Help**: Opens the Rainmeter documentation site in the default web browser.
- **Skins**: Displays a list of all skins found when Rainmeter was started or refreshed. The list consists of the config folder for each skin, which expand to the skin .ini files in each. Clicking on a .ini file will load the skin in Rainmeter.

- **Layouts**: Displays a list of saved layouts. Clicking on one will load that layout in Rainmeter.
- **Edit settings**: Manually edit the settings in Rainmeter.ini with the text editor associated with .ini files.
- **Refresh all**: Refresh the entire Rainmeter application, including all currently active skins.
- **Logging**: Control Rainmeter logging.

- **ConfigNames**: (e.g. illustro\Welcome) Each currently loaded skin configwill be listed. Provides control of settings and actions for each using the same items as Skin context menus below.

- **Exit**: Exits Rainmeter.

Skin context menus



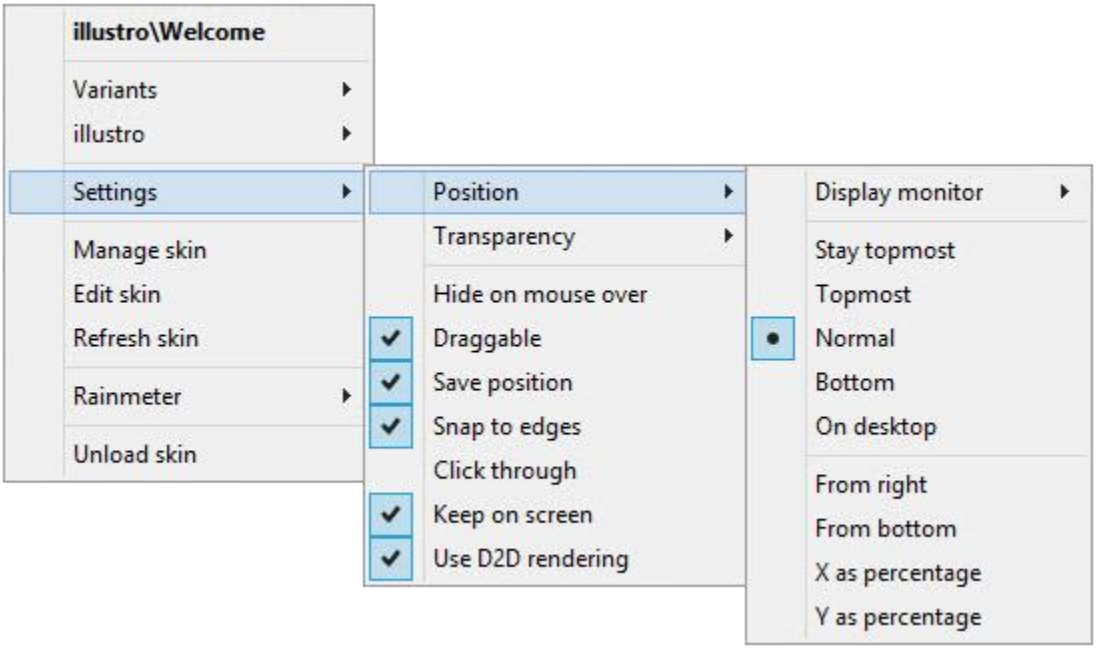
- **ConfigName**: (e.g. **illustro\Welcome**) Clicking the name of the config will open the folder in Windows Explorer.
- **Variants**: Lists the .ini files / variants in the skin config. Clicking on a .ini file will load the skin in Rainmeter.
- **Root Config**: (e.g. illustro) Lists the root-level config folders for the skin. The menu can be used to load skins under the root config.
- **Settings**: See Skins settings submenu below for details.
- **Manage skin**: Loads the Manage window with the skin selected.
- **Edit skin**: Edit the skin with the text editor associated with .ini files.
- **Refresh skin**: Refresh the skin.
- **Rainmeter**: Displays a subset of the main Rainmeter context menu.

- **Unload skin**: Unloads the skin.

Note for skin authors: Custom skin context menu items can be added using Context options in a skin. Details and examples here.

Skin settings submenu

Position



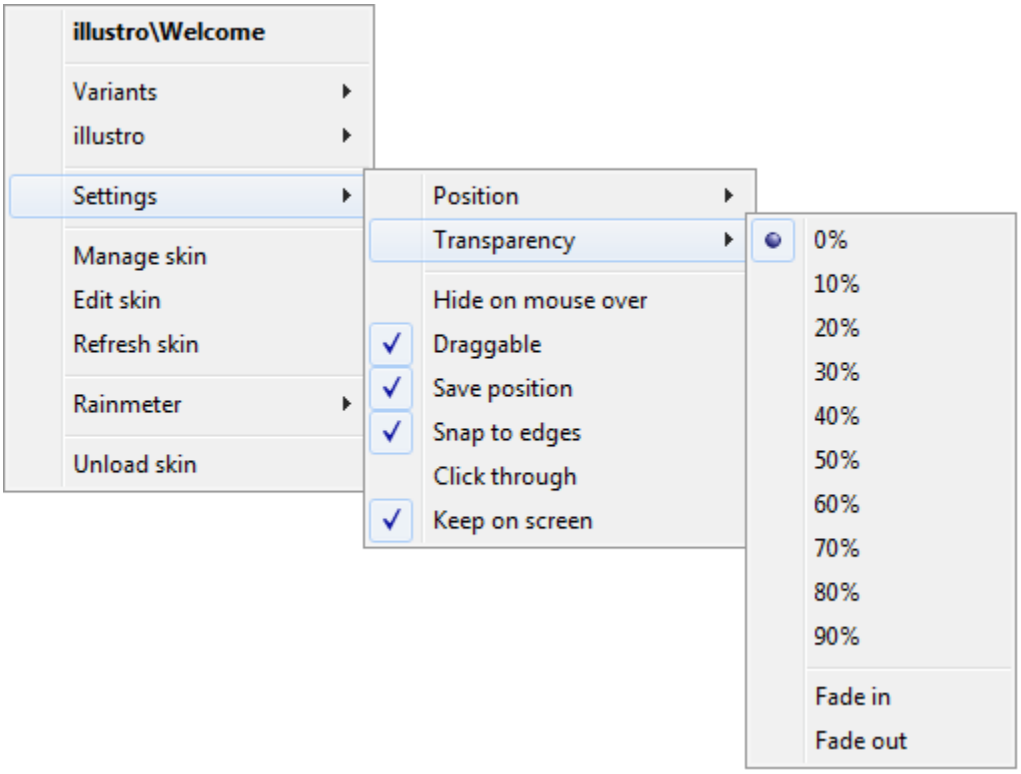
- **Display monitor**: Settings for the monitor on which the skin is displayed.

- **Stay topmost**: The skin will stay on top of all (including topmost) windows.
- **Topmost**: The skin will be on top of normal windows.
- **Normal**: The skin will stay visible when showing the desktop and will be brought to the top on click.
- **Bottom**: The skin will stay behind other windows.
- **On desktop**: Similar to Normal, except that the skin will stick to the desktop and cannot be brought to the foreground.

- **From right**: The WindowX of the skin will be expressed in pixels from the right edge of the screen.
- **From bottom**: The WindowY of the skin will be expressed in pixels from the bottom edge of the screen.
- **X as percentage**: The WindowX of the skin will be expressed as a percentage of the screen width.
- **Y as percentage**: The WindowY of the skin will be expressed as a percentage of the screen height.

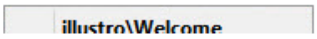
Note: See settings - skin sections for details on these settings.

Transparency



- **Percentage**: (e.g. 0% - 90%) Amount of transparency for the entire skin, expressed as a percentage.
- **Fade in**: On mouse over, the skin will fade from the amount of transparency set as a percentage above to opaque.
- **Fade out**: On mouse over, the skin will fade from opaque to the amount of transparency set as a percentage above.

General settings



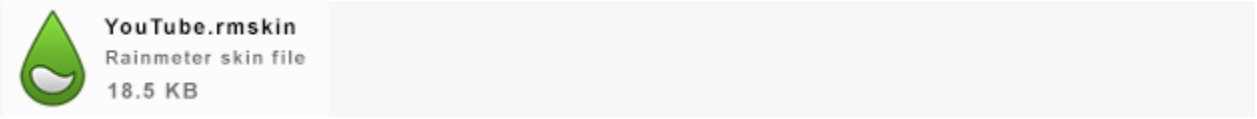
- **Hide on mouse over**: The skin will hide (fully transparent, ignores mouse clicks) on mouse over.
- **Draggable**: The draggable setting for the skin.
- **Save position**: The save position setting for the skin.
- **Snap to edges**: The snap to edges setting for the skin.
- **Click through**: The click through setting for the skin.
- **Keep on screen**: The keep on screen setting for the skin.
- **Use D2D rendering**: The UseD2D setting for the skin.

Installing Skins [home]

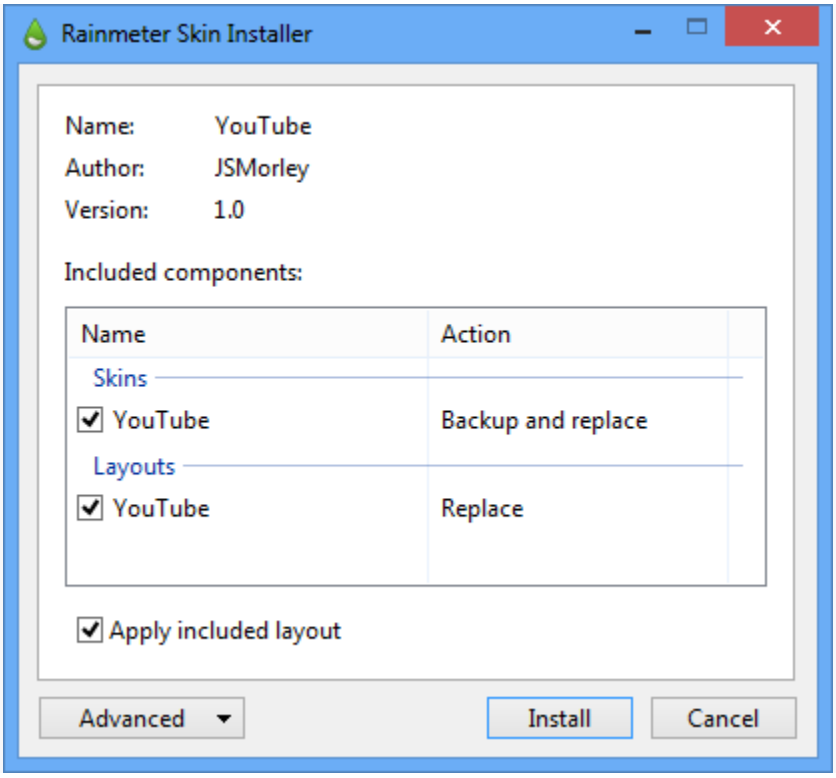
There are two ways of installing Rainmeter skins downloaded from the internet:

- **Automatically** : If the skin is in the .rmskin format.
  In short : Double-click the .rmskin file, click Install.
- **Manually** : If the file is a .zip/.rar/.7z archive.
  In short : Unzip the archive to the Rainmeter Skins folder. Refresh Rainmeter.

Installing Automatically



Skins in the Rainmeter Skin Packager (.rmskin) format can be easily and automatically installed with the **Rainmeter Skin Installer**. During a normal Rainmeter installation, the .rmskin extension is associated in Windows with the Skin Installer program, and simply double-clicking the file will install it in Rainmeter.



This dialog lists each component that will be installed. These may include:

- **Skins** : At least one skin will always be included and installed in the Rainmeter\Skins folder. The author of the package may indicate that one or more skins will automatically be loaded when the installation is complete.
- **Layouts** : If the author has added a layout to the package it will be installed in the Rainmeter\Layouts folder. If**Apply included layout** is selected, the layout will be applied to Rainmeter following installation.
- **Plugins** : If the author has added custom plugins to the package the appropriate 32bit/64bit architecture version of the plugin .dll files will be installed to the Rainmeter\Plugins folder.

Legacy .rmskin format components. No longer supported in new Rainmeter 2.4 .rmskin files:

- **Fonts** : If an author has included font files with the package, they will be installed in the Windows\Fonts folder. This may be disabled by unchecking **Install fonts to system** in the **Advanced** pull-down menu.
- **Addons** : If an author has included addon executable files with the package, they will be installed in the Rainmeter\Addons folder.

If any of the skins to be installed already exist, they will be moved to a **Backup** folder before installation. This may be disabled by unchecking **Backup skins** in the **Advanced** pull-down menu.

Click **Install** to complete the installation of the package.

_____

Note: If Rainmeter is being run as a portable installation, start **SkinInstaller.exe** from the Rainmeter program folder, and browse to the .rmskin file to install it.

Installing Manually

Before Rainmeter 1.3, most skins had to be unzipped and moved to the Skins folder by hand. All versions of Rainmeter are 100% backwards-compatible, so these older skins will still work just fine. Here is how to install them:

First of all, if a skin comes in an archive, such as a **ZIP**, **RAR** or **7Z** file, software is needed to "extract" them. 7-Zipis one of the popular choices, since it can handle almost any archive type. After installing the software, right-click the archive in Explorer to extract it.

The archive may include a **readme.txt** file with further instructions. If not, look for a folder with the same name as the skin - it might be inside another folder that is actually *called* "Skins". Move the folder to the Rainmeter "Skins" folder:

Windows 8/7/Vista: *C:\Users\YourName\Documents\Rainmeter\Skins* [1]
Windows XP: *C:\Documents and Settings\YourName\My Documents\Rainmeter\Skins* [1]

1. "YourName" is an example.

Finally, right-click the Rainmeter Windows Notification area icon and select **Refresh all**. The new skins will now be available to load from the Manage window or context menus.

Publishing Skins (.rmskin) [home]

Rainmeter includes tools that make it easier to share skins that you have created with other users. This page describes how to use the Skin Packager feature, and offers some guidelines for publishing a new skin in a public gallery, such as deviantArt or Customize.org.

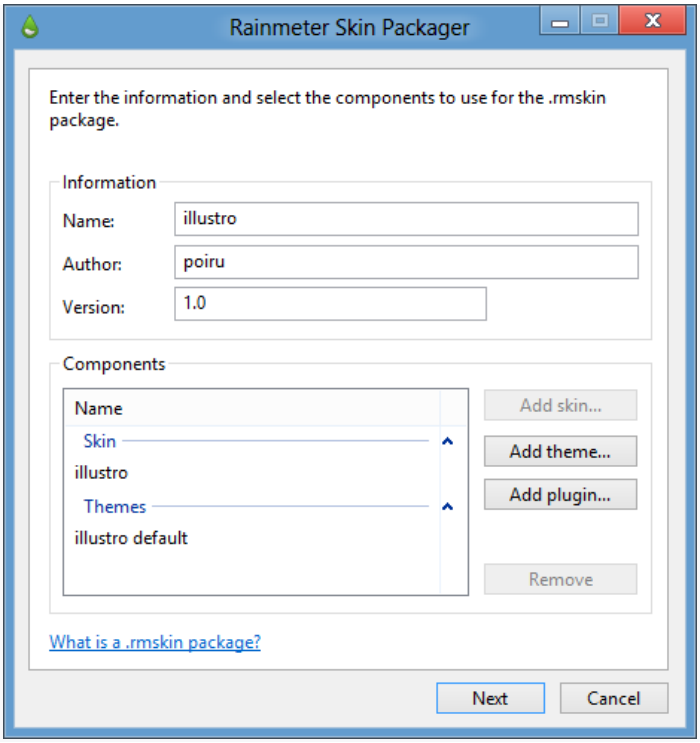
Skin Packager

The Skin Packager is used to create a "package" file that contains all of the files needed to install a skin in Rainmeter.

The package file is created in the *"Rainmeter Skin Installer"* format, with the extension **.rmskin**. The file is actually a modified zip file, with the addition of certain metadata that is used to verify the integrity of the package. (As of version 2.3, Rainmeter will not install a normal ZIP file changed to the ".rmskin" extension; the package must be created with the Skin Packager.)

To launch the Skin Packager, open the Manage window and click the button labeled **Create .rmskin package...** in the bottom-left.

Step 1: Information and Components



Start by providing some basic information about the skin:

- **Name:**
  The name of the skin. (Required.)

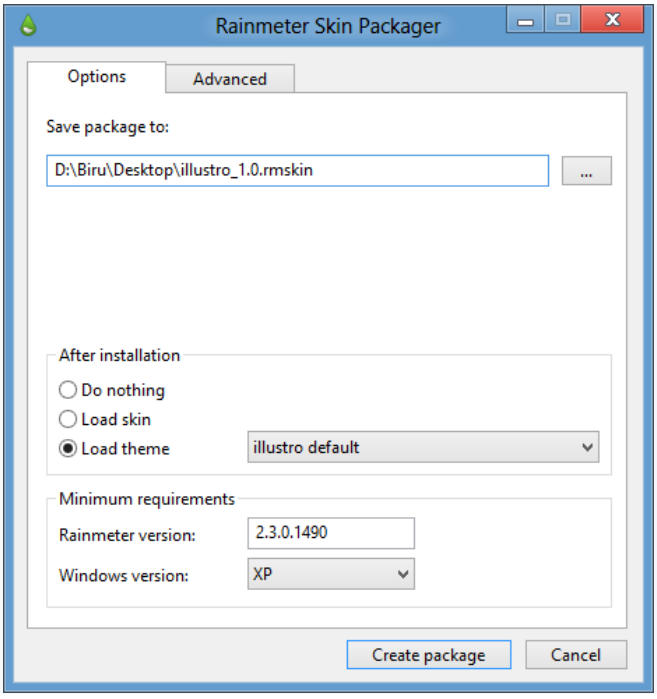
- **Author:**
  The name of the skin's author. (Required.)
- **Version:**
  The current version of the skin. This can be a number (such as "1.0"), a date ("15 November 2008"), or any other identifying string. (Optional.)

Next, choose the files that will be added to the package:

- **Add skin...**
  Choose the root config folder of your skin. This is the folder that will be copied to the user's Skins directory when the skin is installed. It must be a single folder that contains the entire skin or skins, including any resource files needed, such as addon utilities or fonts. You may choose a folder from your own Skins directory, or somewhere else on your computer. Only one root config folder is allowed in a single package. (Required.)
- **Add layout...**
  You may choose any number of layouts to be installed along with the skin.
- **Add plugin...**
  You may choose any number of plugins to be installed along with the skin. Only custom plugins should be distributed in this way. The standard plugins described in this manual are included with all versions of Rainmeter, and never need to be installed separately. To include a custom plugin, you must provide both the 32-bit and 64-bit versions of the plugin to ensure compatibility on all systems.

Click **Next** to continue.

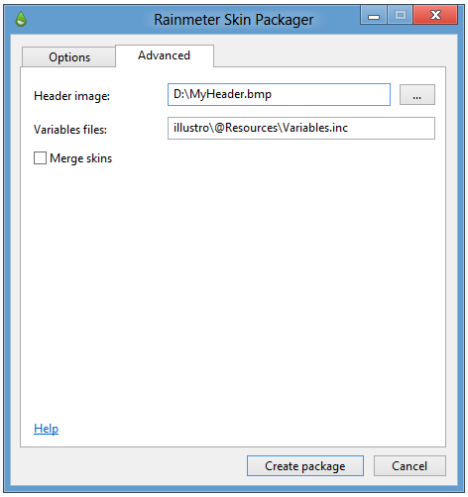
Step 2: Options



All of the following fields are optional. The default settings are appropriate for most skins.

- **Save package to:**
  Choose where the new package file will be created. Defaults to the current user's Desktop folder.
- **After installation**
  You may choose one of two actions for Rainmeter to take after installing the skin. (Before installing, the user will have the option to skip this action.)
  - **Load skin:**
    You may choose one of the skin ".ini" files in your package. This skin will be automatically loaded alongside any other skins that the user is already running. This setting is recommended for most skins.
  - **Load layout:**
    If you added any layouts to the package, you may choose one layout to be loaded automatically, replacing the user's existing layout.
- **Minimum requirements**
  You may set the minimum versions of Rainmeter and/or Windows required for the skin. Rainmeter will not install the skin on a system that does not meet these requirements.
  - **Rainmeter version:**
    Defaults to the current version of Rainmeter on your system. If you are sure that the skin will work on an earlier version, you may change the minimum requirement accordingly. Rainmeter version numbers are written in the form *Major.Minor.Patch.Revision*. For example, Rainmeter 2.3.3 (r1468) would be written as `2.3.3.1468`.
  - **Windows version:**
    Defaults to *Windows XP*, which is the lowest version of Windows supported by Rainmeter. Some skins may require higher versions of Windows, depending on their plugins, addons, or other factors. For example, any skins that make use of the Win7Audio plugin will require Windows Vista or above.

Step 3: Advanced



For additional options, click the **Advanced** tab at the top of the window.

- **Header image:**
  You may choose a custom header image to be displayed in the Skin Installer. The image must be a bitmap image (.bmp) that is exactly 400x60 pixels in size.
- **Variables files:**
  Some skins have one or more include files to store variables that can be changed by the user. If a user is reinstalling or upgrading a skin that they already have in their library, they will most likely want to preserve their existing preferences. You may specify these files in your skin package, so that the existing variable values are used instead of the defaults in the package.

  Specify the file location starting with the root config folder, e.g. `illustro\Clock\Variables.inc`. You may specify multiple files by separating them with pipes (`|`), e.g. `illustro\Clock\Variables.inc | illustro\Feeds\Variables.inc`.
- **Merge skins**
  If selected, the Skin Installer will not remove any existing files found in the user's Skins directory. (Normally, the root config folder is removed and replaced with the version in the skin package.) This can be used to install an "expansion" or a "patch" to a previously-installed skin, without duplicating unchanged files.

Finally, click **Create package** to complete the process. It may take several seconds to create the package, depending on the size of your skin. The file will be saved in your chosen location as *Name_Version.rmskin* (using the "Name" and "Version" fields that you entered in the first step).

Hidden Files

The Skin Packager will ignore any hidden files or folders in your root config folder. You can take advantage of this behavior by storing development files (such as Photoshop "PSD" files, or backup copies of skin code) as hidden files, so that they are excluded from the final product.

Publishing Guidelines

When distributing a skin to a small or private group of users, strict adherence to standards probably isn't too important. However, you may want to publish your finished work in a public repository, such as the Rainmeter skin galleries on deviantArt or Customize.org, for other users to discover, download and share. In this case, taking the time to follow a few simple guidelines will be helpful to you, your users, and the Rainmeter community at large.

Use the Skin Packager.

Most sites that allow users to publish their Rainmeter skins now require the skins to be uploaded in the official ".rmskin" format created by Rainmeter's built-in Skin Packager. This is not only more convenient for regular users, since a skin package can be installed automatically in just a few clicks—it is also safer. Rainmeter always checks the validity of a skin package before installing it, to ensure that the package was created by the Skin Packager and not modified or tampered with in any way.

As with any open-source software ecosystem, malware disguised as Rainmeter skins has been an occasional concern for the community. By using the official format, you can help us reduce this concern by promoting a more secure method of distribution.

Use metadata.

Many skin authors tend to ignore the [Metadata] section in their skins, because it has no effect on a skin's function or performance. But there are good reasons to make sure your skin is fully-tagged before you release it into the wild. A skin with complete metadata is easier to find in the Manage window, and offers a reliable, built-in way to provide users with setup and usage instructions, or at least direct them to a webpage or readme file with more information.

In addition, because a skin's code is often copied or excerpted in posts and comments across the Internet, good metadata makes it easier to trace the code back to its source. The Version tag is especially helpful to avoid confusing multiple revisions of the same skin that have been released at different times.

Choose the right license for your skin (and respect others').

The Rainmeter community is built around a free, open exchange of ideas and resources. We wholeheartedly encourage skin authors to release their works under terms that allow other users to borrow their code and concepts, modify them, and release them in a new, unique form. We believe that this kind of frictionless collaboration is a big part of what has made the Rainmeter ecosystem thrive.

There are a number of open-source software licenses available, and it couldn't be easier to add them to your Rainmeter skin: just paste the name into the License tag in your [Metadata] section. The following licenses are commonly used for Rainmeter skins, plugins and addons:

- Creative Commons
- GNU Public License
- MIT License

That said, the same rule applies to Rainmeter skins as it does to any other content on the Internet: if you are not explicitly given permission to use it, *you don't have permission*. You really need some indication of consent from the creator - even a quick email is enough. In addition, it's always a good idea to credit the original creators when you post your own skin, even if their license doesn't require it. This means not only the original author of some code that you have modified, but also any images, addons, plugins and fonts that are not your own.

Make sure your license and credits are included both on your download page (for example, in the deviation comments on deviantArt) and somewhere in your skin package (either in your metadata tags, or a separate "readme" file).

Clean up your layouts.

The ability to include a custom layout with your skin can make for a great presentation, especially in a large "suite", where it is helpful to provide a template or starting point for your users to get started on customizing their desktops instead of loading skins one by one.

While arranging a layout on your own desktop, and using the Manage window to save it to a file, there are some steps you can take to make your layout load smoothly and cleanly on other users' desktops:

- Remove [Skin] sections that don't belong to your suite. An easy way to keep only the sections you need is to check the "Exclude unloaded skins" option when saving the layout.
- Remove options from your skin sections if they are unchanged from Rainmeter's default settings, such asDraggable or SnapEdges. This will both reduce the size of your layout file, and make it easier to keep track of your code if you need to edit any layout settings by hand.
- Use the "From right" and "From bottom" options to make sure your skins are loaded in the correct position regardless of the user's screen resolution. If you need to determine the skin's position in a more complex way, you can edit the "Rainmeter.ini" settings file for your layout and write your own formulas for the WindowX, WindowY, AnchorX and AnchorY options for each skin section, using built-in variables. Here are some basic formulas you can use:
  - Center a skin on the desktop:
    `WindowX=(#WORKAREAX# + (#WORKAREAWIDTH# / 2 ) - (SkinWidth / 2) + Offset)`[1]
    `WindowY=(#WORKAREAY# + (#WORKAREAHEIGHT# / 2 ) - (SkinHeight / 2) + Offset)`
  - Center a skin on the screen:
    `WindowX=((#SCREENAREAWIDTH# / 2) - (SkinWidth / 2) + Offset)`
    `WindowY=((#SCREENAREAHEIGHT# / 2) - (SkinHeight / 2) + Offset)`
  - Align relative to the left edge of the desktop:
    `WindowX=(#WORKAREAX# + Offset)`
  - Align relative to the right edge of the desktop:
    `WindowX=(Offset + SkinWidth)R`
    or
    `WindowX=(#WORKAREAX# + #WORKAREAWIDTH# - SkinWidth - Offset)`
  - Align relative to the top edge of the desktop:
    `WindowY=(#WORKAREAY# + Offset)`
  - Align relative to the bottom edge of the desktop:
    `WindowY=(Offset + SkinHeight)B`
    or
    `WindowY=(#WORKAREAY# + #WORKAREAHEIGHT# - SkinHeight - Offset)`

1. "Offset" should be a number, e.g. `25R`, `-10B`, `(#WORKAREAX# + 75)`. If you don't need an offset (the skin is exactly centered or adjacent to the edge of the screen), leave it out. "SkinWidth" and "SkinHeight" should be the probable width and height of the skin.

Settings [home]

This page details the files and folders in which Rainmeter-wide and skin-specific **settings** are saved. These include each skin's position, transparency, draggability, active or inactive state, "snapping" behavior, and other general options. Normally, the user does not need to edit these files directly. Most settings can be changed using theManager, the context menu, or bangs.

Note: Many skins include features that are described as "settings," "options," "preferences," etc. that allow the user to customize the appearance and behavior of the skin. However, these features are different from the "settings" described on this page. They are handled entirely by the skin, usually as variables, and are not managed by Rainmeter at all. Options such as fonts, colors, passwords, RSS feeds and "weather codes" are in this category. These are not "standard" features, and are not supported in any formal way.

Files and Folders

All Rainmeter settings files are stored in Rainmeter's application data folder. If Rainmeter has been installed normally, the default location of the folder is:

- `C:\Documents and Settings\YourName\Application Data\Rainmeter`[1] (for Windows XP).
- `C:\Users\YourName\AppData\Roaming\Rainmeter`[1] (for Windows Vista, 7 and 8).

If Rainmeter is installed as a portable application, settings files are found in the installation folder.

1. "YourName" is an example.

Rainmeter.ini

Most settings are saved in a file called **Rainmeter.ini**. The file is written as a standard INI file, which uses the following[1] format:

```
[Section]

Key=Value
```

Select all

Each key is an option that modifies the property identified in the section name. The following sections are used in Rainmeter.ini:

- [Rainmeter]
  Defines global Rainmeter options.
- [Skin] sections
  Defines options related to individual skins. Each skin is identified by its config name.
- [TrayMeasure] section
  Defines options that change the appearance of Rainmeter's tray icon.

When settings are changed in Rainmeter, they are saved to Rainmeter.ini automatically. If settings are changed in Rainmeter.ini, a refresh application command is needed to apply the new settings.

1. "Section," "Key" and "Value" are examples.

Other Files

Rainmeter.data

This file is used by Rainmeter's program and some plugins to store "global" settings that are not related to individual skins. These are system- or user-specific settings that are not saved with layouts.

Rainmeter.stats

This file records Windows' network usage statistics over time. These statistics are used by the Net measures to display "cumulative" network usage data. The contents of Rainmeter.stats may be cleared using the !ResetStats bang.

Layouts

A **layout** (formerly **theme**) is a saved "session" or "state". The user can create a layout using the Manager, which copies the settings in Rainmeter.ini to the layout file. These settings can later be restored using the Manager or the!LoadLayout bang. Layouts may also be installed with 3rd-party skins.

Global options under [Rainmeter] are not replaced when a layout is loaded, preserving local settings such as:

- ConfigEditor
- SkinPath
- DisableVersionCheck
- Language

Plugins

Custom plugins are installed to this folder. Plugins in this folder can be used in the same manner as the standard plugins included with Rainmeter.

Addons

3rd-party "addon" utilities, such as RainRGB, can be installed to this folder. This is a legacy feature. As of Rainmeter 2.3, each skin or suite should include its own addons in the @Resources folder.

[Rainmeter] section [home]

The [Rainmeter] section of Rainmeter.ini defines global options.

Options

SkinPath

> Path to the skins folder.

ConfigEditorDefault: Notepad

> Defines the application that is used to edit the Rainmeter's configuration files when "Edit Skin" is chosen from the context menu. Relative paths can be used here, which may be useful for portable installs.

TrayExecuteM, **TrayExecuteR**, **TrayExecuteDM**, **TrayExecuteDR**

> Action to be executed when the **m**iddle or **r**ight clicking (or **d**ouble-clicking) on the notification area (tray) icon.
>
> **Note:** `TrayExecuteR` will override the context menu that normally appears (hold CTRL and right-click to force context menu).

TrayIconDefault: 1

> Set to `0` to remove the notification area (tray) icon.

DesktopWorkArea

> Defines the area for maximized windows.
>
> With multiple monitors, use `DesktopWorkArea@N` (where *N* is the number of the monitor) to set the work area of a specifc monitor. `DesktopWorkArea` and `DesktopWorkArea@0` both represent the primary monitor.
>
> **Note:** Moving the taskbar will reset the workarea to Windows' default, as will changing screen resolution.

DesktopWorkAreaTypeDefault: 0

If set to `1`, `DesktopWorkArea` will be define margins relative to the edges of the screen.

Example: With `DesktopWorkAreaType=0` and `DesktopWorkArea=10,20,600,500`, maximized windows will use an area of 600x500 pixels, which is located 10 pixels from the left and 20 pixels from the top of the screen. With `DesktopWorkAreaType=1` and `DesktopWorkArea=10,20,30,40`, the area for maximized windows will be 10 pixels from the left, 20 pixels from the top, 30 pixels from the right, and 40 pixels from the bottom of the screen.

`Logging`Default: `0`

If set to `1`, the log will be saved to a file.

`Debug`Default: `0`

If set to `1`, logging will be more verbose. Use `Debug=1` with caution as it degrades performance.

`DisableVersionCheck`Default: `0`

If set to `1`, checking for new updates will be disabled.

`DisableDragging`Default: `0`

If set to `1`, sets the draggable state of **all** active skins to prevent dragging skins with the mouse.

[TrayMeasure] section [home]

The [TrayMeasure] section defines a measure for the notification area icon. To use the default Rainmeter icon, remove the [TrayMeasure] section completely.

Options

`Measure`

Type of the measure to be shown as the tray icon. The measure must return values between some limits.

`TrayMeter`Default: `Histogram`

The manner in which the measure is depicted. This can be either `Histogram` or `Bitmap`.

`TrayColor1`Default: `0,100,0`

Background color for `TrayMeter=Histogram`.

`TrayColor2`Default: `0,255,0`

Foreground color for `TrayMeter=Histogram`.

`TrayBitmap`

Path to the tray icon bitmap file for `TrayMeter=Bitmap`. Defines the name of the bitmap used when Bitmap is chosen for TrayMeter. The bitmap can have any number of 16x16 frames.

It is possible to use separate icon files as the tray bitmap. `%i` can be used in the filename to define an increasing number from 1. E.g. `TrayBitmap=tray-%i.ico` would read the icons files tray-1.ico, tray-2.ico, tray-3.ico and so on as long as it can find them. The format is like printf. (e.g. `TrayBitmap=tray-%02i.ico` will load tray-01.ico, tray-02.ico, ...)

A simple `Measure` that returns a single value, combined with a `TrayBitmap` path and name of a single 16x16 .ico file can be used to replace the default tray icon with a custom one.

Note: The path to a bitmap image or .ico files is a relative path from the main Skins\ folder (i.e.`TrayBitmap=Icons\MyIcon.ico`.)

Examples

Show total network activity as a histogram.

```
[TrayMeasure]
Measure=NetTotal
Interface=0
TrayMeter=Histogram
```

Select all

Replace the default tray icon with a custom one.

```
[TrayMeasure]
Measure=Calc
Formula=1
TrayMeter=Bitmap
TrayBitmap=Icons\MyIcon.ico
```

Select all

Skin sections [home]

Skin sections in Rainmeter.ini (e.g. `[illustro\Clock]`) define options for how the skin is loaded and displayed in Rainmeter.

The skin is defined in Rainmeter.ini using the Config name.

Options

`Active`

The current active state of the config. If Active is `0`, the config is inactive. If a number other than 0, the active skin .ini or variant is the file in alphabetical order in the config folder.

`WindowX`, **`WindowY`**

The X and Y position of the skin. A trailing `%` sign indicates that the value is a percentage of the screen width. A trailing `R` will make the position relative to the right edge of the screen instead of the left. A trailing `B` will make the position relative to the bottom edge of the screen instead of the top.

The `%` and `R` / `B` modifiers can be used together.

By default the position is relative to the primary screen. You can override this with `@N` where *N* is 0 to 32 and denotes which screen to position the skin on (1-32) or the virtual desktop (0). The screen selection will apply to both WindowX and WindowY if the value is set on only one of them.

`AnchorX`, **`AnchorY`**

By default WindowX and WindowY control the position of the upper left corner of the config window. AnchorX and AnchorY allow that anchor position to be changed. The Anchor can be defined in pixels from the upper left corner of the window or as a percentage of the config if % is used. If a letter R or B is added to AnchorX or AnchorY, then the position is relative to the right or bottom edge of the window.

As an example, by setting WindowX, WindowY, AnchorX and AnchorY all to 50% the config will be truly centered in the primary monitor regardless of screen resolution or aspect ratio.

`SavePosition`Default: `1`

If set to `1`, changes to the window position will be saved to Rainmeter.ini.

`AlwaysOnTop`Default: `0`

Defines the Z-position of the skin. Valid values are:

- `2`: Always on top. The skin will be on top of all (including topmost) windows.
- `1`: On top. The skin will be on top of normal windows.

- `0`: Normal. The skin will stay visible when showing the desktop and will be brought to the foreground on click.
- `-1`: Bottom. The skin will stay behind other windows.
- `-2`: On desktop. Similar to `0` except that the skin will stick to the desktop and cannot be brought to the foreground.

`Draggable` Default: `1`

If set to `1`, the skin can be dragged around with the mouse.

`SnapEdges` Default: `1`

If set to `1`, the skin will snap onto screen edges and other skins when moved. To disable snapping temporarily, hold the CTRL key down while moving the skin.

`HideOnMouseOver` Default: `0`

If set to `1`, the skin will disappear when the cursor is over it and reappear when the cursor is moved away. To temporarily disable this behaviour, hold the CTRL down when moving the cursor.

`StartHidden` Default: `0`

If set to `1`, the skin will start hidden. The !Show bang must be used to show the skin.

`AlphaValue` Default: `255`

Sets the transparency of the skin. Valid values range from `0` (fully transparent) to `255` (solid).

`FadeDuration` Default: `250`

Defines the length of the fade duration in milliseconds when activating/deactivating the skin or when the fade bangs are used.

`ClickThrough` Default: `0`

If set to `1`, mouse clicks will pass through the skin. To temporary disable this, hold the CTRL key when clicking.

`KeepOnScreen` Default: `1`

If set to `1`, the skin will be kept within the bounds of the screen.

`UseD2D` Default: `1`

Used to override the global UseD2D rendering setting for an individual skin.

If set to `1` (default), UseD2D is enabled for this skin. If set to `0`, UseD2D is disabled for this skin, and the older GDI+ rendering engine will be used.

Note: This selection will be disabled if the system does not support Direct2D.

`LoadOrder` Default: `0`

Determines the order in which the skins are loaded. The value can be any number (even negative). Skins with the lowest load order are loaded first.

Example: Three skins having `LoadOrder=-1`, `LoadOrder=2` and `LoadOrder=5` would load the configs in that order, with the config with `LoadOrder=-1` appearing beneath the one with LoadOrder=2 which is in turn beneath the config containing `LoadOrder=5`. If two configs have the same value for LoadOrder, they are then loaded in alphabetical order.

Note: The value of LoadOrder has no bearing on the Z-position of the skin. LoadOrder only affects how skins in the same Z-position interact with each other. That is to say, configs set to "Topmost" will always appear above configs set to "Normal", but two configs in "Topmost" will layer themselves according to their LoadOrder value.

`AutoSelectScreen` Default: `0`

If set to `1`, the WindowX/WindowY "@*N*" settings are made automatically based on the position of the meter's window. This setting will reset to 0 when a monitor is selected in the Manage window or on the skins context menu.

Skins [home]

Rainmeter's basic purpose is to run skins. A **skin** is a movable, dynamic, sometimes-interactive window that appears over the Windows desktop, and usually gathers and displays information of some kind.

Skins come with many different sizes, styles, user interfaces, and levels of complexity. Rainmeter includes utilities for publishing and installing 3rd-party skins. Each skin is generally stored and loaded as an independent, self-contained module.

Files and Folders

All skins are stored in Rainmeter's "Skins" folder. If Rainmeter has been installed normally, the default location of the Skins folder is:

- `C:\Documents and Settings\YourName\My Documents\Rainmeter\Skins`[1] (for Windows XP).
- `C:\Users\YourName\Documents\Rainmeter\Skins`[1] (for Windows Vista, 7 and 8).

If Rainmeter is installed as a portable application, the Skins folder is found in the installation folder. The Skins folder can also be changed in Rainmeter's settings.

A skin is stored as a file called "*SkinName*.ini" in its own folder within Skins, as in:`Rainmeter\Skins\SkinName\SkinName.ini`.[1] (The file and folder names do not need to match.)

All of these folder paths may be referenced in a skin as built-in-variables.

1. "YourName" and "SkinName" are examples.

Variants

If one folder contains multiple `SkinName.ini`[1] files, they are each considered **variants** of the same skin. This means that only one variant can be active at a time, and all variants use the same settings. Separate skins must be stored in separate folders.

1. "YourName" and "SkinName" are examples.

@Resources

Any supporting files used by the skin, such as images, addon executables, sounds, or Lua scripts, should also be stored in the skin folder. Complex skins are recommended to store supporting files in a folder called @Resourcesinside the root config folder. The @Resources folder is ignored by Rainmeter, except to load custom fonts andcustom cursors.

Config

Each skin is identified with a certain **config name**. The config name is based on the path from the main Skins folder to the skin file. For example, if a skin is located at...

`C:\Users\YourName\Documents\Rainmeter\Skins\Foo\Bar\SkinName.ini`

...then the config name would be:

`Foo\Bar`

Because skins may have any number of variants, many Rainmeter features—especially bangs—refer to a specific skin by its config name, rather than the skin's filename.

Root config

Skins may also be stored in subfolders within the same **root** config folder. They share the same @Resourcesfolder, and are packaged and installed together. Otherwise, Rainmeter treats each subfolder as a separate config.

For example, the following skins belong to the "illustro" suite:

`C:\Users\YourName\Documents\Rainmeter\Skins\illustro\Clock\Clock.ini`
`C:\Users\YourName\Documents\Rainmeter\Skins\illustro\Network\Network.ini`

This means that they are both located in the same root config folder, "illustro," in the main Skins directory.

`C:\Users\YourName\Documents\Rainmeter\Skins\illustro\`

Each skin has a unique config name:

`illustro\Clock`
`illustro\Network`

But since they belong to the same root config, they share the same @Resources folder:

`C:\Users\YourName\Documents\Rainmeter\Skins\illustro\@Resources\`

A **suite** is an informal term for skins that are organized together in this manner. Suites are often used by skin authors to make a set of skins that share a common style, or complement each other in some way. If a root config folder contains only one `SkinName.ini` file, then the config and root config are the same, and the skin can be stored, packaged and installed by itself.

Format

The `SkinName.ini` file is written as a standard INI file, which uses the following[1] format:

```
[Section]

Key=Value
```

Select all

Each property of the skin is defined by a section. Each key is an option that modifies that property. The following sections may be used in skins:

- [Rainmeter]
  Defines options that affect the entire skin.
- [Variables]
  Defines text strings that can be used throughout the skin.

- Measures
  Objects that retrieve (or "measure") information of some kind.
- Meters
  Objects that display information and other visual elements.
- MeterStyles
  Define options that may be used by several meters.
- [Metadata]
  Defines the name, version, license, and other non-functional information about the skin.

A skin must have at least one meter. All other properties are optional, and may not even be present, depending on what is needed for the skin. For skins that are publicly distributed, the [Metadata] section is strongly recommended.

There are only a few rules about using the INI format:

- All section names in a skin must be unique.
- All option names within a section must be unique.
- Section and option names should include alphanumeric characters only (no spaces or punctuation).
- Option values may be contained by quotes (`"`). If so, Rainmeter will ignore the containing quotes.
- Option values must be kept on a single line.

1. "Section," "Key" and "Value" are examples.

Update

When a skin is loaded, it **updates** on a regular cycle. The length of time between updates is defined by the Updateoption in the [Rainmeter] section, and defaults to 1 second (or 1000 milliseconds). The update determines when the skin reacts to changes in the values of variables, measures and options.

Individual meters and measures can be made to update more slowly by "skipping" cycles, using the UpdateDivideroption. In addition, the !Update bang forces the skin to update immediately, resetting the timed cycle.

Refresh

When a skin **refreshes**, all values are reset, and the skin starts over as if it had just been loaded for the first time. Refreshing also applies any changes that have been made to the code in `SkinName.ini`.

A skin can be refreshed from the context menus, or using the !Refresh bang.

[Rainmeter] section [home]

The [Rainmeter] section of a skin defines options for the entire skin.

Note: The [Rainmeter] section does not support Dynamic Variables or changes using the !SetOption bang.

General Options

`Update`Default: `1000`

> Defines the update interval of the skin in milliseconds. On each update, meters and measures are updated.
>
> Using `-1` will update the skin only once on load (or on refresh). The skin can then be manually updated using the !Update bang.
>
> **Note:** Update is not related to the system clock, and if a computer is busy or a complicated skin takes longer than the interval to complete a full update of the skin, updates can be unreliable in relation to elapsed time. Do not use the Update value to drive clocks or other timing sensitive functions.

`AccurateText`Default: `0`

> If set to `1`, String meters are measured and rendered using improved padding and character spacing similar to that provided by Direct2D.

`DynamicWindowSize`Default: `0`

> If set to `1`, the window size is adjusted on each update to fit the meters.

`DragMargins`Default: `0,0,0,0`

> Uses 4 values separated by commas to define the area from where the window can be dragged. The values define a margin of non-draggable area. It's also possible to use negative numbers in which case the margin is calculated from the opposite side. E.g. `DragMargins=0,-100,0,0`.

`OnRefreshAction`

> Action to execute when the skin is loaded or refreshed.

`OnUpdateAction`

> Action to execute on each Update of the skin.

`OnCloseAction`

Action to execute when the skin is unloaded or when Rainmeter is closed.

`OnFocusAction`

Action to execute when the skin recieves focus in Windows (set by clicking the mouse on the skin).

`OnUnfocusAction`

Action to execute when the skin loses focus in Windows.

`OnWakeAction`

Action to execute when Windows returns from the sleep or hibernate states.

`TransitionUpdate`Default: `100`

Defines the update time in milliseconds for meter transitions. While a transition is active the meter will update at this rate. Currently, only the Bitmap meter supports meter transitions.

`ToolTipHidden`Default: `0`

If set to `1`, all tooltips in the skin will be hidden. More information on tooltips.

Background options

General image options

All general image options are valid for `Background` **except** `Tile`, which is handled with `BackgroundMode`.

`Background`

Path of a background image file.

`BackgroundMode`Default: `1`

Defines the background mode for the skin. Valid values are:

- `0` : Image as defined by `Background`

- `1` : Transparent background
- `2` : Fill with a solid color
- `3` : Fill by scaling image as defined by `Background`
- `4` : Fill by tiling image as defined by `Background`

`BackgroundMode=2` : The color is set by adding a SolidColor option.

`BackgroundMargins`Default: `0,0,0,0`

If `BackgroundMode=3`, defines margins of the `Background` image that are not scaled. The parameters are`left,top,right,bottom`.

Example: `BackgroundMargins=0,10,0,20`.
10 pixels from the top and 20 pixels from the bottom of the image are not scaled.

`SolidColor`, **`SolidColor2`**Default: `0,0,0,0`

This option will specify the background color when `BackgroundMode=2`. If `SolidColor2` is also specified, the background is a gradient composed of `SolidColor` and `SolidColor2`.

Hint: `SolidColor=0,0,0,1` can be used to make transparent areas of the background clickable.

`GradientAngle`

Angle of the gradient in degrees (for `SolidColor` and `SolidColor2`) when `BackgroundMode=2`.

`BevelType`Default: `0`

If enabled, draws a bevel around the edges of the entire skin when `BackgroundMode=2`. Valid values are:

- `0`: No bevel
- `1`: Raised
- `2`: Sunken

Context options

`ContextTitle`, **`ContextTitle2`**, **`ContextTitle3`**...

If not blank, adds an item to the skin's context menu under "Custom skin actions". Up to 15 *ContextTitleN*options are allowed, with up to 30 characters per option. Additional characters are truncated with an ellipsis (`...`). If more than 3 *ContextTitleN* options are given, "Custom skin actions" becomes a submenu. If a*ContextTitleN* option is not valid, all

subsequent *ContextTitleN* options are ignored. In addition, if more than 3 options are given, and the value of *ContextTitleN* includes only dashes (`-`), the item is displayed as a separator.

Note: Variables in `ContextTitleN` are always dynamic. Variable values are read at the time the context menu is opened.

`ContextAction`, **`ContextAction2`**, **`ContextAction3`**...

Action triggered by clicking the corresponding ContextTitleN item. *ContextActionN* is required for *ContextTitleN*to be valid, unless the item is a separator.

Note: Variables in `ContextActionN` are always dynamic. Variable values are read at the time the context menu item is clicked.

Aero Blur options

`Blur`Default: `0`

Set to 1 to enable Aero Blur on Windows Vista or Windows 7 operating systems. If no BlurRegions are specified, the entire skin background is blurred. Note that Windows 8 has removed this capability.

`BlurRegion, BlurRegion2, ..., BlurRegion`*`N`*

Defines areas and shapes of the one or more regions of the skin to be blurred. The format of the option is:`BlurRegion=Type, TopX, TopY, BottomX, BottomY ,`*`Radius`*.

Valid values for `Type` are:

- `0` : Region is disabled
- `1` : Rectangular region
- `2` : Rectangular region with rounded corners (requires `Radius`)
- `3` : Elliptical region

The parameters following `Type` define the size and shape of the region. They are, in order:

- `TopX` : Top left horizontal point in the skin
- `TopY` : Top left vertical point in the skin
- `BottomX` : Bottom right horizontal point in the skin
- `BottomY` : Bottom right vertical point in the skin
- `Radius` : Radius of the corners of rounded rectangles (required for `Type=2`)

Note: Aero Blur options can be dynamically controlled with several bangs.

Example

```
[Rainmeter]

Update=1000

Blur=1

BlurRegion=1,10,10,190,50

; BlurRegion creates a rectangle starting at 10 pixels from the left, 10 pixels down from the
top, and ending at 190 pixels to the right, 50 pixels from the top.


BlurRegion2=3,10,70,80,110

BlurRegion3=2,10,130,190,170,15


[Background]

Meter=Image

W=200

H=180

SolidColor=0,0,0,50

LeftMouseUpAction=[!AddBlur "1,0,0,200,80"]
```

Select all

Deprecated options

`Author`

> The author of the skin.
>
> **Note:** This option is deprecated. Author should be defined in the [Metadata] section of the skin.

`AppVersion`

> Defines the minimum version of Rainmeter required to use the skin. The formula to calculate the value is: `major * 1000000 + minor1 * 1000 + minor2`.
>
> **Note:** This option is deprecated. Use the version capabilities in Skin Packager.

`LocalFont`, **`LocalFont2`**, ...

Loads the specified TTF font files for use with FontFace in String meters.

Note: This option is deprecated. Use the @Resources\Fonts folder instead.

[Metadata] section [home]

The [Metadata] section of a skin describes the skin. The information is presented in the Manage window.

Options

Name

> The name of the skin.

Author

> The author of the skin.

Information

> A description of the skin, setup and usage instructions, credits, or other documentation elements. Use `|` for line breaks.

Version

> The version of the skin.

License

> The name of a standard license or explicit permissions and conditions for ports, mods and derivative works.

Example

```
[Metadata]

Name=MassToEnergy

Author=Albert Einstein

Information=Skin to calculate the energy potential of a given mass. | Edit the mass of the obj
ect in metric tons in the [Variables] MassOfObject | Do NOT change the SpeedOfLight variable.

Version=1.1905

License=Creative Commons Attribution-Non-Commercial-Share Alike 3.0
```

Select all

@Include option [home]

The `@Include` option loads the content of an external .ini at the position it is defined. The loaded file is treated as if the contents were included in the actual skin .ini file. A frequent use case is to have an include file with a [Variables] section in order to share variables between multiple skins.

Options

`@Include`, **`@Include2`**, **`@IncludeN`**

> Path to the INI formatted file to include. Is is recommended that include files use the .inc extension (rather than .ini) and are placed in the @Resources folder.
>
> The *N* in `@IncludeN` can also represent text. For example: `@Include2`, `@Include3`, `@IncludeVariables`, and `@IncludeMeters` are all valid.

Remarks

The statement may be placed in any section. When the skin is loaded, all new sections from the included file are inserted immediately after the section where the statement is placed. Rainmeter treats these sections - whether they're measures, meters, MeterStyles, etc. - exactly as if they had been written in the skin .ini itself, and appropriately determines things like layering, relative positions and referenced measure values.

You may also include files within files. Once again, the ordering is determined by placement: when any file includes another file, the new contents are added within its own sections, immediately after the section where the statement is made.

If there is a conflict - that is, if the same section exists in more than one file - Rainmeter will treat whichever one comes first in the ordering as the "real" section. Any options on the later instances will be added to the first one, and otherwise the later instances are simply ignored. If there are different values given for the same key, the last value is taken. Unlike new sections, options on pre-existing sections are added in their original order, so the calling section may overwrite values from the included file if they are placed below the @include statement.

See also: @Include Guide

Example

IncludeFile.inc:

```
[Variables]

Color=255,255,255,255
```

Select all

Skin.ini:

```
[Variables]

Font=Arial

@Include=IncludeFile.inc


[SomeMeter]

FontFace=#Font#

FontColor=#Color#
```

Select all

@Resources folder [home]

The @Resources folder in the root folder of a skin is the reccomended location to store and access images, fonts, sounds, include files, addons or other additional files used by the skin. The @Resources folder is ignored when scanning for skins, so using it to store images and other resources will improve the initial load time of Rainmeter.

The @Resources folder must be created at the root config level of the skin (e.g. `Skins\illustro\@Resources`) and the `#@#` built-in variable can be used as a shortcut to specify it.

Fonts

TrueType (.ttf) fonts in the `@Resources\Fonts` folder are automatically loaded and can be used with the FontFaceoption in string meters.

For example, to include the font SomeFont.ttf, it should be placed in:`Skins\SomeSkin\@Resources\Fonts\SomeFont.ttf`, and used with `FontFace=SomeFontFamily`.

Cursors

Custom cursors (.ani or .cur) in the `@Resources\Cursors` folder are automatically loaded and can be used withMouseActionCursorName.

Examples

```
ImageName=#@#Images\MyImage.png

LeftMouseUpAction=["#@#Addons\MyAddon.exe" "Parameter"]

@Include=#@#Variables.inc
```

Select all

Meters [home]

A **meter** is an object that defines a visual element that is displayed in a skin. Meters are one of the two major kinds of objects in a skin, along with measures.

Usage

A meter does not have a "value" in the way that measures do. Some meters are used to display or respond to informational values in two ways:

- A meter can be bound to a measure. In this way, the meter will automatically display the value in a way that is appropriate for the type of meter. For example, a string meter would display the string value of a measure as a block of text, while a bar meter would display the same measure's number value as a percentage of its maximum value. Some meter types must be bound to a specific measure; on others, binding is optional.
- A meter can use variables in any option. This includes section variables, which provide an alternative way of using measure values. Dynamic variables are allowed in all options on all meters (other than the `Meter`option).

Not all meters are used to display information. Some are used to create static elements, such as background images, frames and labels. Specific options and requirements for each meter type are detailed in their individual articles.

Format

A meter is written as a section in the skin. All meters use the `Meter` option to define the section as a specific type of meter. Most other meter options depend on the type, but there are some general options that are valid in many or all meters.

Below is an example of a complete working meter:

```
[MyMeter]

Meter=String

Text=Hello, world!
```

Select all

Positions

A meter has a certain **position**, which is given by its X and Y options, and **dimensions**, which are given by its W and H options. This means that every meter is actually bounded by a rectangular block of pixels, even though it may have a transparent background and therefore appear to be "free-floating."

Meters are positioned within the skin window. This means that when the skin is moved, its meters move with it. It also means that meter positions are given relative to the top-left corner of

the skin, rather than the desktop. For example, a meter with the option `X=15` starts 15 pixels from the left edge of the skin.

The dimensions of the **skin window** are determined by the positions and dimensions of all meters when the skin is loaded. If the skin has DynamicWindowSize enabled, the window will be "pushed" outward if meters are moved rightward or downward or expand in width or height. However, meters that move leftward beyond `X=0`, or upward beyond `Y=0`, will appear either "cut off" or completely invisible. There is no technical limit to the size of the skin window, but, practically speaking, skins should be made to fit within the current desktop work area.

Left: a skin with a single meter that appears to be "free-floating." Right: the same skin with colored backgrounds added, showing how the meter (purple) is actually positioned inside the skin window (green) and "pushes" the window boundaries to the lower-right.

Order

The order of meters in the skin code is important in two ways:

- **"Z" position.** Meters are drawn in order. Later meters appear "on top of" or "in front of" earlier meters. This means that, for example, a background image that appears behind all other meters must be the first meter that appears in the skin code.
- **Relative positions.** A meter's `X` or `Y` position may be set "relative" to the position of the previous meter. This is useful for groups of meters that follow a common pattern, such as lists, tabs or menu items.

MeterStyles

Meters can use option values from other meters. Using the MeterStyle option, one meter may **inherit** all options from one or more "parent" sections.

For more, see MeterStyles.

General Meter Options [home]

Options available for use with all meters.

Options

`Meter`

Type of the meter (e.g. `Bar` or `String`).

`MeterStyle`

Specifies one or more sections as MeterStyles from which option values are inherited. Multiple MeterStyles are delimited with pipes (`|`).

`MeasureName`, **`MeasureName2`**, **`MeasureName3`**...

"Binds" the meter to one or more measures. This means that the meter displays the values of these measures in some way. The exact form of the display depends on the type of meter. See each meter type's page for details about what kind of values are valid for that type, and how the values are displayed.

`X`, **`Y`**Default: `0`

Specify the x and y position of the meter relative to the top-left edge of the skin. If the value is appended with`r`, the position is relative to the previous meter x or y position (e.g. `5r` or `(5 * 2)r`). If the value is appended with `R`, the position is relative to the bottom-right edge of the previous meter.

`W`, **`H`**

The width and height of the meter. String meters and Image meters (with an image) can automatically determine width and height. For all other cases, `W` and `H` must be defined.

`Hidden`Default: `0`

If set to `1`, the meter is hidden. The visibility can also be changed with the !ShowMeter and !HideMeter bangs.

`UpdateDivider`Default: `1`

Sets the update frequency of the meter.

Example: If set to `1`, the meter is updated on every update cycle. If set to `5`, the meter is updated on every fifth update cycle and so forth.

`OnUpdateAction`

Action to execute on each Update of the meter. This option obeys any UpdateDivider on the meter.

SolidColor, **SolidColor2**Default: `0,0,0,0`

Color of the meter background. If `SolidColor2` is also specified, the background is a gradient composed of`SolidColor` and `SolidColor2`.

Hint: `SolidColor=0,0,0,1` can be used to make transparent areas of the meter clickable.

PaddingFormat: `Left,Top,Right,Bottom`

Allows adding padding in pixels around any or all sides a meter. The width and height of the meter will dynamically be adjusted to the new size.

Example: `Padding=5,10,5,10`

GradientAngle

Angle of the gradient in degrees (for `SolidColor` and `SolidColor2`).

BevelTypeDefault: `0`

If enabled, draws a bevel around the edges of the rectangle specified by `H` and `W`. Valid values are:

- `0`: No bevel
- `1`: Raised
- `2`: Sunken

AntiAliasDefault: `0`

If set to `1`, antialising is used to display the meter.

DynamicVariablesDefault: `0`

If set to `1`, the meter is dynamic.

See also: Dynamic Variables

TransformationMatrixDefault: `1;0;0;1;0;0`

Defines a 3x2 matrix which can be used to transform the meter. Transformations include: scaling, skewing, and translating (ie. moving). There must be **exactly** 6 values separated by semicolons `;`. Combining these can have drastic effects on the meter it is applied to.

See also: Transformation Matrix

Examples:

- `TransformationMatrix=-1; 0; 0; 1; 40; 0`: This will flip `X` along the line `X=20`.

- `TransformationMatrix=1; 0; 0; -1; 0; 100`: This will flip `Y` along the line `Y=50`.
- `TransformationMatrix=0.5; 0; 0; 1; 25; 0`: This will scale `X` by `0.5` at `X=50`.

Note: All transformations are relative to the top left corner of the window and not to the meter itself. So if you want to rotate the meter by its center the translation component in the matrix must be relative to the top left corner of the window.

Also note that the even if the meter's visual location and orientation is changed by the transformation the place where it would be located without the transformation will still be used to define the window size and register the mouse clicks. This might change in the future though.

General Image Options [home]

Options available for use with all images. These options are to modify the display of an image file, and do not work with square/rectangle Image meters created entirely with SolidColor.

Note: Valid image file types in Rainmeter are **.png**, **.jpg**, **.bmp**, **.gif**, **.tif** and **.ico**. If no extension is provided on an image file name, **.png** is assumed.

Options

ImagePath

Path of the image location.

ImageCrop

Crops the image. The value should be in the form: `X, Y, W, H, Origin`. Origin is optional and can be set to one of the following:

- `1`: Top left.
- `2`: Top right.
- `3`: Bottom right.
- `4`: Bottom left.
- `5`: Center (both W and H).

Greyscale Default: `0`

If set to `1`, the image is greyscaled.

ImageTint Default: `255,255,255,255`

Color to tint the image with. If the alpha value is specified, the image can be made semi-transparent (0 means invisible, 255 mean fully opaque). The default value (`255,255,255,255` for opaque white) and leaves the image unaltered.

Note: Combining `Greyscale` and `ImageTint` recolors the image to the specified color. Without `Greyscale`, the specified color is added to the image (i.e. the image is tinted).

ImageAlpha Default: `255`

Opacity of the image ranging from `0` (invisible) to `255` (opaque). If set, overrides the alpha component specified in `ImageTint`.

ImageFlip Default: `None`

Flips the image. Valid values are `None`, `Horizontal`, `Vertical` or `Both`.

ImageRotate Default: `0.0`

Rotates the image by the specified angle in degrees. Negative angles can be used for counter-clockwise rotation.

UseExifOrientation Default: `0`

If set to `1`, the image is rotated based on the EXIF data encoded in the image by a camera.

ColorMatrix*N*

Defines a 5x5 matrix used to manipulate the color values of the image. It is divided into five separate options, one for each row, each numbered. The default matrix is as follows:

```
ColorMatrix1=1; 0; 0; 0; 0

ColorMatrix2=0; 1; 0; 0; 0

ColorMatrix3=0; 0; 1; 0; 0

ColorMatrix4=0; 0; 0; 1; 0

ColorMatrix5=0; 0; 0; 0; 1
```

Select all

The values on the main diagonal are, from top-left to bottom-right: Red, Green, Blue, Alpha and a placeholder. The values represent the percentage of the particular value present in the image, where 0.0 is none and 1.0 is normal. The remaining entries in the matrix allow a color to have its value modified in terms of another color (e.g. the value of Red might have half of the Blue value added), with the entries in the final row (ColorMatrix5) determining offset values that are added directly to the color (e.g. `ColorMatrix5=0.5; 0; 0; 0; 1` adds 50% to the red value).

See also: ColorMatrix Guide.

MeterStyles [home]

Meters can use option values from other meters. Using the MeterStyle option, one meter may **inherit** options from one or more "parent" sections.

Usage

If an option is given on both the parent meter and the child meter, the child's setting overrides the parent's for that meter. If multiple parents are given, separated by pipes (`|`), options on later parents override those on earlier parents.

Inherited options are used just as if they were explicitly written in the child meter. For example, if the#CURRENTSECTION# variable is used in an inherited option, it resolves as the name of the child meter, not the parent. Similarly, if a relative position is used, such as `X=5R`, the meter immediately previous to the child meter is used, not the parent's.

The only options that cannot be inherited are:

- Meter. The meter type must be explicitly set in order to identify a section as a meter.
- MeterStyle itself. This means that MeterStyles cannot be inherited through more than one generation. In other words, meters cannot have "grandparents."

MeterStyle Sections

Meters may inherit options from sections that are not meters. A skin section of any name[1] that does not have the`Meter` or `Measure` option is treated as a MeterStyle. A MeterStyle section is completely ignored by Rainmeter unless it is referenced in the `MeterStyle` option of a meter. Options that are not valid for the child meter are also simply ignored, which means that the parent and child can be different types.

1. Aside from [Rainmeter] and [Variables], which are reserved for special purposes.

Example

```
[MyFirstParent]

Meter=String

Text=I'm a parent!

X=10

FontColor=255,0,0

FontFace=Segoe UI

FontSize=15

AntiAlias=1
```

```
StringStyle=Bold


[MySecondParent]

StringStyle=Italic

FontColor=0,255,0


[MyChild]

Meter=String

MeterStyle=MyFirstParentMeter | MySecondParentMeter

FontColor=0,0,255
```

Select all

MyChild inherits all options from the *MyFirstParent* meter and the *MySecondParent* MeterStyle. This means it is displayed with all of the same font, color, size and other options given for those sections. The second parent's `StringStyle` option overrides the first parent's, which means the child meter displays italic text. Likewise, the child meter's `FontColor` option overrides both parents', so the text displays with a color of `0,0,255` (blue).

Tooltips [home]

Creates a tooltip which appears when the mouse is hovered over the meter.

Options

`ToolTipText`

Text to display. This option must be specified in order to use the tooltip.

Values from `MeasureName` from the meter can be used with `%1`, `%2` etc. as appropriate for various meter types:

- **Line, String:** `%1`, `%2`, `%3`, ...
- **Histogram:** `%1`, %2
- **Others:** `%1`

Note: To wrap the text on multiple lines, use the #CRLF# variable.

`ToolTipTitle`

Title of the tooltip. Only one line of text can be used.

`ToolTipIcon`

Specifies the icon to use for the tooltip. This can be the path to a .ico file or one of the following preset icons:

- `Info`
- `Warning`
- `Error`
- `Question`
- `Shield`

Note: `ToolTipTitle` must be specified to use `ToolTipIcon`.

`ToolTipType`Default: `0`

If set to `1`, a balloon tooltip is displayed. Otherwise a normal tooltip displayed.

`ToolTipWidth`Default: `1000`

Maximum width for the tooltip. When the width is reached, the text will automatically wrap.

Note: As a `ToolTipTitle` cannot be wrapped, do not set this width less than the length of the title, or there could be unexpected results.

ToolTipHiddenDefault: `0`

If set to `1`, the tooltip is not displayed.

Note: This option can also be used in the [Rainmeter] section to hide all tooltips in the skin.

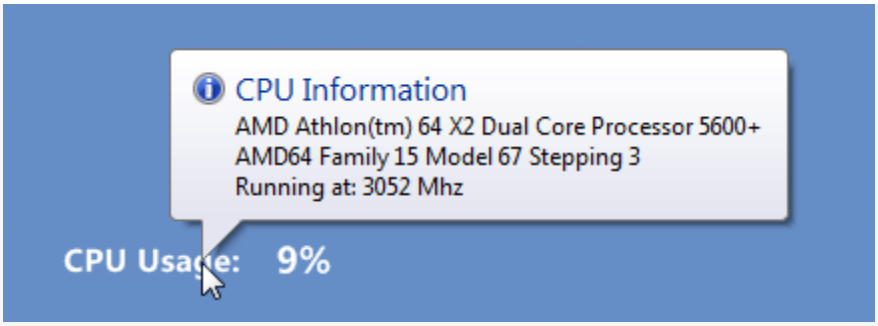
Example

```
[Rainmeter]
Update=1000

[MeasureCPU]
Measure=CPU

[MeasureCPUSpeed]
Measure=Registry
RegHKey=HKEY_LOCAL_MACHINE
RegKey=HARDWARE\DESCRIPTION\System\CentralProcessor\0
RegValue=~MHz
UpdateDivider=86400

[MeasureCPUName]
Measure=Registry
RegHKey=HKEY_LOCAL_MACHINE
RegKey=HARDWARE\DESCRIPTION\System\CentralProcessor\0
RegValue=ProcessorNameString
UpdateDivider=86400

[MeasureCPUIdentifier]
Measure=Registry
RegHKey=HKEY_LOCAL_MACHINE
RegKey=HARDWARE\DESCRIPTION\System\CentralProcessor\0
RegValue=Identifier
```

```
UpdateDivider=86400


[MeasureCPUText]

Meter=String

X=0

Y=0

FontFace=Segoe UI

FontColor=255,255,255,255

SolidColor=0,0,0,1

FontSize=12

StringStyle=Bold

StringAlign=Left

AntiAlias=1

Text="CPU Usage:"

ToolTipTitle=CPU Information

ToolTipType=1

ToolTipIcon=INFO

ToolTipText=[MeasureCPUName]#CRLF#[MeasureCPUIdentifier]#CRLF#Running at: [MeasureCPUSpeed] Mh
z

DynamicVariables=1


[MeterCPU%]

MeasureName=MeasureCPU

Meter=String

X=140

Y=-2r

FontFace=Segoe UI

FontColor=255,255,255,255

FontSize=14

StringStyle=Bold
```

```
StringAlign=Right

AntiAlias=1

NumOfDecimals=0

Percentual=1

Text="%1%"
```

Select all



An example skin demontrating the Tooltip option.

Bar meter [home]

`Meter=Bar` displays a horizontal or vertical bar that fills according to the percentual value of a measure.

Options

General meter options

> All general meter options are valid.

General image options

> All general image options are valid for `BarImage`.

`MeasureName`

> Name of the measure to use. The measure used must be able to return percentual values.

`BarColor`

> Color of the bar.

`BarImage`

> Path to an image to use for the bar instead of `BarColor`.

`BarBorder`

> If `BarImage` is specified, defines the number of pixels on either side of the image that are always drawn (i.e. top and bottom margins for vertical bars, left and right margins for horizontal bars).

`BarOrientation`Default: `Horizontal`

> Orientation of the bar. Valid values are `Horizontal` and `Vertical`.

`Flip`Default: `0`

> If set to `1`, the direction of the bar is flipped.

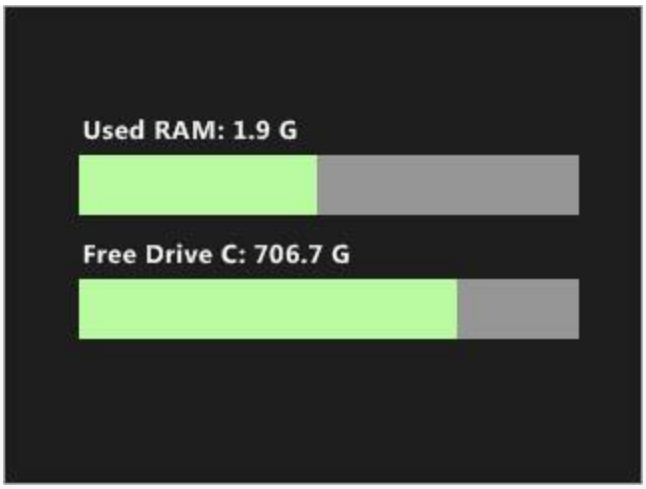
Example

```
[Rainmeter]
```

```
Update=1000
Author=RainmeterTeam

[MeasureUsedMemory]
Measure=PhysicalMemory

[MeasureUsedDrive]
Measure=FreeDiskSpace
Drive=C:

[MeterMemoryText]
Meter=String
MeasureName=MeasureUsedMemory
FontFace=Segoe UI
FontSize=10
FontColor=255,255,255,255
StringStyle=Bold
AutoScale=1
AntiAlias=1
Text=Used RAM: %1

[MeterUsedMemoryBar]
MeasureName=MeasureUsedMemory
Meter=BAR
Y=3R
W=250
H=30
BarColor=185,250,160,255
SolidColor=150,150,150,255
BarOrientation=Horizontal
```

```
[MeterDriveText]
Meter=String
MeasureName=MeasureUsedDrive
Y=10R
FontFace=Segoe UI
FontSize=10
FontColor=255,255,255,255
StringStyle=Bold
AutoScale=1
AntiAlias=1
Text=Free Drive C: %1

[MeterUsedDriveBar]
MeasureName=MeasureUsedDrive
Meter=Bar
Y=3R
W=250
H=30
BarColor=185,250,160,255
SolidColor=150,150,150,255
BarOrientation=Horizontal
```

Select all Download



An example skin demontrating a Bar meter.

Bitmap meter [home]

`Meter=Bitmap` displays a frame of an image depending on the measure value.

Options

General meter options

All general meter options are valid, **except** `W` and `H`.

General image options

All general image options are valid for `BitmapImage`, **except** `ImageCrop`, and `ImageRotate`.

`MeasureName`

Name of the measure used to determine the frame to display. The measures used must be able to returnpercentual values.

Example: If the image contains 5 frames, the first frame is displayed when the percentual measure value is between 0% and 19%, the second frame is shown from 20% to 39% , and so on.

`BitmapImage`

Path to the image. The frames in the images can be laid out either horizontally or vertically (the orientation is determined automatically from the height or the width of the image). The size of a single frame is also determined automatically, so no extra space should surround the frames.

If `BitmapExtend` is not set to `1`, the measure being used must return values from 0.0 to 1.0 or must have both `MaxValue` and `MinValue` set.

`BitmapFrames` Default: 1

Number of frames in the image.

`BitmapTransitionFrames`

The number of transition frames per each actual frame. The transition frames are frames which are displayed when the measure value changes. The same number of transition frames must be used after each regular frame. The TransitionUpdate option in the [Rainmeter] section determines the rate at which the transition frames change during the transition. The total duration of the transition is therefore `TransitionUpdate`multiplied by `BitmapTransitionFrames`.

Note: `BitmapFrames` always defines the total number of frames in the bitmap, including the transition frames.

Example: If the bitmap has 10 values and each transition consists of 4 additional frames, then `BitmapFrames`should be set to `50` and `BitmapTransitionFrames` to `4`.

`BitmapZeroFrame`Default: `0`

If set to `1`, the first frame is used only when the measured value is zero. Otherwise the frames are linearly determined by the measured value.

`BitmapExtend`Default: `0`

If set to `1` the bitmap is extended to display the whole value.

Example: If you define a bitmap that defines frames from 0 to 9 you can use this to display the measured value as the bitmap numbers.

`BitmapDigits`Default: `0`

Number of digits that are drawn (for `BitmapExtend=1`). The first frame is used if the value doesn't have as many digits as this defines.

`BitmapAlign`

Alignment of the bitmap value (for `BitmapExtend=1`). Valid values are `Left`, `Center`, and `Right`.

`BitmapSeparation`Default: `0`

Positive or negative number used as the separation between digits when `BitmapDigits` is higher than one.

Example

```
[Rainmeter]
Update=1000


[MeasureCPU]
Measure=CPU
```

```
[MeterLabel]
Meter=Image
ImageName=#@#Images\CPULabel.png
X=0
Y=0

[MeterCPU]
Meter=Bitmap
MeasureName=MeasureCPU
X=0
Y=5R
BitmapImage=#@#Images\nums.png
BitmapFrames=10
BitmapExtend=1
BitmapDigits=2

[MeterPercentSign]
Meter=Image
ImageName=#@#Images\percent.png
X=70R
Y=r
```

Select all Download

An example skin demontrating a Bitmap mete

Button meter [home]

`Meter=Button` displays a button with normal, hover, and pressed states.

Options

General meter options

> All general meter options are valid, **except** `W` and `H`.

General image options

> All general image options are valid for `ButtonImage`, **except** `ImageCrop`, and `ImageRotate`.

`ButtonImage`

> Path to the button image. The image should have 3 frames laid out either either horizontally or vertically (the orientation is determined by the width and height of the image). The first frame corresponds to the normal state, the second to the clicked state, and the third to the hover state.
>
> **See also:** Button Images

`ButtonCommand`

> Action to execute when the button is clicked.
>
> **Note:** Similar to `LeftMouseUpAction`. The difference is that `ButtonCommand` ignores transparent pixels in the image at all times, where `LeftMouseUpAction` will only ignore clicks on transparent areas if there is not some other meter behind the image.

Example

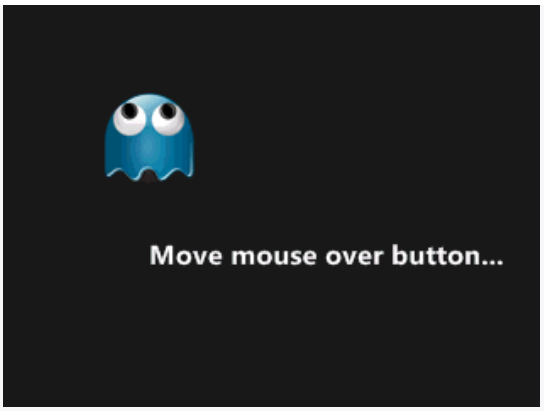
```
[Rainmeter]

Update=500

DynamicWindowSize=1


[MeterPacman]

Meter=Button
```

```
ButtonImage=#@#Images\PacButton.png

ButtonCommand=["Notepad"]
```

```
MouseOverAction=[!SetOption MeterText Text "Mouse OVER state..."][!UpdateMeter MeterText][!Redraw]
```

```
MouseLeaveAction=[!SetOption MeterText Text "Mouse OFF state..."][!UpdateMeter MeterText][!Redraw]

LeftMouseDownAction=[!SetOption MeterText Text "Mouse DOWN state..."][!UpdateMeter MeterText][!Redraw]


[MeterText]

Meter=String

FontFace=Segoe UI

FontSize=12

FontColor=255,255,255,255

StringStyle=Bold

AntiAlias=1

X=30

Y=90

Text=Move mouse over button...
```

Select all Download



An example skin demontrating a Button meter.

Histogram meter [home]

`Meter=Histogram` displays a histogram for the current and past values of one or two measures.

The primary graph is defined with MeasureName, with an optional secondary graph defined by MeasureName2.

Options

General meter options

> All general meter options are valid.

MeasureName, **MeasureName2**

> Name of the primary (required) and secondary (optional) measures to use for the histogram. The measure(s) used must be able to return percentual values.

AutoscaleDefault: `0`

> If set to `1`, the histogram is automatically scaled to show all the values.

GraphStartDefault: `Right`

> Starting point of the graph. Valid values are `Left` and `Right`.

GraphOrientationDefault: `Vertical`

> Orientation of the graph elements. Valid values are `Horizontal` and `Vertical`.

FlipDefault: `0`

> If set to `1`, the meter is flipped vertically.

PrimaryColor, **SecondaryColor**, **BothColor**Default: `008000`

> Color for the primary, secondary or both histograms.

PrimaryImagePath, **SecondaryImagePath**, **BothImagePath**

> Path to the location of optional image used behind the primary, secondary or both histograms.

PrimaryImage, **SecondaryImage**, **BothImage**

Optional image used behind the primary, secondary or both histograms.

Note: The image size cannot be modified with the W or H general meter options, and will be displayed in the original image size. The histogram will be constrained to the size of the image.

PrimaryImageCrop, **SecondaryImageCrop**, **BothImageCrop**

See ImageCrop.

PrimaryImageTint, **SecondaryImageTint**, **BothImageTint**

See ImageTint.

PrimaryImageAlpha, **SecondaryImageAlpha**, **BothImageImageAlpha**

See ImageAlpha.

PrimaryImageFlip, **SecondaryImageFlip**, **BothImageFlip**

See ImageFlip.

PrimaryImageRotate, **SecondaryImageRotate**, **BothImageRotate**

See ImageRotate.

PrimaryColorMatrix*N*, **SecondaryColorMatrix*N***, **BothImageColorMatrix*N***

See ColorMatrix*N*.

Deprecated Features

The following options have been deprecated and should not be used. They are still supported, but may be removed in future versions.

- `SecondaryMeasure`
  MeasureName2 should be used to define an optional secondary measure for the meter.

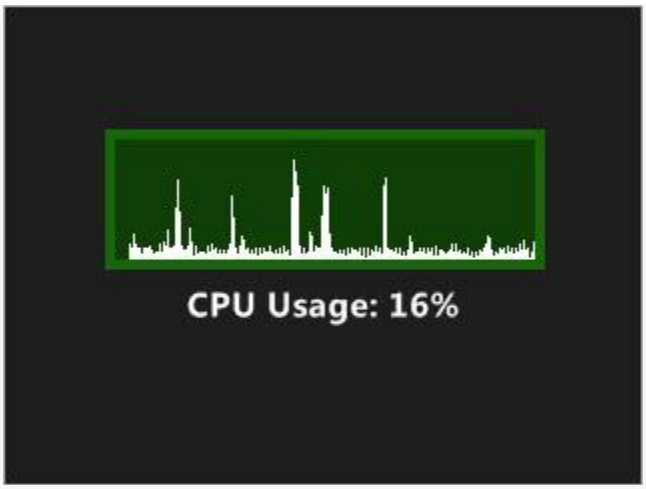
Example

```
[Rainmeter]

DynamicWindowSize=1

Update=500
```

```
[MeasureCPU]
Measure=CPU
Processor=0
MinValue=0
MaxValue=100

[MeterCPUBackgroundImage]
Meter=Image
SolidColor=24,102,10
X=0
Y=0
W=220
H=70

[MeterCPUHistogram]
Meter=Histogram
MeasureName=MeasureCPU
X=5
Y=5
W=210
H=60
PrimaryColor=255,255,255,255
SolidColor=0,0,0,100
AntiAlias=1

[MeterText]
Meter=String
MeasureName=MeasureCPU
X=110
Y=10R
```

```
FontFace=Segoe UI

FontSize=13

FontColor=255,255,255,255

StringStyle=Bold

StringAlign=Center

AntiAlias=1

Text=CPU Usage: %1%
```

Select all Download



An example skin demontrating a Histogram meter.

Image Meter [home]

`Meter=Image` displays either a static image or an image dependant on a measure value(s).

Options

General meter options

All general meter options are valid.

General image options

All general image options are valid for `ImageName`.

MeasureName, **MeasureName2**, ..., **MeasureName*N***

Name(s) of the measure(s) to use in `ImageName` with the `%N` syntax. This option does not need to be specified if not needed.

ImageNameDefault: `%1.png`

Name of the image file. The values of the measure(s) specified with `MeasureName` can be used with the `%N`syntax as follows:

```
MeasureName=SomeMeasure
MeasureName2=SomeMeasure2
ImageName=%1-%2.png
; %1 and %2 will be replaced by the string values of SomeMeasure and
; SomeMeasure2, respectively.
```

Select all

Note: If an extension is not provided for the image file, .png is assumed.

PreserveAspectRatioDefault: `0`

Controls how `W` and `H` scale the image when `Tile=0`. Valid values are:

- `0`: The image is scaled to exactly fit the bounds specified by `W` and `H` without regard to aspect ratio.
- `1`: The image is scaled to fit within the bounds specified by `W` and/or `H` while preserving the aspect ratio.
- `2`: The image is scaled and cropped to fill the bounds specified by `W` and/or `H` while preserving the aspect ratio.

ScaleMargins Format: `Left, Top, Right, Bottom`

Margins of the image to exclude from scaling when `Tile=0` and `PreserveAspectRatio=0`.

Example: `ScaleMargins=10, 50, 10, 50`

Tile Default: `0`

If set to `1`, the image is tiled (repeated) within the bounds defined by `W` and `H`.

Note: Using the SolidColor option, along with W and H options, an image meter may be used to draw squares, rectangles or lines without needing any `ImageName` or `MeasureName`.

Deprecated Features

The following options have been deprecated and should not be used. They are still supported, but may be removed in future versions.

- `Path`
  The Path option to define the location of images is replaced by the ImagePath option in general image options.

Example

```
[Rainmeter]

Update=1000


[MeterBackground]

Meter=Image

ImageName=#@#Images\Background.jpg

W=200

H=200

GreyScale=1

ImageTint=171,54,3,150


[MeasureMyPictures]

Measure=Plugin

Plugin=QuotePlugin

PathName=#@#\Pictures

Subfolders=0
```

```
FileFilter=*.jpg;*.gif;*.bmp;*.png
UpdateDivider=10


[MeterShowPicture]
```

```
Meter=Image
```

```
MeasureName=MeasureMyPictures
X=25
Y=25
W=150
H=150
PreserveAspectRatio=1
LeftMouseUpAction=!Refresh
```

Select all`Download`

An example skin demontrating an Image meter.

Line meter [home]

`Meter=Line` displays the measure values as a series of data points connected by straight line segments.

Options

General meter options

All general meter options are valid.

LineCountDefault: `1`

Number of lines in the meter.

MeasureName, **MeasureName2**, ..., **MeasureName*N***

Names of the measures to use as the source for a line. The measure(s) used must be able to return percentual values.

LineColor, **LineColor2**, ..., **LineColor*N***

Color for a line.

LineWidthDefault: `1.0`

Width of the line(s).

Scale, **Scale2**, ..., **Scale*N***Default: `1.0`

Scales (multiplies) the measure value to use for a line by the specified number.

Note: If `AutoScale` is enabled, this option is ignored.

AutoScaleDefault: `0`

If set to `1`, the lines are automatically scaled so that the largest value is visible in the meter. Otherwise the largest maximum value of the all of the measures used is used as the scale.

HorizontalLinesDefault: `0`

If set to `1`, horizontal marker lines are displayed behind the lines.

HorizontalLineColorDefault: `0,0,0,255`

Color of the horizontal marker lines (for `HorizontalLines=1`).

GraphStartDefault: `Right`

Starting point of the graph. Valid values are `Left` and `Right`.

GraphOrientation Default: `Vertical`

Orientation of the graph elements. Valid values are `Horizontal` and `Vertical`.

Flip Default: `0`

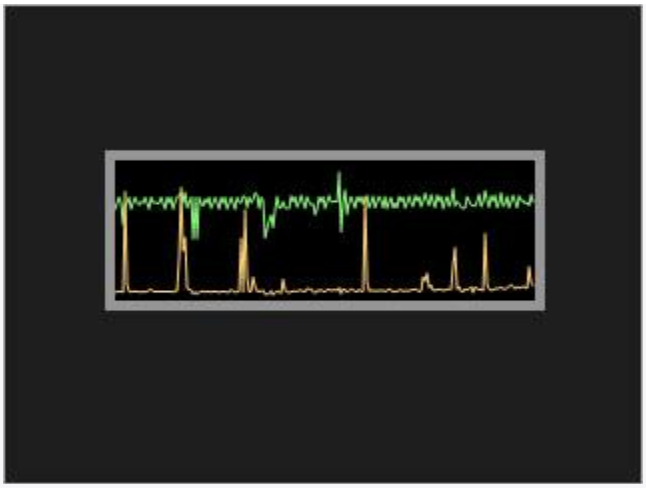
If set to `1`, the meter is flipped vertically.

Example

```
[Rainmeter]
Update=1000
DynamicWindowSize=1

[MeterBackground]
Meter=Image
W=220
H=80
SolidColor=150,150,150,255

[MeasureNetIn]
Measure=NetIn

[MeasureNetOut]
Measure=NetOut

[MeterNetworkLine]
Meter=Line
MeasureName=MeasureNetOut
MeasureName2=MeasureNetIn
X=5
Y=5
```

```
W=210

H=70

LineCount=2

LineColor=140,252,124,255

LineColor2=254,211,122,255

SolidColor=0,0,0,255

AutoScale=1

AntiAlias=1
```

Select allDownload



An example skin demontrating a Line meter.

Rotator meter [home]

`Meter=Rotator` displays an image that rotates around a point based on a measure.

The values for StartAngle and RotationAngle are defined in Radians

Options

General meter options

All general meter options are valid.

General image options

All general image options are valid for `ImageName`.

`ImageName`

Path of the image file.

`MeasureName`

Name of the measure whose percentual value controls the angle of rotation. The measure must be able to return percentual values.

`OffsetX`, **`OffsetY`**Default: `0.0`

X-offset and Y-offset of the center of rotation.

`StartAngle`Default: `0.0`

Start angle for the line in radians.

`RotationAngle`Default: `(2 * pi)`

The size of the rotation angle for the line in radians. Positive values result in clockwise rotation while negative values result in counter-clockwise rotation.

`ValueRemainder`Default: `0`

Use remainder instead of the actual measured value. This can be used to create an analog clock.

Remarks

The Rotator meter displays an image that rotates around a point where the angle of rotation is determined by the measure it is attached to. Rotator meters require that the measure being used is from 0.0 to 1.0 or has both MaxValue and MinValue set.

The center of rotation will be located at the center of the height and width specified for the meter. If height and width are not specified, the center of rotation will be located at the X and Y

coordinates provided for the meter. It is also important to remember that if the height and width are not specified, any portion of the meter that lies outside of the skin window will be cut off.

Here is an example of how to Rotate an Image Around its Center.

Example

```
[Rainmeter]

Update=1000
```

```
[MeasureTime]

Measure=Time


[MeterClockFace]

Meter=Image

ImageName=#@#Images\ClockFace.png

W=110

H=116


[MeterHoursHand]

Meter=ROTATOR

MeasureName=MeasureTime

X=0

Y=0

W=110

H=116

ImageName=#@#Images\Hours.png

OffsetX=3

OffsetY=3

StartAngle=4.7124

RotationAngle=6.2832

ValueRemainder=43200
```

```
[MeterMinutesHand]
Meter=ROTATOR
MeasureName=MeasureTime
X=0
Y=0
W=110
H=116
ImageName=#@#Images\Minutes.png
OffsetX=3
OffsetY=3
StartAngle=4.7124
RotationAngle=6.2832
ValueRemainder=3600

[MeterSecondsHand]
Meter=ROUNDLINE
MeasureName=MeasureTime
X=0
Y=0
W=110
H=116
LineLength=52
LineColor=247,220,129,255
LineWidth=2
AntiAlias=1
StartAngle=4.7124
RotationAngle=6.2832
ValueRemainder=60
```

Select all Download

An example skin demontrating a Rotator meter.

Roundline meter [home]

`Meter=Roundline` displays a single circular line that rotates around a point based on a measure.

The values for StartAngle and ControlAngle / RotationAngle are defined in Radians

Options

General meter options

All general meter options are valid.

MeasureName

Name of the measure whose percentual value controls the angle of rotation. The measure must be able to return percentual values.

Note: If MeasureName is not specified, then a number value of `1.0` is used by the meter.

LineWidth Default: 1

Width of the line.

LineLength

Length of the line. The length is always measured from the center of rotation (regardless of the `LineStart` option).

ControlLength, LengthShift

If ControlLength is set to 1, the measure value controls the `LineLength` from `LineLength` to `LineLength + LengthShift`.

LineStart

Defines the distance from the center at which the line starts.

ControlStart, StartShift

If `ControlStart` is set to 1, the measure controls the `LineStart` from `LineStart` to `LineStart + StartShift`.

StartAngle

The starting angle for the line. This is in radians and the zero angle is at the right. The default rotation direction is clockwise.

ControlAngle, RotationAngle

Unless `ControlAngle` is set to 0, the measure controls the `RotationAngle` from 0 to `RotationAngle`. The size of the rotation angle in radians for the line. Use a negative value for counter-clockwise rotation.

ValueRemainder

Use remainder instead of the actual measured value. This can be used to draw an analog clock.

LineColor

> The Color of the line.

Solid

> Set to 1 and the meter will draw a pie-chart instead.

Remarks

The angle of the line is determined by the measure. By default, the minimum position is pointing to the right, the line then moves clockwise until it is pointing to the right again.

If the width and height are not defined, the center point is at the X and Y position of the meter and any part of the meter that intersects with the edges of the skin window will be cut off. If width and height are specified, the center point will be in the middle of the bounding box and the skin window will include the meter.

If `MeasureName` is not specified, then a circle may be drawn without requiring any measure in the skin to create a value.

Example

```
[Rainmeter]

Update=1000


[MeasureFreeDisk]

Measure=FreeDiskSpace

Drive=C:


[MeasureTotalDisk]

Measure=FreeDiskSpace

Drive=C:

Total=1


[MeterTotalDisk]

Meter=Roundline

MeasureName=MeasureTotalDisk

X=0
```

```
Y=0
W=120
H=120
StartAngle=4.712
RotationAngle=6.283
LineLength=60
LineColor=62,140,132,255
Solid=1
AntiAlias=1

[MeterFreeDisk]
Meter=Roundline
MeasureName=MeasureFreeDisk
X=0
Y=0
W=120
H=120
StartAngle=4.712
RotationAngle=6.283
LineLength=56
LineColor=216,242,240,255
Solid=1
AntiAlias=1

[MeterTotalText]
Meter=String
MeasureName=MeasureTotalDisk
X=60
Y=130
FontFace=Segoe UI
```

```
FontSize=12

FontColor=62,140,132,255

StringStyle=Bold

StringAlign=Center

AntiAlias=1

AutoScale=1

Text=Total: %1


[MeterFreeText]

Meter=String

MeasureName=MeasureFreeDisk

X=60

Y=R

FontFace=Segoe UI

FontSize=12

FontColor=216,242,240,255

StringStyle=Bold

StringAlign=Center

AntiAlias=1

AutoScale=1

Text=Free: %1
```

Select all Download

An example skin demontrating a Roundline meter.

String meter [home]

`Meter=String` displays text.

Options

General meter options

All general meter options are valid.

`MeasureName`, `MeasureName2`, ..., `MeasureName`*`N`*

Name(s) of the measure(s) bound to the meter. The meter will display the current value of the measure defined in `MeasureName`, with values for additional measures bound to the meter available using the `%N` syntax in the Text option.

The meter does not require a `MeasureNameN` option if the Text option alone will be used to define the string to display.

Text Default: `%1`

Text to display. If `MeasureName` is specified, `Text` will default to the value of the measure.

The values of the measure(s) specified with `MeasureName` can be used with the `%N` syntax as follows:

```
MeasureName=SomeMeasure

MeasureName2=SomeMeasure2

Text=This is text containing %1 and %2.

; %1 and %2 will be replaced by the string values of SomeMeasure and

; SomeMeasure2, respectively.
```

Select all

The `Text` option can take any combination of the following forms:

- Display the value of a measure bound with `MeasureName`
- Display the values of multiple measures bound with `MeasureNameN`, and formatted with the `%N` syntax.
- Display static text defined in the option.
- Display the current value of any variables or section variables. *Note that the meter must contain the DynamicVariables option to use the current value of variables or measures defined as a section variable.*

Prefix

Text displayed before `Text`.

Note: It is preferable to put the entire string in `Text` instead of using this option.

Postfix

Text displayed after `Text`.

Note: It is preferable to put the entire string in `Text` instead of using this option.

FontFace Default: `Arial`

Family name of the font to use for the text. The font must either be installed in Windows directly or must be loaded at runtime.

See also: Fonts Guide.

FontSizeDefault: `10`

Size of the font.

FontColorDefault: `0,0,0,255`

Color of the font.

StringAlignDefault: `Left`

Horizontal and vertical alignment of the string. Valid values are:

- `Left`, `Right`, `Center` (or `LeftTop`, `RightTop`, `CenterTop`)
- `LeftBottom`, `RightBottom`, `CenterBottom`
- `LeftCenter`, `RightCenter`, `CenterCenter`

The string will be aligned using the values of the `X` and or `Y` settings as the anchor point. So to CenterCenter align a string within a meter with a width and height of 100, set `X=50`, `Y=50` and`StringAlign=CenterCenter`.

StringStyleDefault: `Normal`

Style of the string. Valid values are `Normal`, `Bold`, `Italic`, and `BoldItalic`.

StringCaseDefault: `None`

Converts the string to a case. Valid values are `None`, `Upper`, `Lower`, and `Proper`.

StringEffectDefault: `None`

Effect applied to a string. Valid values are `None`, `Shadow`, and `Border`.

FontEffectColorDefault: `0,0,0,255`

Color of the StringEffect.

ClipStringDefault: `0`

Controls how strings are truncated (clipped) or wrapped to fit in or expand the containing meter. Valid values are:

- `0`: Disabled. The string will not be clipped or wrapped. (default)
- `1`: Enabled. The string will be clipped with an added ellipsis `...` when it exceeds the specified `W` (width) option on the meter. If the `H` (height) option is large enough to allow multiple lines, the text is wrapped until the value of `H` is reached, then clipped.
- `2`: Auto. The string will be clipped or wrapped based on the value of `W` and/or `H`. If the width or height are not specified, the meter itself will change size to accommodate the string. This setting works in conjunction with

the ClipStringW and ClipStringH options below, to set a "maximum" size that the meter should expand to accomodate the string before clipping.

Note: The changing size of meters when `ClipString=2` can cause truncation issues with the overall window size of the skin, unless DynamicWindowSize=1 is set in the [Rainmeter] section of the skin.

ClipStringW

Sets a maximum width that the meter will expand to accommodate the string when `ClipString=2`. This setting is ignored if the `W` option is set.

ClipStringH

Sets a maximum height that the meter will expand to accommodate the string when `ClipString=2`. This setting is ignored if the `H` option is set.

AngleDefault: `0.0`

Defines the angle of the text in radians.

Note: The size and position of the text are always calculated as if the text is horizontal.

PercentualDefault: `0`

If set to `1`, the value of bound measures are converted to a percentage. This is useful if a measure does not return a percentage value, but either automatically defines a valid "range" of values (e.g. FreeDiskSpace) or when the MinValue and/or MaxValue options are manually set on the measure.

NumOfDecimalsDefault: `0`

Number of decimals to display with numerical measure values.

ScaleDefault: `1`

Scaling factor used for the measure values. The measure value is divided by the specified value. If the specified value has a decimal point (e.g. `1000.0`), the result will also display decimals.

Note: If `AutoScale` is enabled, this option is ignored.

AutoScaleDefault: `0`

Automatically scales the measure values. The scaled result is appended with k, M, G, etc. as appropriate. Valid values are:

- `0`: Disabled.
- `1`: Scales by 1024.
- `1k`: Scales by 1024 with kilo as the lowest unit.
- `2`: Scales by 1000.

- `2k`: Scales by 1000 with kilo as the lowest unit.

Note: Using the SolidColor option, with a value of `SolidColor=0,0,0,1`, will give a string meter a solid but virtually transparent background. This can make executing mouse actions on the text easier and more reliable.

Example

```
[Rainmeter]

Update=1000


[MeasureDate]

Measure=Time

Format=%A, %b %#d, %Y


[MeterDate]

Meter=String

MeasureName=MeasureDate

X=0

Y=0

FontColor=255,255,255,255

FontFace=Segoe UI

FontSize=14

StringEffect=Shadow

FontEffectColor=0,0,0,255

AntiAlias=1

Text=Today is: %1


[MeterText1]

Meter=String

X=0

Y=0R
```

```
FontColor=197,239,252,255

FontFace=Segoe UI

FontSize=12

AntiAlias=1

Text=Relative to bottom of previous meter


[MeterText2]

Meter=String

X=300

Y=10R

W=300

H=20

StringAlign=Right

FontColor=255,255,255,255

FontFace=Segoe UI

StringStyle=Italic

FontSize=14

AntiAlias=1

Text=Right justified italic text


[MeterText3]

Meter=String

X=150

Y=5R

W=300

H=20

StringAlign=Center

FontColor=252,245,197,255

FontFace=Segoe UI

StringStyle=Bold
```

```
FontSize=14

AntiAlias=1

Text=Centered bold text


[MeterText4]

Meter=String

X=0

Y=10R

W=200

H=50

ClipString=1

FontColor=197,252,223,255

FontFace=Segoe UI

StringStyle=Bold

FontSize=14

AntiAlias=1

Text=This text will wrap due to ClipString=1
```

Select all Download

An example skin showing various string options.

Measures [home]

A **measure** is an object that retrieves information to be used by the skin. Measures are one of the two major kinds of objects in a skin, along with meters.

Usage

Unlike meters, measures do not display anything by themselves. You cannot "see" a measure in a skin. The measure simply provides the information as a value, which can be used in several ways:

- A meter can be bound to a measure. In this way, the meter will automatically display the value in a way that is appropriate for the type of meter. This also takes the measure's range and scale into account (if applicable).
- The measure value can be referenced as a section variable. In this way, the value can be used in almost any meter or measure option. As a section variable, the value is used "literally," which means it does not automatically account for the range or scale.
- A measure can trigger an action when the value passes into a certain range.

Some measure types may have their own special rules. These are detailed in their individual articles.

Format

A measure is written as a section in the skin. All measures use the `Measure` option to define the section as a specific type of measure. Most other measure options depend on the type, but there are some general optionsthat are valid in many or all measures.

Below is an example of a complete working measure:

```
[MeasureFour]

Measure=Calc

Formula=2+2

UpdateDivider=-1
```

Select all

Values

A measure actually provides two values: a **string**, or "raw text," and a **number**, which can be used in calculation formulas. Depending on the type of measure, these values may be the same, which means you can use them interchangeably; or, they may be completely different. The article for each measure type explains what is provided for both the number and string values.

- When a meter is bound to a measure, it automatically uses the correct value for its type.
- When using Section Variables, different syntax is used to refer to a measure's string or number value.
- The Substitute option affects only the string value of the current measure.
- IfAction options use only the number value of the current measure.
- The `Formula` option in a Calc measure uses only the number values of other measures, unless section variables are used instead.

Although a string value is not a "true" number, it can be used like a number in formulas and options, as long as it contains only numeric characters and valid operators. (Otherwise, it is treated as zero.) Likewise, a number value can be displayed or stored to a variable as a string of text, although this may cause the value to lose precision.

Percentage

Some meters require that a measure provide a value that can be used as a **percentage**. Measures that provide a actual maximum range value based on the resource being measured (or with the range of 0.0 to 1.0) are interpreted as a percentage by these meters. Measures which do not provide a range will require that theMinValue and MaxValue options be set on the measure to define a low to high range. Setting these options does not effect the actual value of the measure, but only defines the high and low range used by meters.

Note: The range for a measure can be viewed in the Skins tab of the About window.

Order

Each time the skin updates, all measures in the skin are evaluated in order. This means that if a measure references the value of another measure that has not been evaluated yet, it will receive the value from the previous update.

For example:

```
[MeasureCounterPlusOne]

Measure=Calc

Formula=MeasureCounter + 1


[MeasureCounter]

Measure=Calc

Formula=Counter
```

Select all

The intent is for [MeasureCounterPlusOne] to add one to the value of [MeasureCounter], so that it is always greater than [MeasureCounter]. If the measures were in the opposite order, this would work. However, because [MeasureCounterPlusOne] is evaluated before [MeasureCounter], it will only add to the old, "outdated" value, and the two measures will remain equal.

General Measure Options [home]

Options available for use with all measures.

Options

`Measure`

Type of the measure (e.g. `CPU` or `FreeDiskSpace`).

`UpdateDivider`Default: 1

Frequency at which the measure value is updated. If set to `-1`, the measure will be updated only once on load or on refresh. Otherwise, the global `Update` option is multiplied by the specified value to determine the update frequency.

Example: If `Update=1000` (in the [Rainmeter] section) and `UpdateDivider=30`, the measure is updated every 30 seconds.

`OnUpdateAction`

Action to execute on each Update of the measure. This option obeys any UpdateDivider on the measure.

`OnChangeAction`

Action to execute when the number or string value of the measure changes. The initial change from "nothing" to a value when the skin is loaded or refreshed is ignored.

Note: With plugins that perform work in the background (e.g. WebParser and FileView), the action will be executed on the first change when loading or refreshing the skin.

InvertMeasureDefault: `0`

If set to `1`, the measure value is inverted.

Example: With the FreeDiskSpace measure, this option can be used to measure the used disk space instead of the free disk space.

MaxValueDefault: `1.0`

The maximum value of a measure. This is used to set a range of values for use in meters that require apercentual value. The actual value of the measure will remain unchanged.

MinValueDefault: `0.0`

The minimum value of a measure. This is used to set a range of values for use in meters that require apercentual value. The actual value of the measure will remain unchanged.

AverageSizeDefault: `1`

If set, the measure value will be an average of the specified number of past actual values.

DynamicVariablesDefault: `0`

If set to `1`, the measure is dynamic.

See also: Dynamic Variables

DisabledDefault: `0`

If set to `1`, the measure value is never updated. A disabled measure always returns 0 as its value in numerical contexts (e.g. when used in a Calc formula or with IfActions), but may still return a textual value (e.g. when used in String meters).

PausedDefault: `0`

If set to `1`, the measure value is never updated. A paused measure will return its most recent value.

IfActions [home]

IFActions are action options you add to a measure to execute one or more Bangs or commands when a defined threshold value is returned by the measure. IfActions are done by using pairs of Above, Equal, and Below Value and Action statements. You may have one of each kind of IfAction in a single measure.

Options

`IfAboveValue`

The value used by IfAboveAction.

`IfAboveAction`

Action to be executed when the measure goes above the value defined in IfAboveValue. The action is executed only at the moment when the measure exceeds the value, so it needs to go below the defined value before the action is executed again.

`IfBelowValue`

The value used by IfBelowAction.

`IfBelowAction`

Action to be executed when the measure goes below the value defined in IfBelowValue. The action is executed only at the moment when the measure falls below the value, so it needs to go above the defined value before the action is executed again.

IfEqualValue

The value used by IfEqualAction.

IfEqualAction

Action to be executed when the measure is equal with the value defined in IfEqualValue. The action is executed only once when the measure is equal to the value, so it needs to go above or below the defined value before the action is executed again. The compared value is rounded to an integer.

Example

```
[Rainmeter]
Update=1000
DynamicWindowSize=1

[MeasureCPU]
Measure=CPU
Processor=0
IfAboveValue=49
IfAboveAction=[!SetOption MeterCPU FontColor 255,0,0,255][!Redraw]
IfBelowValue=50
IfBelowAction=[!SetOption MeterCPU FontColor 0,255,0,255][!Redraw]

[MeterCPU]
Meter=String
MeasureName=MeasureCPU
X=0
Y=0
```

```
FontSize=15

FontColor=0,255,0,255

AntiAlias=1

Text=CPU: %1%
```

Select all

Substitute [home]

`Substitute` replaces some or all of a string value returned by a measure with another string.

Options

`Substitute`

> A list of comma delimited `"pattern":"replacement"` pairs. All occurences of `pattern` in the measure string value are replaced with `replacement` (e.g. `"This":"That"` substitutes all occurences of *This* with *That*).
>
> If multiple `"pattern":"replacement"` pairs are specified, each substitution is attempted in the specified order (e.g. `"This":"That","Here":"There"` first replaces all occurences of *This* with *That* and then replaces all occurences of *Here* with *There*.
>
> Note that the order in which the pairs are specified can be important. For example, with`"1":"One","10":"Ten"`, all occurences of *1* are replaced with *One*, but occurences of *10* will not be replaced with *Ten* (because the first pair already changed all *1* characters). For correct behaviour, the order should be reversed (i.e. `"10":"Ten","1":"One"`).
>
> Instead of `"pattern":"replacement"`, single quotes can be used either around the pattern or the replacement (i.e. `'pattern':"replacement"` or `"pattern":'replacement'`, but

not `'pattern':'replacement'`). This can be useful when either the pattern or the replacement contains double quotes (e.g. `'"':"double quote"` replaces all occurences of " with *double quote*).

RegExpSubstitute Default: `0`

If set to `1`, Perl compatible regular expressions can be used in the `pattern` part of `Substitute` pairs.

If captures are used in the pattern (e.g. `(.*)`), they can be referenced in the `replacement` part using `\1`(first capture), `\2` (second capture), etc. The entire match can also be referenced with `\0`.

Examples

Normal substitution:

```
[MeasureYear]
Measure=Time
Format=%Y
Substitute="2012":"Twenty Twelve","2013":"Twenty Thirteen"
; Assuming that the current year is 2012, the string value of [MeasureYear] will
; be "Twenty Twelve" (without quotes). Since Subsitute only affects the string
; value, the number value of [MeasureYear] will continue to be 2012.

[MeasureCalc1]
Measure=Calc
Formula=MeasureYear
; Since the number value of MeasureYear is used above, the value of [MeasureCalc1]
; will also be 2012.

[MeasureCalc2]
Measure=Calc
Formula=[MeasureYear]
DynamicVariables=1
; Since the string value of MeasureYear is used above, a syntax error will occur
```

```
; (as "Formula=Twenty Twelve" is not a valid formula).
```

Select all

Regular expression substitution:

```
; The Time measure is used below for example purposes.


[MeasureEx1]

Measure=Time

Format=I am Rainy

RegExpSubstitute=1

Substitute="(\w+) (\w+) (\w+)":"\3, \1 \2","Rainy":"Yoda"

; Reorders the sentence and then replaces Rainy with Yoda.

; The result is: Yoda, I am


[MeasureEx2]

Measure=Time

Format=Hello, world!

RegExpSubstitute=1

Substitute="^(.{0,5}).+$":"\1..."

; Truncates string by length (in this case 5) and appends "...".

; The result is: Hello...


[MeasureEx3]

Measure=Time

Format=192.168.1.101

RegExpSubstitute=1

Substitute="^(\d{1,3}).(\d{1,3}).(\d{1,3}).\d{1,3}$":"\1.\2.\3.***"

; Masks an IP address. The result is: 192.168.1.***
```

Select all

Calc measure [home]

`Measure=Calc` calculates mathematical formulas.

Note: The formula syntax and operators described for the Calc measure can also be used in formulas in other measure and meter options. Formulas used outside of a Calc measure must be enclosed in parentheses.

Options

General measure options

> All general measure options are valid.

`Formula`

> Formula to calculate. See below for syntax.

`UpdateRandom`Default: `0`

> If set to `1`, the random constant is regenerated on each update cycle.

`UniqueRandom`Default: `0`

If set to `1`, any measure using the random constant and UpdateRandom will not repeat until all values between and including LowBound and HighBound have been used. Note that any dynamic change to LowBound orHighBound will reset the tracking of values.

LowBoundDefault: `0.0`

Lower bound of the random constant.

HighBoundDefault: `100.0`

Upper bound of the random constant.

Formula Syntax

Operators

- `+`: addition
- `-`: substraction
- `*`: multiplication
- `/`: division
- `**`: power
- `%`: remainder or modulus
- `&`: bitwise and
- `|`: bitwise or
- `^`: bitwise xor
- `~`: bitwise not

Logical Operators

- `<>`: not equal
- `=`: equal to
- `>`: greater than
- `<`: less than
- `<=`: less than or equal to
- `>=`: greater than or equal to
- `&&`: logical and
- `||`: logical or

Note: Conditional statements evaluate to 1 or 0 (true / false).

Functions

- `atan(x)`, `asin(x)`, `acos(x)`, `cos(x)`, `sin(x)`, `tan(x)`: Standard trigonometric functions. *x* is in radians.
- `rad(x)` - Converts *x* degrees to radians.

- `abs(x)` - Absolute value of *x*.
- `exp(x)` - Returns e^x.
- `log(x)` - Base 10 logarithm of *x*.
- `ln(x)` - Natural logarithm of *x*.
- `sqrt(x)` - Square root of *x*.
- `sgn(x)` - Return 1 if x is positive, -1 if x is negative.
- `frac(x)` - Fractional, or decimal, part (e.g. frac(1.234) = 0.234).
- `trunc(x)` - Integer part (e.g. trunc(1.234) = 1).
- `floor(x)` - Floor of *x*.
- `ceil(x)` - Ceiling of *x*.
- `round(x, precision)` - Rounds *x* to an integer, or to a specified number of decimal places. *precision* is optional.

Constants

- `pi`: Mathematical constant Pi (~3.14159265...).
- `e`: Mathematical constant e (~2.71828182...).
- `random`: A random number. The number will be between and include the values set in `LowBound` and `HighBound`.
- `counter`: The number of update cycles from the time the skin is loaded. This number only resets when the skin is unloaded and then loaded again - not when the skin is refreshed.

Conditional Operations

`<condition> ? <expr. if true.> : <expr. if false.>`

This will evaluate condition as being either true or false. If it is true, the expression to the left of the colon (:) is evaluated. If it is false, the expression to the right is evaluated. This is equivalent to the following if-then-else statement:

```
if (condition)
  then
    expr. if true
  else
    expr. if false
end if
```

```
[MeasureOne]
Measure=Calc
Formula=5
```

```
[MeasureTwo]

Measure=Calc

Formula=MeasureOne &lt; 6 ? 1 : -1
```

Select all

This measure would return the number 1 since the condition `MeasureOne < 6` evaluates to true.

Conditional operators can be nested. It should be noted that there is a maximum of 30 nested operators.

```
[MeasureOne]

Measure=Calc

Formula=2


[MeasureTwo]

Measure=Calc

Formula=MeasureOne &lt; 1 ? 99 : (MeasureOne &lt; 2 ? 98 : (MeasureOne &lt; 3 ? 97 : 96))
```

Select all

This measure would return 97. Since the first statement of `MeasureOne < 1` is false, the formula begins testing the nested formulas in order until the condition becomes true . If none of the conditions are met, the final false value of 96 would be set.

Other Bases

The Calc measure allows numbers to be represented numbering systems other than decimal. To use another base, prefix the number with a zero then the letter representing the system you wish to use. The following are accepted prefixes, which are case (lower) sensitive:

- `0b` - Binary number (base 2) (ex: 0b110110 - returns 54 in decimal)
- `0o` - Octal number (base 8) (ex: 0o123 - returns 83 in decimal)
- `0x` - Hexadecimal number (base 16) (ex: 0xF1 - returns 241 in decimal)

CPU measure [home]

`Measure=CPU` measures CPU usage.

Options

General measure options

> All general measure options are valid.

`Processor`Default: `0`

> If set to `0`, measures the average of all CPU cores. If set to a number (`1`, `2`, etc.), measures a specific CPU core.

Value

The value of a CPU measure is a percentage from 0 to 100.

Example

```
[Rainmeter]
```

```
Update=1000

BackgroundMode=2

SolidColor=0,0,0,255


[MeasureAverageCPU]

Measure=CPU


[MeasureCPU1]

Measure=CPU

Processor=1


[MeterText]

Meter=String

MeasureName=MeasureAverageCPU

MeasureName2=MeasureCPU1

X=5

Y=5

W=100

H=35

FontColor=255,255,255,255

NumOfDecimals=1

Text="Average: %1%#CRLF#Core 1: %2%"
```

Select all

FreeDiskSpace measure [home]

`Measure=FreeDiskSpace` measures disk usage. By default the value will be the amount of free space. To obtain the used space, add InvertMeasure=1.

The measure will have a range from 0 to the actual total size of the measured disk resource for use by meters requiring a percentage.

Options

General measure options

> All general measure options **except** `MaxValue` are valid.

`Drive`Default: `C:`

> Defines the drive to measure.

`Total`Default: `0`

> If set to `1`, provides the total drive space.

`Label`Default: `0`

> If set to `1`, the string value is the drive label. The number value is not altered.

`Type`Default: `0`

If set to `1`, provides the drive type as a string and a number.

String	Number
Error	0
Removed	1
Removable	3
Fixed	4
Network	5
CDRom	6
Ram	7

Note: FreeDiskSpace does not support CD or DVD drives other than with `Type` and `Label`.

IgnoreRemovable Default: 1

If set to 1, removable drives are ignored. If set to 0, removable drives are measured. Be sure to set IgnoreRemovable=0 to measure a USB drive.

DiskQuota Default: 1

If set to 1, user specific disk quotas in Windows are obeyed. If set to 0, user account specific disk quotas are ignored when obtaining free or used space on the disk.

Example

```
[Rainmeter]
Update=1000
DynamicWindowSize=1

[MeasureDiskLabel]
Measure=FreeDiskSpace
Drive=C:
Label=1
```

```
UpdateDivider=5


[MeasureTotalDiskSpace]
Measure=FreeDiskSpace
Drive=C:
Total=1
UpdateDivider=5


[MeasureFreeDiskSpace]
Measure=FreeDiskSpace
Drive=C:
UpdateDivider=5


[MeasureUsedDiskSpace]
Measure=FreeDiskSpace
Drive=C:
InvertMeasure=1
UpdateDivider=5


[MeterDriveInfo]
Meter=String
MeasureName=MeasureDiskLabel
MeasureName2=MeasureTotalDiskSpace
MeasureName3=MeasureFreeDiskSpace
MeasureName4=MeasureUsedDiskSpace
X=0
Y=0
FontSize=10
FontColor=255,255,255,255
SolidColor=0,0,0,255
```

```
AntiAlias=1

AutoScale=1

Text="C:\ (%1): [Total: %2B] [Free: %3B] [Used: %4B]"
```

Select all

Memory measures [home]

`Measure=PhysicalMemory` measures Physical Memory (equal to RAM).
`Measure=SwapMemory` measures Swap Memory (equal to RAM + Pagefile.sys).
`Measure=Memory` measures Virtual Memory (equal to RAM + RAM + Pagefile.sys.

By default the value will be the amount of used memory. To obtain the free memory, add InvertMeasure=1.

The measures will have a MinValue / MaxValue range from 0 to the actual total size of the measured memory resource for use by meters requiring a percentage.

Options

General measure options

> All general measure options **except** `MaxValue` are valid.

`Total`Default: `0`

> If set to `1`, measures the total memory.

Example

```
[Rainmeter]

Update=1000

DynamicWindowSize=1


[MeasurePhysMemTotal]

Measure=PhysicalMemory

Total=1

UpdateDivider=3600
```

```
[MeasurePhysMemUsed]
Measure=PhysicalMemory
UpdateDivider=2

[MeasurePhysMemFree]
Measure=PhysicalMemory
InvertMeasure=1
UpdateDivider=2

[MeterText]
Meter=String
MeasureName=MeasurePhysMemTotal
MeasureName2=MeasurePhysMemUsed
MeasureName3=MeasurePhysMemFree
FontColor=255,255,255,255
SolidColor=0,0,0,255
NumOfDecimals=1
AutoScale=1
Text="RAM Total: %1B, RAM Used: %2B, RAM Free: %3B"
```

Select all

Net measures [home]

`Measure=NetIn` measures incoming (download) network traffic.
`Measure=NetOut` measures outgoing (upload) network traffic.
`Measure=NetTotal` measures total network traffic.

The bandwidth is measured in bytes.

Notes: The measured value is the number of bytes per second, times the total update rate of the measure, as defined by the `Update` option for the skin and any `UpdateDivider` option on the measure.

The measured value is the network traffic in and out of the network interface controller (NIC) of the computer running the skin. If the computer is on a network connected to a router (LAN), the traffic will include all interaction between the computer and both the router and any other devices on the LAN. It will not include traffic directly to and from other devices and the internet (WAN), and cannot be used to determine the amount of internet traffic.

Options

General measure options

> All general measure options are valid.

`Interface` Default: `0`

> Index of the network interface controller (NIC) to measure. If set to `0`, all interfaces are measured.

`Cumulative` Default: `0`

> If set to `1`, measures the cumulative network traffic. This can be used to measure the total consumed bandwidth during certain interval.

Example

```
[Rainmeter]

Update=1000

BackgroundMode=2

SolidColor=0,0,0,255


[MeasureNetIn]

Measure=NetIn
```

```
[MeasureNetOut]
Measure=NetOut


[MeterText]
Meter=String
MeasureName=MeasureNetIn
MeasureName2=MeasureNetOut
X=5
Y=5
W=100
H=20
FontColor=255,255,255,255
NumOfDecimals=1
AutoScale=1
Text="In: %1B, Out: %2B"
```

Select all

Plugin measure [home]

`Measure=Plugin` allows interactation with various DLL plugins specifically written to work with Rainmeter.

Options

General measure options

All general measure options are valid.

Plugin specific options

Plugins have their own specific options that must be placed in the measure section. For more information, refer to the Plugins entry for each.

Plugin

Name of the plugin.

Example

```
[Rainmeter]
Update=1000
BackgroundMode=2
SolidColor=0,0,0,255

[MeasureUserName]
Measure=Plugin
Plugin=SysInfo
SysInfoType=USER_NAME

[MeterText]
Meter=String
MeasureName=MeasureUserName
X=5
Y=5
W=100
H=25
FontColor=255,255,255,255
Text="User: %1"
```

Select all

Registry measure [home]

`Measure=Registry` measures the value of a key in the Windows Registry.

Options

General measure options

All general measure options are valid.

RegHKey

Name of the root key. Valid values are:

- `HKEY_CURRENT_CONFIG`
- `HKEY_CURRENT_USER`

- `HKEY_LOCAL_MACHINE`
- `HKEY_CLASSES_ROOT`
- `HKEY_PERFORMANCE_DATA`
- `HKEY_DYN_DATA`

RegKey

Name of the subkey.

RegValue

Name of the value. If not specified, the default value is retrieved.

Note: The two registry key value types supported are `REG_SZ` (string) and `REG_DWORD` (number). If a `REG_SZ` string value is numeric, both the string and number measure values will be set.

Example

```
[Rainmeter]
Update=1000
BackgroundMode=2
SolidColor=0,0,0,255

[MeasureWindowsVersion]
Measure=Registry
RegHKey=HKEY_LOCAL_MACHINE
RegKey=Software\Microsoft\Windows NT\CurrentVersion
RegValue=ProductName
UpdateDivider=-1

[MeterText]
Meter=String
MeasureName=MeasureWindowsVersion
X=5
Y=5
W=100
```

```
H=25

FontColor=255,255,255,255

Text="Version: %1"
```

Select all

Script measure [home]

`Measure=Script` measures information returned using the Lua scripting language.

See also: Lua Scripting for details on using Lua with Rainmeter.

Options

General measure options

> All general measure options are valid.

`ScriptFile`

> Path to the .lua script file.

`User defined options`

Any Key=Value option can be added to the measure, defining values which can be accessed by the Lua script using the SELF object.

Bangs

Script measure can be controlled with the !CommandMeasure bang. See Lua Scripting for more information.

Example

```
[Rainmeter]
Update=1000


[MeasureScript]
Measure=Script
ScriptFile=MyScript.lua
MyStringOption=Hello World
MyNumberOption=27
```

Select all

Time measure [home]

`Measure=Time` measures the current date and time.

Options

General measure options

All general measure options are valid.

Format Default: `%H:%M:%S`

Format for the measure value. This can be a combination of text and Format codes.

Note: If a `Format` is not given, the string value returned is in the format `%H:%M:%S`, however the number valuewill be a Windows timestamp. If a `Format` is given, the number value will be the value defined by the format, or zero if the format does not define a numerical value.

TimeStamp

Windows timestamp or formula defining a timestamp. If defined, the measure will use this date/time instead of the current system values.

TimeZoneDefault: `local`

If specified, GMT time is used, modified with the specified positive or negative offset number. E.g.`TimeZone=-5` would measure the time as `GMT -5.0`. If not specified, or set to `local`, local time for the computer is used.

DaylightSavingTimeDefault: `1`

If `DaylightSavingTime` is set to `0` and `TimeZone` is supplied, the current local offset for daylight saving time is not applied to the value.

Note: All locations do not follow the same daylight saving schedule, if any. If a Time measure is intended to provide the time in a specific location, and that location follows a different schedule from the user's local system, the default value will be wrong on certain dates over the course of the year. In order to provide an accurate time, `DaylightSavingTime` must be set to `0`, and `TimeZone` must be set in a way that accounts for the current daylight savings rules for that location.

Format codes

The following formatting codes can be used in the `Format` option.

- `%a`: Abbreviated weekday name.
- `%A`: Full weekday name.
- `%b`: Abbreviated month name.
- `%B`: Full month name.
- `%c`: Date and time representation appropriate for locale.
- `%d`: Day of month as number (01 - 31).
- `%H`: Hour in 24-hour format (00 - 23).
- `%I`: Hour in 12-hour format (01 - 12).
- `%j`: Day of year as number (001 - 366).
- `%m`: Month as number (01 - 12).
- `%M`: Minute as number (00 - 59).
- `%p`: Current locale's A.M./P.M. indicator for 12-hour clock.
- `%S`: Second as number (00 - 59).
- `%U`: Week of year as number, with Sunday as first day of week (00 - 53).
- `%w`: Weekday as number (0 - 6, Sunday is 0).
- `%W`: Week of year as number, with Monday as first day of week (00 - 53).
- `%x`: Date representation for current locale.
- `%X`: Time representation for current locale.
- `%y`: Year without century (00 - 99).
- `%Y`: Year with century.
- `%z`, `%Z`: Either the time-zone name or time zone abbreviation, depending on registry settings.
- `%%`: Percent sign.

- `%#c`: Long date and time representation, appropriate for current locale (e.g. "Tuesday, March 14, 1995, 12:41:29").
- `%#x`: Long date representation, appropriate to current locale (e.g. "Tuesday, March 14, 1995").

`locale-time` and `locale-date` will use the format that is set currently in Windows.

Note: To remove leading zeros in numerical format codes use `#` after `%` (e.g. `%#d` instead of `%d`).

Example

```
[Rainmeter]
Update=1000
BackgroundMode=2
SolidColor=0,0,0,255

[MeasureDate]
Measure=Time
Format=%A, %B %#d, %Y

[Measure12HrTime]
Measure=Time
Format=%#I:%M %p

[Measure24HrTime]
Measure=Time
Format=%H:%M

[MeterText]
Meter=String
MeasureName=MeasureDate
MeasureName2=Measure12HrTime
MeasureName3=Measure24HrTime
X=5
Y=5
```

```
W=200

H=25

FontColor=255,255,255,255

Text="Date: %1, 12-hour time: %2, 24-hour time: %3"
```

Select all

Uptime measure [home]

`Measure=Uptime` measures the time since the last restart of the computer.

The string value of the measure is defined by the Format option. The number value will be the amount of time since the last restart in seconds.

Options

General measure options

All general measure options are valid.

Format Default: `%4!i!d %3!i!:%2!02i!`

Format of the measure value. This can be a combination of text and the following codes:

- `%4`: Days.
- `%3`: Hours.
- `%2`: Minutes.
- `%1`: Seconds.

The following modify the codes:

- `!i!`: Putting this after the format code shows the numbers with no leading zeros.
- `!02i!`: Putting this after the format code shows the numbers with leading zeros.

AddDaysToHours Default: 1

If set to `1` and if `%4` (days) is not used in the `Format` option, `%3` (hours) is incremented by days * 24. Set to `0` to disable this behaviour.

Example

```
[Rainmeter]
Update=1000
BackgroundMode=2
SolidColor=0,0,0,255

[MeasureUptime]
Measure=Uptime
Format="%4!i! days, %3!i! hours, %2!i! minutes %1!i! seconds"

[MeterText]
Meter=String
MeasureName=MeasureUptime
X=5
Y=5
W=200
H=25
FontColor=255,255,255,255
Text="Uptime: %1"
```

Select all

Plugins [home]

Plugins are dynamic link library (.dll) programs specifically written to provide additional functionality not built into Rainmeter.

Usage

Plugins are used with a Plugin measure in the skin. All of the general measure options are valid with the measure, as well as additional options which are specific to each plugin.

Click on any of the plugins listed to learn more about the purpose and options for each.

Custom Plugins

3rd-party plugins developed for Rainmeter but not included with the installation may be used. Normally these would be distributed with a skin and installed using the Rainmeter Skin Installer, but may be manually installed by placing the .dll file in the `Plugins` folder under the settings path for the Rainmeter installation.

Note: Any 3rd-party plugin .dll must have been compiled for the same 32bit or 64bit architecture as the version of Rainmeter it is used with.

Those interested in creating 3rd-party plugins for Rainmeter can find more information and source code templates for both C++ and C# at Developers.

AdvancedCPU measure [home]

`Plugin=AdvancedCPU` measures CPU usage by processes.

Note: The value is a calculation of process CPU time scaled by the number of CPU cores. In order to use the value as a percentage in meters, it must have the MaxValue of the measure dynamically set to the current value of an AdvancedCPU measure which has no `CPUInclude` or `CPUExclude` options. See the Example below.

Options

General measure options

All general measure options are valid.

`CPUInclude`

List of process names separated by `;` (semicolon) to use in measurements . If specified, `CPUExclude` is ignored.

`CPUExclude`

List of processes separated by `;` (semicolon) to ignore in measurements.

Note: The process **Idle** is a placeholder process used to manage unused CPU, therefore it should be excluded with `CPUExclude=Idle` to obtain CPU or `TopProcess` usage.

`TopProcess`Default: `0`

Measures the process that is currently using the most CPU. Valid values are:

- `1` : The value will be the process CPU usage.
- `2` : The string value will be the process name. The number value is not altered.

Example

```
[Rainmeter]
Update=1000
DynamicWindowSize=1
BackGroundMode=2
SolidColor=0,0,0,255

;Measure maximum CPU time for use in MaxValue in other measures
[MeasureCPUMax]
Measure=Plugin
Plugin=AdvancedCPU

;Measure current CPU usage with MaxValue set
[MeasureCPU]
Measure=Plugin
Plugin=AdvancedCPU
CPUExclude=Idle
MaxValue=[MeasureCPUMax]
DynamicVariables=1

;Measure Name of top process
[MeasureTopName]
Measure=Plugin
Plugin=AdvancedCPU
CPUExclude=Idle
TopProcess=2
```

```
;Measure CPU usage of top process with MaxValue set
[MeasureTop%]
Measure=Plugin
Plugin=AdvancedCPU
CPUExclude=Idle
TopProcess=1
MaxValue=[MeasureCPUMax]
DynamicVariables=1

;Show current CPU usage as a percentage
[MeterCPU]
Meter=String
MeasureName=MeasureCPU
Y=2R
FontColor=255,255,255,255
FontSize=12
AntiAlias=1
Percentual=1
NumOfDecimals=2
DynamicVariables=1
Text=CPU Usage: %1

;Show name of top process
[MeterTopName]
Meter=String
MeasureName=MeasureTopName
Y=2R
FontColor=255,255,255,255
FontSize=12
```

```
AntiAlias=1


;Show CPU usage of top process as a percentage
[MeterTop%]
Meter=String
MeasureName=MeasureTop%
X=2R
Y=0r
FontColor=255,255,255,255
FontSize=12
AntiAlias=1
Percentual=1
NumOfDecimals=2
Text=(%1)
```

Select all

CoreTemp plugin [home]

`Plugin=CoreTemp` retrieves infromation from the CoreTemp application. The CoreTemp application **must** be running in the background.

Note: If the value of the measure is to be used in a meter which requires a percentage, then appropriate MinValueand/or MaxValue options must be added to the measure.

Options

General measure options

All general measure options are valid.

`CoreTempType`

Defines the information to measure. Valid values are:

- `Temperature`: Current temperature. `CoreTempIndex` must also be specified.
- `MaxTemperature`: Maximum temperature between all cores.
- `BusSpeed`: Bus frequency.
- `BusMultiplier`: Bus multiplier.
- `CpuName`: CPU model name.
- `CpuSpeed`: Core frequency.
- `TjMax`: Maximum allowed temperature. `CoreTempIndex` must also be specified.
- `Load`: Core load as percentage. `CoreTempIndex` must also be specified.
- `Vid`: Voltage value.

`CoreTempIndex`

Zero-based index of the core to measure. The first core is `0`, the second core is `1`, etc.

Example

```
[Rainmeter]

Update=1000

BackgroundMode=2

SolidColor=0,0,0,255
```

```
[MeasureMaxTemp]
Measure=Plugin
Plugin=CoreTemp
CoreTempType=MaxTemperature

[MeasureCore1Temp]
Measure=Plugin
Plugin=CoreTemp
CoreTempType=Temperature
CoreTempIndex=0

[MeasureCpuSpeed]
Measure=Plugin
Plugin=CoreTemp
CoreTempType=CpuSpeed

[MeterMaxTemp]
Meter=String
MeasureName=MeasureMaxTemp
X=5
Y=5
W=200
H=20
FontColor=255,255,255,255
Text="Max Temp: %1°C"

[MeterCore1Temp]
Meter=String
MeasureName=MeasureCore1Temp
```

```
X=5

Y=25

W=200

H=20

FontColor=255,255,255,255

Text="Core 1 Temp: %1°C"


[MeterCpuSpeed]

Meter=String

MeasureName=MeasureCpuSpeed

X=5

Y=45

W=200

H=20

FontColor=255,255,255,255

Text="Frequency: %1 MHz"
```

Select all

FileView plugin [home]

`Plugin=FileView` retrieves information about folders and files.

The plugin gathers all the folder and file names, sizes, dates and icons in the selected folder. It counts the number of files and folders and obtains the combined size. It can also search recursively through all the subfolders of the selected folder to obtain the totals for count and size.

FileView operates with a "parent / child" approach. A main "parent" FileView measure is used to obtain all the information for a selected folder, and then "child" measures are used to read individual entries from the parent using the `Path=` option.

Usage

FileView measures take the form:

```
[Rainmeter]
Update=1000
DynamicWindowSize=1

[MeasureFolder]
Measure=Plugin
Plugin=FileView
Path="C:\Program Files\Rainmeter"
ShowDotDot=0
ShowFolder=0
Count=3
```

Select all

In this example, this "parent" measure will obtain name, size, date and icon information about all files in the selected folder, then create three Index values for the first three files. The information is used in subsequent "child" FileView measures:

```
[MeasureChild1]
Measure=Plugin
Plugin=FileView
Path=[MeasureFolder]
Type=FileName
Index=1
```

```
[MeasureChild2]
Measure=Plugin
Plugin=FileView
Path=[MeasureFolder]
Type=FileName
Index=2

[MeasureChild3]
Measure=Plugin
Plugin=FileView
Path=[MeasureFolder]
Type=FileName
Index=3
```

Select all

The values of the three child measures are now the FileName information parsed into Indexes 1 through 3 by theparent measure. These can then be used with MeasureName and other options in meters.

Another way to use the information in child measures is:

```
[MeasureChild1]
Measure=Plugin
Plugin=FileView
Path=[MeasureFolder]
Type=FileName
Index=1

[MeasureChild2]
Measure=Plugin
Plugin=FileView
Path=[MeasureFolder]
Type=FileSize
Index=1
```

```
[MeasureChild3]
Measure=Plugin
Plugin=FileView
Path=[MeasureFolder]
Type=FileDate
Index=1
```

Select all

The values of the three child measures are now the FileName, FileSize and FileDate information from the first file (Index 1) of the parent measure.

In these examples, three indexes are created due to the `Count=3` option on the parent measure. However, the information for all files and / or folders are obtained by the plugin, and subsequent or previous entries can be assigned to the three indexes by using !CommandMeasure statements described below. This will allow a skin to "scroll" through the files and / or folders in a dynamic way.

Important Note: A FileView measure will not re-read the disk information on a normal update cycle or usingUpdateDivider on the measure, nor when the !Update / !UpdateMeasure bangs are used. If the options on the parent measure are changed dynamically with !SetVariable or !SetOption, the Update plugin command will need to be used to update the values.

```
[MeasureFolder]
Measure=Plugin
Plugin=FileView
Path="C:\ImageFolder"
Count=3

[MeterChangeFolder]
Meter=Image
W=26
H=25
SolidColor=0,0,0,255
LeftMouseUpAction=[!SetOption MeasureFolder Path "C:\VideoFolder"][!CommandMeasure MeasureFolder Update]
```

Select all

If it is desired that a folder be monitored for new or changed files, the Update plugin command should be used in conjunction with an OnUpdateAction statement to keep the measure current with changing disk information. Be aware that having the plugin physically access the disk has a performance cost, and care should be taken to use an approriate UpdateDivider.

```
[MeasureFolder]
Measure=Plugin
Plugin=FileView
Path="C:\ImageFolder"
Count=3
UpdateDivider=5
OnUpdateAction=!CommandMeasure MeasureFolder Update
```

Select all

Options

General measure options

> All general measure options are valid.

Path

> In a **parent measure**, defines the path of the folder to read. By default, the **My Computer** folder is used.
>
> In a **child measure**, defines the parent measure `[MeasureName]` to read values from.

Parent measure options

FinishAction

> Action to execute when the plugin has completed reading the folders and files. This can be used to ensure that a large folder structure is fully read before other actions are taken.

RecursiveDefault: `0`

> If set to `1`, the plugin searches all sub-folders updating only the file count, folder count and overall folder size. This option does not index specific files in sub-folders.

If set to `2`, the plugin indexes all files in the folder tree defined in Path. Folders are not indexed. Plugin commands FollowPath and PreviousFolder are disabled, and ShowFile, ShowFolder, and ShowDotDot options have no effect.

`Count`Default: `1`

The number of items to be indexed at one time.

`ShowDotDot`Default: `1`

If set to `1`, the **..** folder (representing the previous folder) will be included. Otherwise the **..** folder is ignored.

`ShowFolder`Default: `1`

If set to `0`, folders are ignored.

Note: To control the **..** folder, use ShowDotDot above.

`ShowFile`Default: `1`

If set to `0`, files are ignored.

`ShowHidden`Default: `1`

If set to `0`, hidden files and folders are ignored.

`ShowSystem`Default: `0`

If set to `1`, protected operating system files are included.

`HideExtensions`Default: `0`

If set to `1`, file extensions are removed when used with `Type=FileName`.

`Extensions`

Semi-colon separated list of file extensions that limits the type of files to be included.

Example: If `Extensions="jpg;png"`, only .jpg and .png files are included.

`SortType`Default: `Name`

Type of information to sort the entries by. Valid values are: `Name`, `Size`, `Type`, `Date`.

SortDateTypeDefault: `Modified`

File and folder date entry to use for sorting when `SortType=Date`. Valid values are: `Modified`, `Created`,`Accessed`.

SortAscendingDefault: `1`

If set to `1`, the entries are sorted in ascending order. Otherwise a descending order is used.

WildcardSearchDefault: `*`

Wildcards used to filter included files and/or folders. Standard `*` and `?` characters can be used.

Child measure options

IndexDefault: `1`

Index of the file or folder from the parent measure. This should not exceed the `Count` number in the parent measure. If it does, the items will wrap around. For example, if `Count=8`, `Index=9` will be treated as`Index=1`.

IgnoreCountDefault: `0`

If set to `1`, the Index will represent the actual index of the file or folder in the list. This is useful to display a particular file or folder at all times (like the `..` folder).

TypeDefault: `FolderPath`

Type of information to obtain from the parent measure. Valid values are: `FolderPath`, `FolderSize`,`FileCount`, `FolderCount`, `FileName`, `FileType`, `FileSize`, `FileDate`, `FilePath`, and `Icon`.

Note: If `Type=Icon`, the full path of the icon file is returned. See IconPath below. This can be used with`MeasureName` or `ImageName` in an Image meter to display the icon.

DateTypeDefault: `Modified`

The date entry to retrieve from the parent measure when `Type=Date`. Valid values are: `Modified`, `Created`,`Accessed`.

IconPath

Path of the folder to save icons when `Type=Icon`. If no path is given, the icons are saved in the skin folder and are named "iconX.ico", where "X" is the index number.

IconSizeDefault: `Medium`

Size of the icon to save. Valid values are: `Small` (for 16x16), `Medium` (for 32x32), `Large` (for 48x48),`ExtraLarge` (for 256x256).

Plugin Commands

Parent measure commands

Update

Updates the measure, reading the disk and recreating all values in the parent measure.

Example: `LeftMouseUpAction=!CommandMeasure "ParentMeasureName" "Update"`

PageUp

Decreases the current page count.

For example, if `Count=8`, and there are 25 files in the list, there is a total of 4 pages. If items 8-15 (page 2) are being displayed, the `PageUp` will decrease the page count by one changing the displayed items to items 0-7 (page 1).

Example: `LeftMouseUpAction=!CommandMeasure "ParentMeasureName" "PageUp"`

PageDown

Increases the page count.

IndexUp

Decreases the index by 1. This is useful for mouse scroll actions.

Example: `MouseScrollUpAction=!CommandMeasure "ParentMeasureName" "IndexUp"`

IndexDown

Increases the index by 1. This is useful for mouse scroll actions.

PreviousFolder

This will change the path to the folder one higher in the folder structure. Behaves like clicking on the **..** folder.

Child measure commands

FollowPath

If the index referenced in the child measure is currently a folder, then the parent measure's `Path` is updated to the new path. If it is a file, then it is opened with the default Windows associated application. In order to better simulate Windows behavior, it is recommended that this command be used with a double-click action.

Example: `LeftMouseDoubleClickAction=!CommandMeasure "ChildMeasureName" "FollowPath"`

Open

This will open the file or folder that the index represents. If it is a folder, the folder is opened in Windows Explorer. If it is a file, then it is opened with the default Windows associated application. In order to better simulate Windows behavior, it is recommended that this command be used with a double-click action.

Example: `LeftMouseDoubleClickAction=!CommandMeasure "ChildMeasureName" "Open"`

Example

```
[Rainmeter]

Update=1000

MouseScrollUpAction=[!CommandMeasure mPath "IndexUp"][!UpdateMeasure mPath][!UpdateMeasureGroup Children][!UpdateMeter *][!Redraw]

MouseScrollDownAction=[!CommandMeasure mPath "IndexDown"][!UpdateMeasure mPath][!UpdateMeasureGroup Children][!UpdateMeter *][!Redraw]


[Variables]

IconSize=Large


;-------------------------------------------------

; Styles

;-------------------------------------------------
```

```
[TextStyle]

FontColor=255,255,255,255

AntiAlias=1


[TextHighlight]

FontColor=150,150,255,255


[IconStyle]

X=5

Y=r

AntiAlias=1


[HighlightStyle]

SolidColor=0,0,0,1

X=5

Y=5R

W=380

H=([Index1Icon:H] &gt; [Index1Info:H] ? [Index1Icon:H] : [Index1Info:H])

DynamicVariables=1

MouseOverAction=[!SetOption #CURRENTSECTION# SolidColor "50,50,255,150"][!UpdateMeter #CURRENTSECTION#][!Redraw]

MouseLeaveAction=[!SetOption #CURRENTSECTION# SolidColor ""][!UpdateMeter #CURRENTSECTION#][!Redraw]


[InfoStyle]

X=5R

Y=r

Text="%1 #CRLF#%2 #CRLF#%3 "

AutoScale=1

AntiAlias=1
```

```
;--------------------------------------------------
; Measures
;--------------------------------------------------

[mPath]
Measure=Plugin
Plugin=FileView
Path="C:\"
Count=8

[mFolderCount]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=FolderCount
Group=Children

[mFileCount]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=FileCount
Group=Children

[mFolderSize]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=FolderSize
Group=Children
```

```
;-----------------------------------------------------
; Index 1

[mIndex1Name]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=FileName
Index=1
Group=Children

[mIndex1Size]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=FileSize
Index=1
Group=Children

[mIndex1Date]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=FileDate
Index=1
Group=Children

[mIndex1Icon]
Measure=Plugin
```

```
Plugin=FileView
Path=[mPath]
Type=Icon
IconSize=#IconSize#
Index=1
Group=Children

;-----------------------------------------------------
; Index 2

[mIndex2Name]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=FileName
Index=2
Group=Children

[mIndex2Size]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=FileSize
Index=2
Group=Children

[mIndex2Date]
Measure=Plugin
Plugin=FileView
Path=[mPath]
```

```
Type=FileDate

Index=2

Group=Children


[mIndex2Icon]

Measure=Plugin

Plugin=FileView

Path=[mPath]

Type=Icon

IconSize=#IconSize#

Index=2

Group=Children


;-----------------------------------------------------

; Index 3


[mIndex3Name]

Measure=Plugin

Plugin=FileView

Path=[mPath]

Type=FileName

Index=3

Group=Children


[mIndex3Size]

Measure=Plugin

Plugin=FileView

Path=[mPath]

Type=FileSize

Index=3
```

```
Group=Children


[mIndex3Date]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=FileDate
Index=3
Group=Children


[mIndex3Icon]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=Icon
IconSize=#IconSize#
Index=3
Group=Children


;-----------------------------------------------------
; Index 4


[mIndex4Name]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=FileName
Index=4
Group=Children
```

```
[mIndex4Size]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=FileSize
Index=4
Group=Children

[mIndex4Date]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=FileDate
Index=4
Group=Children

[mIndex4Icon]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=Icon
IconSize=#IconSize#
Index=4
Group=Children

;-----------------------------------------------------
; Index 5

[mIndex5Name]
Measure=Plugin
```

```
Plugin=FileView
Path=[mPath]
Type=FileName
Index=5
Group=Children

[mIndex5Size]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=FileSize
Index=5
Group=Children

[mIndex5Date]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=FileDate
Index=5
Group=Children

[mIndex5Icon]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=Icon
IconSize=#IconSize#
Index=5
Group=Children
```

```
;-----------------------------------------------------
; Index 6

[mIndex6Name]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=FileName
Index=6
Group=Children

[mIndex6Size]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=FileSize
Index=6
Group=Children

[mIndex6Date]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=FileDate
Index=6
Group=Children

[mIndex6Icon]
Measure=Plugin
```

```
Plugin=FileView
Path=[mPath]
Type=Icon
IconSize=#IconSize#
Index=6
Group=Children

;-----------------------------------------------------
; Index 7

[mIndex7Name]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=FileName
Index=7
Group=Children

[mIndex7Size]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=FileSize
Index=7
Group=Children

[mIndex7Date]
Measure=Plugin
Plugin=FileView
Path=[mPath]
```

```
Type=FileDate
Index=7
Group=Children

[mIndex7Icon]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=Icon
IconSize=#IconSize#
Index=7
Group=Children

;-----------------------------------------------------
; Index 8

[mIndex8Name]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=FileName
Index=8
Group=Children

[mIndex8Size]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=FileSize
Index=8
```

```
Group=Children


[mIndex8Date]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=FileDate
Index=8
Group=Children

[mIndex8Icon]
Measure=Plugin
Plugin=FileView
Path=[mPath]
Type=Icon
IconSize=#IconSize#
Index=8
Group=Children


;-----------------------------------------------------
; Meters
;-----------------------------------------------------

[Background]
Meter=Image
SolidColor=0,0,0,200
W=400
H=500

[PathTitle]
```

```
Meter=String
MeterStyle=TextStyle
Text=Path:

[Path]
Meter=String
MeasureName=mPath
MeterStyle=TextStyle | TextHighlight
Text="%1 "
X=R

[FolderCountTitle]
Meter=String
MeterStyle=TextStyle
X=0
Y=R
Text=Folders:

[FolderCount]
Meter=String
MeasureName=mFolderCount
MeterStyle=TextStyle | TextHighlight
X=R
Y=r

[FileCountTitle]
Meter=String
MeterStyle=TextStyle
X=10R
Y=r
```

```
Text=Files:


[FileCount]

Meter=String

MeasureName=mFileCount

MeterStyle=TextStyle | TextHighlight

X=R

Y=r


[FolderSizeTitle]

Meter=String

MeterStyle=TextStyle

X=10R

Y=r

Text=Size:


[FolderSize]

Meter=String

MeasureName=mFolderSize

MeterStyle=TextStyle | TextHighlight

X=R

Y=r

AutoScale=1


[Index1]

Meter=Image

MeterStyle=HighlightStyle

LeftMouseDoubleClickAction=[!CommandMeasure mIndex1Name "FollowPath"][!UpdateMeasure mPath][!U
pdateMeasureGroup Children][!UpdateMeter *][!Redraw]


[Index1Icon]
```

```
Meter=Image
MeasureName=mIndex1Icon
MeterStyle=IconStyle

[Index1Info]
Meter=String
MeasureName=mIndex1Name
MeasureName2=mIndex1Size
MeasureName3=mIndex1Date
MeterStyle=TextStyle | InfoStyle

[Index2]
Meter=Image
MeterStyle=HighlightStyle
LeftMouseDoubleClickAction=[!CommandMeasure mIndex2Name "FollowPath"][!UpdateMeasure mPath][!UpdateMeasureGroup Children][!UpdateMeter *][!Redraw]

[Index2Icon]
Meter=Image
MeasureName=mIndex2Icon
MeterStyle=IconStyle

[Index2Info]
Meter=String
MeasureName=mIndex2Name
MeasureName2=mIndex2Size
MeasureName3=mIndex2Date
MeterStyle=TextStyle | InfoStyle

[Index3]
Meter=Image
```

```
MeterStyle=HighlightStyle
LeftMouseDoubleClickAction=[!CommandMeasure mIndex3Name "FollowPath"][!UpdateMeasure mPath][!UpdateMeasureGroup Children][!UpdateMeter *][!Redraw]

[Index3Icon]
Meter=Image
MeasureName=mIndex3Icon
MeterStyle=IconStyle

[Index3Info]
Meter=String
MeasureName=mIndex3Name
MeasureName2=mIndex3Size
MeasureName3=mIndex3Date
MeterStyle=TextStyle | InfoStyle

[Index4]
Meter=Image
MeterStyle=HighlightStyle
LeftMouseDoubleClickAction=[!CommandMeasure mIndex4Name "FollowPath"][!UpdateMeasure mPath][!UpdateMeasureGroup Children][!UpdateMeter *][!Redraw]

[Index4Icon]
Meter=Image
MeasureName=mIndex4Icon
MeterStyle=IconStyle

[Index4Info]
Meter=String
MeasureName=mIndex4Name
MeasureName2=mIndex4Size
```

```
MeasureName3=mIndex4Date
MeterStyle=TextStyle | InfoStyle


[Index5]
Meter=Image
MeterStyle=HighlightStyle
LeftMouseDoubleClickAction=[!CommandMeasure mIndex5Name "FollowPath"][!UpdateMeasure mPath][!UpdateMeasureGroup Children][!UpdateMeter *][!Redraw]


[Index5Icon]
Meter=Image
MeasureName=mIndex5Icon
MeterStyle=IconStyle


[Index5Info]
Meter=String
MeasureName=mIndex5Name
MeasureName2=mIndex5Size
MeasureName3=mIndex5Date
MeterStyle=TextStyle | InfoStyle


[Index6]
Meter=Image
MeterStyle=HighlightStyle
LeftMouseDoubleClickAction=[!CommandMeasure mIndex6Name "FollowPath"][!UpdateMeasure mPath][!UpdateMeasureGroup Children][!UpdateMeter *][!Redraw]


[Index6Icon]
Meter=Image
MeasureName=mIndex6Icon
MeterStyle=IconStyle
```

```
[Index6Info]
Meter=String
MeasureName=mIndex6Name
MeasureName2=mIndex6Size
MeasureName3=mIndex6Date
MeterStyle=TextStyle | InfoStyle

[Index7]
Meter=Image
MeterStyle=HighlightStyle
LeftMouseDoubleClickAction=[!CommandMeasure mIndex7Name "FollowPath"][!UpdateMeasure mPath][!UpdateMeasureGroup Children][!UpdateMeter *][!Redraw]

[Index7Icon]
Meter=Image
MeasureName=mIndex7Icon
MeterStyle=IconStyle

[Index7Info]
Meter=String
MeasureName=mIndex7Name
MeasureName2=mIndex7Size
MeasureName3=mIndex7Date
MeterStyle=TextStyle | InfoStyle

[Index8]
Meter=Image
MeterStyle=HighlightStyle
LeftMouseDoubleClickAction=[!CommandMeasure mIndex8Name "FollowPath"][!UpdateMeasure mPath][!UpdateMeasureGroup Children][!UpdateMeter *][!Redraw]
```

```
[Index8Icon]
Meter=Image
MeasureName=mIndex8Icon
MeterStyle=IconStyle

[Index8Info]
Meter=String
MeasureName=mIndex8Name
MeasureName2=mIndex8Size
MeasureName3=mIndex8Date
MeterStyle=TextStyle | InfoStyle

[PageUp]
Meter=String
MeterStyle=TextStyle
FontSize=15
X=5
Y=10R
Text=Page Up
LeftMouseDoubleClickAction=[!CommandMeasure mPath "PageUp"][!UpdateMeasure mPath][!UpdateMeasureGroup Children][!UpdateMeter *][!Redraw]
MouseOverAction=[!SetOption #CURRENTSECTION# SolidColor "50,50,255,150"][!UpdateMeter #CURRENTSECTION#][!Redraw]
MouseLeaveAction=[!SetOption #CURRENTSECTION# SolidColor ""][!UpdateMeter #CURRENTSECTION#][!Redraw]

[PageDown]
Meter=String
MeterStyle=TextStyle
FontSize=15
```

```
X=10R

Y=r

Text=Page Down

LeftMouseDoubleClickAction=[!CommandMeasure mPath "PageDown"][!UpdateMeasure mPath][!UpdateMeasureGroup Children][!UpdateMeter *][!Redraw]

MouseOverAction=[!SetOption #CURRENTSECTION# SolidColor "50,50,255,150"][!UpdateMeter #CURRENTSECTION#][!Redraw]

MouseLeaveAction=[!SetOption #CURRENTSECTION# SolidColor ""][!UpdateMeter #CURRENTSECTION#][!Redraw]


[PreviousFolder]

Meter=String

MeterStyle=TextStyle

FontSize=15

X=10R

Y=r

Text=Previous Folder

LeftMouseDoubleClickAction=[!CommandMeasure mPath "PreviousFolder"][!UpdateMeasure mPath][!UpdateMeasureGroup Children][!UpdateMeter *][!Redraw]

MouseOverAction=[!SetOption #CURRENTSECTION# SolidColor "50,50,255,150"][!UpdateMeter #CURRENTSECTION#][!Redraw]

MouseLeaveAction=[!SetOption #CURRENTSECTION# SolidColor ""][!UpdateMeter #CURRENTSECTION#][!Redraw]
```

Select all

FolderInfo plugin [home]

`Plugin=FolderInfo` retrieves information about folders.

Note: Measuring a folder with a very large number of folders and files (e.g. My Documents) can cause Rainmeter to have erratic or even unstable behavior.

Options

General measure options

All general measure options are valid.

`Folder`

Must be either:

- The path to the folder (e.g. `Folder=C:\MyFolder`)
- Or the name of another measure (e.g. `Folder=[MainMeasure]`). In this case, the folder specified in MainMeasure will be used.

Important: If multiple measures use the same folder, specify the actual path on the first FolderInfo measure and specify the name of the first measure in subsequent measures. See example below for proper usage.

`InfoType`

Defines the type of information. Valid values are:

- `FileCount`: Number of found files.
- `FolderCount`: Number of found folders.
- `FolderSize`: Size of the folder in bytes.

`RegExpFilter`

Regular expression used to include or exclude counted files.

`IncludeSubFolders`Default: `0`

If set to `1`, subfolders are included.

`IncludeHiddenFiles`Default: `0`

If set to `1`, hidden files are included.

`IncludeSystemFiles`Default: `0`

If set to `1`, system files are included.

Example

```
[Rainmeter]
Update=1000
BackgroundMode=2
SolidColor=0,0,0,255

[MeasureFolder]
Measure=Plugin
Plugin=FolderInfo
Folder=#SKINSPATH#
InfoType=FolderSize
IncludeHiddenFiles=1
IncludeSubFolders=1
IncludeSystemFiles=1
RegExpFilter=.*
UpdateDivider=10

[MeasureFileCount]
Measure=Plugin
Plugin=FolderInfo
Folder=[MeasureFolder]
InfoType=FileCount
UpdateDivider=10

[MeterSize]
Meter=String
MeasureName=MeasureFolder
X=5
Y=5
W=200
H=20
```

```
FontColor=255,255,255,255

AutoScale=1

Text="Total Size: %1B"


[MeterCount]

Meter=String

MeasureName=MeasureFileCount

X=5

Y=25

W=200

H=20

FontColor=255,255,255,255

Text="File Count: %1"
```

Select all

InputText plugin [home]

`Plugin=InputText` executes bangs with user input.

InputText works by defining a series of commands, which can be triggered by any "action" string (this includes mouse actions, conditional actions in a Calc measure, and other commands in an InputText measure, including the same measure). When triggered, a free-floating text input box is created to gather

user input at the specified points in the command series. When all input has been submitted, the commands are carried out.

Options

`Command1`, **`Command2`**, ..., **`Command`*`N`***

Actions to execute when the plugin is triggered. The string `$UserInput$` will be replaced by whatever the user types into it. This string can be repeated, in which case, multiple input boxes will be created in sequence. In addition, a command can override the values of other keys in this measure when an input box is called on that command. **Note:** The value must be placed in quotation marks if it contains spaces.

`DefaultValue`

Default text that will appear in the input box.

`Password`Default: `0`

If set to `1`, input will be displayed as asterisks.

`X`, **`Y`**

Position of the input box.

Note: Relative positioning is not supported.

`W`, **`H`**

Size of the input box.

Note: `H` should be large enough to accomodate the font.

`SolidColor`

Color of the background.

Note: The alpha channel changes the opacity of the entire input box, not just its background.

`FontColor`

Color of the input text.

FontFace

> Family name of the input text font.
>
> **Note:** Only fonts installed directly in Windows can be used.

FontSize

> Size of the input text font.

StringStyle

> Style of the input text. Valid values are `Normal`, `Bold`, `Italic`, and `BoldItalic`.

StringAlignDefault: `Left`

> Alignment of the input text. Valid values are `Left`, `Right`, `Center`.
>
> **Note:** With right-aligned text, the cursor will move from left to right as text is typed. This is advised for skins using languages such as Arabic, which read right-to-left.

FocusDismissDefault: `0`

> If set to `1`, clicking anywhere other than on the input field will close the input without taking action. Otherwise, the mouse is disabled until Enter or Esc is pressed.

OnDismissActionDefault: `0`

> A bang or action that is executed when an open InputText field is dismissed without hitting Enter to execute the plugin commmands.

Bangs

```
LeftMouseUpAction=!CommandMeasure "MeasureInputTextPlugin" "ExecuteBatch ALL X=100"

IfAboveAction=!CommandMeasure "MeasureInputTextPlugin" "ExecuteBatch 3"

Command4=!CommandMeasure "MeasureInputTextPlugin" "ExecuteBatch 2-4 Password=1"
```

Select all

Notes

- Press Enter to submit user input. Press Escape to dismiss the input box without executing the script.
- Although the bang `!Refresh` would normally only refresh the current skin, InputText bangs are passed through Rainmeter itself, which means that the bang will perform a global refresh instead. To only refresh the skin, use `!Refresh #CURRENTCONFIG#`.

Example

```
[Rainmeter]

Update=1000

DynamicWindowSize=1


[Variables]

FirstVar=!WriteKeyValue

SecondVar=!SetVariable

FontSize=14


[MeterBackground]

Meter=Image

SolidColor=1E3A69FF

W=250

H=105


[MeasureInput]

Measure=Plugin

Plugin=InputText

SolidColor=76A0E8FF

FontColor=255,255,255,255

FontFace=Seqoe UI

StringStyle=Italic

FontSize=#FontSize#

X=5
```

```
Y=5

H=25

W=240

DefaultValue="Change Me!"

Command1=!WriteKeyValue Variables FirstVar "$UserInput$" "#CURRENTPATH#InputText.inc"

Command2=!Refresh #CURRENTCONFIG#

Command3=!SetVariable SecondVar "$UserInput$" Y=40 DefaultValue="Change Me Too!"

Command4=["$UserInput$"] Y=75 DefaultValue="Text file path and name"


[MeterWriteKeyValue]

Meter=String

X=5

Y=5

FontSize=15

FontColor=255,255,255,255

AntiAlias=1

Text=#FirstVar#

LeftMouseUpAction=!CommandMeasure "MeasureInput" "ExecuteBatch 1-2"


[MeterSetVariable]

Meter=String

X=5

Y=10R

FontSize=15

FontColor=255,255,255,255

AntiAlias=1

DynamicVariables=1

Text=#SecondVar#

LeftMouseUpAction=!CommandMeasure "MeasureInput" "ExecuteBatch 3"
```

```
[MeterOpenEditor]

Meter=String

X=5

Y=10R

FontSize=15

FontColor=255,255,255,255

AntiAlias=1

DynamicVariables=1

Text=Open Text File

LeftMouseUpAction=!CommandMeasure "MeasureInput" "ExecuteBatch 4"
```

Select all

iTunes plugin [home]

`Plugin=iTunesPlugin` retrieves the currently playing track infromation from the iTunes application.

Note: The iTunes plugin is deprecated. Use the NowPlaying plugin instead.

Options

General measure options

All general measure options are valid.

`Command`

Defines the information to measure. Valid values are:

- `GetSoundVolume`: Player volume between 0 - 100.
- `GetPlayerPosition`: Player position in seconds.
- `GetPlayerPositionPercent`: Player position as a percentage.
- `GetCurrentTrackAlbum`: Album.
- `GetCurrentTrackArtist`: Artist.
- `GetCurrentTrackBitrate` Bitrate.
- `GetCurrentTrackBPM`: Beats per minute.
- `GetCurrentTrackComment`: Track comment.
- `GetCurrentTrackComposer`: Track composer.
- `GetCurrentTrackEQ`: EQ preset name.
- `GetCurrentTrackGenre`: Genre (category).
- `GetCurrentTrackKindAsString`: File description.
- `GetCurrentTrackName`: Track name.
- `GetCurrentTrackRating`: Rating from 0 - 100.
- `GetCurrentTrackSampleRate`: Sample rate.
- `GetCurrentTrackSize`: File size.
- `GetCurrentTrackTime`: Length of the track.
- `GetCurrentTrackTrackCount`: Number of tracks on the album.
- `GetCurrentTrackTrackNumber`: Track number or index.
- `GetCurrentTrackYear`: Track year.
- `GetCurrentTrackArtwork`: Artwork file path. Use in combination with `DefaultArtwork`.

`DefaultArtwork`

Path of the artwork folder relative to the skin folder. Used with `Command=GetCurrentTrackArtwork`.

Bangs

iTunes measures can be controlled with the !CommandMeasure bang with the argument parameter being:

- `Backtrack`: Reposition to the beginning of the current track, or go to the previous track if already at start of current track.
- `FastForward`: Skip forward in a playing track.
- `NextTrack`: Advance to the next track in the current playlist.
- `Pause`: Pause playback.
- `Play`: Play the currently targeted track.
- `PlayPause`: Toggle the playing/paused state of the current track.

- `PreviousTrack`: Return to the previous track in the current playlist.
- `Resume`: Disable fast forward/rewind and resume playback if playing.
- `Rewind`: Skip backwards in a playing track.
- `Stop`: Stop playback.
- `Power`: Open/close iTunes application.
- `Quit`: Exit the iTunes application.
- `SoundVolumeUp`: Turn the volume up 5%.
- `SoundVolumeDown`: Turn the volume down 5%.
- `ToggleiTunes`: Show/hide iTunes window.

MediaKey plugin [home]

`Plugin=MediaKey` sends various keystrokes found on multimedia keyboards. It works with Spotify, Zune, foobar, and Windows Media Player. The plugin is controlled by the !CommandMeasure bang.

Options

General measure options

All general measure options are valid.

Bangs

To control keystrokes on multimedia keyboards, use the !CommandMeasure bang. In the examples below `MeasureMediaKey` is the section name for the MediaKey measure. Your section name may differ.

```
!CommandMeasure "MeasureMediaKey" "NextTrack"
```

Moves to the next track in the list.

```
!CommandMeasure "MeasureMediaKey" "PrevTrack"
```

Moves to the previous track in the list.

```
!CommandMeasure "MeasureMediaKey" "Stop"
```

Stop playback.

```
!CommandMeasure "MeasureMediaKey" "PlayPause"
```

Plays/Pause playback.

```
!CommandMeasure "MeasureMediaKey" "VolumeMute"
```

Mutes the sound.

```
!CommandMeasure "MeasureMediaKey" "VolumeDown"
```

Decreases volume.

```
!CommandMeasure "MeasureMediaKey" "VolumeUp"
```

Increases volume.

Example

```
[MeasureMediaKey]

Measure=Plugin

Plugin=MediaKey
```

```
[MeterMute]
Meter=String
SolidColor=0,0,0,255
FontColor=255,255,255,255
Text=Mute
LeftMouseUpAction=!CommandMeasure "MeasureMediaKey" "VolumeMute"
```

Select all

NowPlaying plugin [home]

`Plugin=NowPlaying` retrieves information about the currently playing track from a number of media players.

Options

General measure options

All general measure options are valid.

PlayerName

Can be either:

- The player interface name (e.g. `PlayerName=iTunes`)
- Or the name of another measure (e.g. `PlayerName=[MainMeasure]`)

Important: If multiple measures use the same player, specify the player interface name on the first NowPlaying measure and specify the name of the first measure in subsequent measures. See example below for proper usage.

PlayerType

Type of the measure value. Valid values are:

- `Artist`: Track artist.
- `Album`: Current album.
- `Title`: Track title.
- `Number`: Track number.
- `Year`: Track year.
- `Cover`: Path to cover art.
- `File`: Path to the playing media file.
- `Duration`: Total length of track in seconds.
- `Position`: Current position in track in seconds.
- `Progress`: Percentage of track completed.
- `Rating`: Rating of current track (0 to 5).
- `Repeat`: 0 if repeat/loop track is off, 1 if on.
- `Shuffle`: 0 if shuffle/random tracks is off, 1 if on.
- `State`: 0 for stopped, 1 for playing, and 2 for paused.
- `Status`: 0 for inactive (player closed) and 1 for active (player open).
- `Volume`: From 0 to 100.

Note: With measures of type `Duration` or `Position`, the string value is in the form `MM:SS` and the number value is the actual number of seconds.

PlayerPath

If defined, used to launch the player with the `OpenPlayer` command. If not defined, the plugin will attempt to automatically detect the path.

TrackChangeAction

Action to execute when the track changes.

DisableLeadingZeroDefault: `0`

If set to `1`, the format of Duration and Position is `M:SS` instead of `MM:SS`. This option must be set on the main measure.

Bangs

NowPlaying measures can be controlled with the !CommandMeasure bang with the argument parameter being:

- `Pause`: Pause current track.
- `Play`: Play current track.
- `PlayPause`: Play (if stopped/paused) or pause (if playing) current track.
- `Stop`: Stop current track.
- `Next`: Change to next track.
- `Previous`: Change to previous track.
- `OpenPlayer`: Opens the player. If already open, the player will be brought to the top.
- `ClosePlayer`: Closes the player.
- `TogglePlayer`: Opens/closes the player depending on current state.
- `SetPosition n`: Where *n* is either an absolute value (`SetPosition 50` to jump to 50% of the track) or a relative value (`SetPosition +5` to jump 5% forward or `SetPosition -10` to jump 10% backward).
- `SetRating n`: Where *n* is a value between `0` (no rating) and `5` (maximum rating).
- `SetShuffle n`: Where *n* is `1` (shuffle on), `0` (shuffle off), or `-1` (toggle shuffle).
- `SetRepeat n`: Where *n* is `1` (repeat on), `0` (repeat off), or `-1` (toggle repeat).
- `SetVolume n`: Where *n* is either an absolute value (`SetVolume 50` to set volume to 50%) or a relative value (`SetVolume +20` to increase volume by 20% or`SetVolume -40` to decrease volume by 40%).

Fully supported players

The following players are fully supported. All features should work unless stated otherwise.

- **AIMP**: `PlayerName=AIMP`
  Fully supported. Tested with AIMP 2.61.
- **foobar2000**: `PlayerName=CAD`
  Fully supported. The foo_cad plugin (download) needs to be installed.
- **iTunes**: `PlayerName=iTunes`
  Fully supported. Tested with iTunes 10.2.
- **J. River Media Center** and **Media Jukebox**: `PlayerName=CAD`
  Fully supported through the CAD interface. The intcad plugin needs to be installed.
- **MediaMonkey**: `PlayerName=MediaMonkey`
  Fully supported. Tested with MediaMonkey 3.2.5.

- **MusicBee**: `PlayerName=CAD`
  Fully supported. MusicBee 1.2 (or higher) is required.
- **Winamp**: `PlayerName=Winamp`
  Fully supported.
- **VLC**: `PlayerName=CAD`
  Fully supported. The libcad plugin for VLC must be installed.
- **WMP**: `PlayerName=WMP`
  Fully supported, except for the `Repeat` / `Shuffle` types.

Partially supported players

The following players are partially supported. Only some features will work.

- **Spotify**: `PlayerName=Spotify`
  Partially supported. Only the types `Artist`, `Track` and the bangs `Play`, `PlayPause`, `Stop`, `Next`, and`Previous` are available.
- **Last.fm Client**, **Media Player Classic**, **TTPlayer**, **OpenPandora**, **Zune**: `PlayerName=WLM`
  Partially supported. Even in the best case, only the types `Title`, `Artist`, `Album` and the bangs `Play`,`Pause`, `PlayPause`, `Next`, `Previous`, `Stop` are supported.

  Note: In **Media Player Classic**, '*Send Now Playing information to MSN Messenger*' option must be enabled in the player's settings (View -> Options -> Tweaks).

Example

For a more complete example, check the Soita skin.

```
[Rainmeter]

Update=1000

BackgroundMode=2

SolidColor=0,0,0,255


; MeasurePlayer is the main measure.

; MeasureArtist and MeasureAlbum are secondary measures.


[MeasurePlayer]

Measure=Plugin

Plugin=NowPlaying
```

```
; The main measure specifies the media player on PlayerName.
PlayerName=iTunes
PlayerType=TITLE
; PlayerPath, TrackChangeAction, and DisableLeadingZero are valid here on
; the main measure only.

[MeasureArtist]
Measure=Plugin
Plugin=NowPlaying
; Secondary measures specify the name of the main measure on PlayerName.
PlayerName=[MeasurePlayer]
PlayerType=ARTIST

[MeasureAlbum]
Measure=Plugin
Plugin=NowPlaying
PlayerName=[MeasurePlayer]
PlayerType=ALBUM

[MeterPrev]
Meter=String
X=5
Y=5
FontColor=FFFF00
Text="Prev"
LeftMouseUpAction=[!CommandMeasure "MeasurePlayer" "Previous"]

[MeterNext]
Meter=String
X=20R
```

```
Y=5
FontColor=FFFF00
Text="Next"
LeftMouseUpAction=[!CommandMeasure "MeasurePlayer" "Next"]

[MeterTitle]
Meter=String
MeasureName=MeasurePlayer
X=5
Y=35
W=400
H=20
FontColor=255,255,255,255
Text="Title: %1"

[MeterArtist]
Meter=String
MeasureName=MeasureArtist
X=5
Y=55
W=400
H=20
FontColor=255,255,255,255
Text="Artist: %1"

[MeterAlbum]
Meter=String
MeasureName=MeasureAlbum
X=5
Y=75
```

```
W=400

H=20

FontColor=255,255,255,255

Text="Album: %1"
```

Select all

PerfMon plugin [home]

`Plugin=PerfMon` retrieves infromation from the Windows Performance Monitor.

The Performance Monitor exposes several counters, which track various information. All available counters can be viewed using the Performance Monitor application. To open it, either run PerfMon.msc or type it in the Start menu.

Note: The names of the objects, counters and instances must be in English.

Options

General measure options

> All general measure options are valid.

PerfMonObject

Name of the performance object.

Examples: `Memory`, `Processor`

PerfMonCounter

Name of the performance counter.

Examples: `% Processor Time`, `Disk Read Bytes/sec`

PerfMonInstance

Name of the counter instance. Can be left unspecified if the counter does not have any instances.

Example: `_Total`

PerfMonDifferenceDefault: 1

If set to `0`, the current value of the counter is used as the measure value. If set to `1`, the difference between the current and previous counter value is used instead. This is useful as many of the performance values are counters and you usually want to know how much the counter has increased since last update.

Example

```
[Rainmeter]
Update=1000
BackgroundMode=2
SolidColor=0,0,0,255

[MeasureProcessCount]
; Measures the number of processes running.
Measure=Plugin
Plugin=PerfMon
PerfMonObject=System
PerfMonCounter=Processes
```

```
PerfMonDifference=0


[MeasureDiskAccess]
; Measures disk access of the C: drive.
Measure=Plugin
Plugin=PerfMon
PerfMonObject=LogicalDisk
PerfMonCounter=Disk Bytes/sec
PerfMonInstance=C:


[MeterProcessCount]
Meter=String
MeasureName=MeasureProcessCount
X=5
Y=5
W=200
H=20
FontColor=255,255,255,255
Text=Total processes: %1


[MeterDiskAccess]
Meter=String
MeasureName=MeasureDiskAccess
X=5
Y=25
W=200
H=20
FontColor=255,255,255,255
AutoScale=1
Text=Disk access: %1B/sec
```

Select all

Ping plugin [home]

`Plugin=PingPlugin` measures the latency between the computer and a host in milliseconds.

Options

General measure options

All general measure options are valid.

`DestAddress`

The address (e.g. `example.com`) or the IP (e.g. `192.168.0.1`) of the server.

`UpdateRate`Default: 32

Frequency at which the ping is measured. This is relative to the measure update frequency.

`Timeout`Default: `30000`

Maximum time in milliseconds to wait for a reply from the server.

TimeoutValueDefault: 30000

Used as the value of the measure when a timeout occurs.

FinishAction

Action to execute as soon as a successful value is returned, or when the number of milliseconds set in theTimeout option is reached.

Example

```
[Rainmeter]
Update=1000
BackgroundMode=2
SolidColor=0,0,0,255

[MeasurePing]
Measure=Plugin
Plugin=PingPlugin
DestAddress=www.google.com

[MeterPing]
Meter=String
MeasureName=MeasurePing
X=5
Y=5
W=200
H=20
FontColor=255,255,255,255
Text=google.com: %1ms
```

Select all

Power plugin [home]

`Plugin=PowerPlugin` measures power related information.

Options

General measure options

All general measure options are valid.

`PowerState`

Type of information to measure. Valid values are:

- `ACLine`: 0 for battery or 1 when plugged in.
- `Status`: 0 for no battery, 1 for charging, 2 for critical, 3 for low, or 4 for high.
- `Status2`: Equal to BatterFlag in SYSTEM_POWER_STATUS.
- `Lifetime`: Battery lifetime.
- `Percent`: Battery lifetime as a percentage.
- `Hz`: Current CPU frequency in Hz.
- `MHz`: Current CPU frequency in MHz.

Note: Battery information may not be available with some laptops.

Format Default: `%H:%M`

Format of the time with `PowerState=Lifetime`. The syntax is the same as Time measures.

Example

```
[Rainmeter]
Update=1000
BackgroundMode=2
SolidColor=0,0,0,255

[MeasureBatteryPercent]
Measure=Plugin
Plugin=PowerPlugin
PowerState=Percent

[MeterBatteryPercent]
Meter=String
MeasureName=MeasureBatteryPercent
X=5
Y=5
W=200
H=20
FontColor=255,255,255,255
Text=Battery left: %1
```

Select all

Process plugin [home]

`Plugin=Process` measures the running state of a process.

The measure's value will be:

- `1` : The specified `ProcessName` is running.
- `-1` : The specified `ProcessName` is not running.

Options

General measure options

All general measure options are valid.

`ProcessName`

Name of the process (e.g. `Firefox.exe`).

Example

```
[Rainmeter]

Update=1000
```

```
BackgroundMode=2

SolidColor=0,0,0,255


[MeasureProcess]

Measure=Plugin

Plugin=Process

ProcessName=Notepad.exe

Substitute="-1":"not running","1":"running"


[MeterProcess]

Meter=String

MeasureName=MeasureProcess

X=5

Y=5

W=200

H=20

FontColor=255,255,255,255

Text=Notepad.exe is %1.
```

Select all

Quote plugin [home]

`Plugin=QuotePlugin` Retrieves a **random** value from one of two kinds of resources:

- A random string value (by default a single line) from a text file.
- A random file path and and name selected from a folder.

Options

General measure options

All general measure options are valid.

`PathName`

Path of a folder or a file.

- If the specified path is a file, the measure value is a random part of a text file and the `Separator` option is valid for use.
- If the specified path is a folder, the measure value is a the path and name of a random file in the folder and the `Subfolders` and `FileFilter` options are valid for use.

`Separator`Default: `#CRLF#`

Delimiter used to separate text. By default, the delimiter is a newline.

Subfolders Default: 1

If set to `1`, subfolders are taken into account.

FileFilter

If specified, only file names matching a semicolon `;` delimited list of filters are taken into account. The standard wildcard characters `*` and `?` can be used.

Example: `*.jpg;*.gif`

Example

```
[Rainmeter]
Update=1000
BackgroundMode=2
SolidColor=0,0,0,255

[MeasureQuote]
Measure=Plugin
Plugin=QuotePlugin
PathName=%HOMEDRIVE%%HOMEPATH%\My Documents\My Pictures\
Subfolders=1
FileFilter=*.jpg;*.gif

[MeterQuote]
Meter=Image
MeasureName=MeasureQuote
X=0
Y=0
W=200
H=200
```

Select all

RecycleManager plugin [home]

`Plugin=RecycleManager` monitors the state of the recycle bin.

Options

General measure options

All general measure options are valid.

`RecycleType`Default: `Count`

Type of information to measure. Valid values are:

- `Count`: Number of files in the recycle bin.
- `Size`: Total size of the recycle bin in bytes.

Bangs

RecycleManager measures can be controlled with the !CommandMeasure bang with the argument parameter being:

- `OpenBin`
- `EmptyBin`: Empties the bin after confirmation.
- `EmptyBinSilent`: Empties the bin silently without confirmation.

Example

```
[Rainmeter]
```

```
Update=1000
BackgroundMode=2
SolidColor=0,0,0,255

[MeasureBinSize]
Measure=Plugin
Plugin=RecycleManager
RecycleType=Size
UpdateDivider=5

[MeterBinSize]
Meter=String
MeasureName=MeasureBinSize
X=5
Y=5
W=200
H=20
FontColor=255,255,255,255
SolidColor=0,0,0,1
AutoScale=1
Text=Bin size: %1B
LeftMouseUpAction=!CommandMeasure MeasureBinSize OpenBin
RightMouseUpAction=!CommandMeasure MeasureBinSize EmptyBin
```

Select all

ResMon plugin [home]

`Plugin=ResMon` monitors the use of resources.

Options

General measure options

All general measure options are valid.

ResCountTypeDefault: `GDI`

Type of resource to measure. Valid values are:

- `GDI`: Number of GDI objects.
- `USER`: Number of USER objects.
- `Handle`: Number of open handles.
- `Window`: Number of open system window handles.

ProcessName

If specified, the resource use of the specified process is measured.

Example

```
[Rainmeter]

Update=1000

DynamicWindowSize=1

BackgroundMode=2

SolidColor=0,0,0,255
```

```
[MeasureSystemGDI]
Measure=Plugin
Plugin=Resmon
ResCountType=GDI

[MeasureRainmeterGDI]
Measure=Plugin
Plugin=Resmon
ResCountType=GDI
ProcessName="Rainmeter.exe"

[MeterOne]
Meter=String
MeasureName=MeasureSystemGDI
MeasureName2=MeasureRainmeterGDI
FontSize=12
FontColor=255,255,255,255
StringStyle=Bold
AntiAlias=1
Text=GDI Handles: %1 (Rainmeter: %2)
```

Select all

SpeedFan plugin [home]

`Plugin=SpeedFanPlugin` retrieves information from the SpeedFan application. The SpeedFan application **must**be running in the background.

Note: If the value of the measure is to be used in a meter which requires a percentage, then appropriate MinValueand/or MaxValue options must be added to the measure.

Options

General measure options

All general measure options are valid.

SpeedFanTypeDefault: `Temperature`

Type of information to measure. Valid values are:

- `Temperature`: Temperature in the unit defined by `SpeedFanScale`.
- `Fan`: Fan speed.
- `Voltage`

SpeedFanNumberDefault: `0`

The index retrieved from the Speedfan application starting with `0`. It is easiest to obtain these numbers from the Configure tab of the Speedfan application.

Example: To obtain the second temperature value with `SpeedFanType=Temperature`, specify`SpeedFanNumber=1`.

SpeedFanScaleDefault: `C`

Temperature unit for `SpeedFanType=Temperature`. Valid values are:

- `C`: Celsius.
- `F`: Fahrenheit.

- `K`: Kelvin.

Note: The unit in the SpeedFan application itself must be set to Celsius to obtain the correct value.

Example

```
[Rainmeter]
Update=1000
DynamicWindowSize=1
BackgroundMode=2
SolidColor=0,0,0,255

[MeasureGPUTemp]
Measure=Plugin
Plugin=SpeedFanPlugin
SpeedFanType=Temperature
SpeedFanNumber=0
SpeedFanScale=C

[MeasureFanSpeed]
Measure=Plugin
Plugin=SpeedFanPlugin
SpeedFanType=Fan
SpeedFanNumber=3

[MeterSpeedFan]
Meter=String
MeasureName=MeasureGPUTemp
MeasureName2=MeasureFanSpeed
FontSize=12
FontColor=255,255,255,255
StringStyle=Bold
```

```
AntiAlias=1

Text=GPU Temp: %1#CRLF#Fan Speed: %2
```

Select all

SysInfo plugin [home]

`Plugin=SysInfo` measures various information about your system.

Options

General measure options

All general measure options are valid.

SysInfoType

Piece of system information to retrieve.
String values (Can only be used with String meters):

- `COMPUTER_NAME`: The computers name as specified in your system settings.
- `USER_NAME`: The username of the current user.
- `SCREEN_SIZE`: The resolution of the primary display monitor in a string of the form "width x height".
- `WORK_AREA`: The size of the client area of the primary display monitor in a string of the form "width x height".
- `OS_VERSION`: The current version of Windows.
- `OS_BITS`: Returns whether your OS is 32 or 64 bit.
- `ADAPTER_DESCRIPTION`: The description of the adapter specified by `SysInfoData`.
- `NET_MASK`: The current net mask. `SysInfoData` specifies which net mask if there are multiple.
- `IP_ADDRESS`: The current IP address. `SysInfoData` specifies which IP if there are multiple. 0-999 for installed, 1000-1999 for active
- `GATEWAY_ADDRESS`: The current gateway address. `SysInfoData` specifies which adapter if there are multiple.
- `HOST_NAME`: The current host name.
- `DOMAIN_NAME`: The current domain.
- `DNS_SERVER`: The DNS server address.

Numeric Values (Can be used anywhere):

- `SCREEN_WIDTH`: The width, in pixels, of the monitor. `SysInfoData` specifies which monitor if there are multiple.

- `SCREEN_HEIGHT`: The height, in pixels, of the monitor. `SysInfoData` specifies which monitor if there are multiple.
- `VIRTUAL_SCREEN_TOP`: Y-Coordinate of the upper left corner of the virtual screen. Use `SysInfoData` to specify the monitor. These coordinates are relative to the primary monitor and can be negative.
- `VIRTUAL_SCREEN_LEFT`: X-Coordinate of the upper left corner of the virtual screen. Use `SysInfoData` to specify the monitor. These coordinates are relative to the primary monitor and can be negative.
- `VIRTUAL_SCREEN_WIDTH`: Width of the virtual screen which encompasses all display monitors.
- `VIRTUAL_SCREEN_HEIGHT`: Height of the virtual screen which encompasses all display monitors.
- `WORK_AREA_TOP`: Y-Coordinate of the upper left corner of the client area. Use `SysInfoData` to specify the monitor. These coordinates are relative to the primary monitor and can be negative.
- `WORK_AREA_LEFT`: X-Coordinate of the upper left corner of the client area. Use `SysInfoData` to specify the monitor. These coordinates are relative to the primary monitor and can be negative.
- `WORK_AREA_WIDTH`: Width of the client area of the primary display monitor. Use `SysInfoData` to specify a different monitor, if there are several.
- `WORK_AREA_HEIGHT`: Height of the client area of the primary display monitor. Use `SysInfoData` to specify a different monitor, if there are several.
- `NUM_MONITORS`: The number of display monitors currently active.

`SysInfoData`

Additional data that some of the SysInfoType items require. This is always a number. For SysInfoTypes which use SysInfoData to specify a monitor, the appropriate value to pass can be found by opening the Windows Display Properties dialog. It is important to note that "1" is not always the primary display monitor.

Example

```
[Rainmeter]
Update=1000
DynamicWindowSize=1
BackgroundMode=2
SolidColor=0,0,0,255

[MeasureOSVersion]
Measure=Plugin
Plugin=SysInfo
SysInfoType=OS_VERSION
UpdateDivider=-1
```

```
[MeasureScreenWidth]
Measure=Plugin
Plugin=SysInfo
SysInfoType=SCREEN_WIDTH
SysInfoData=1

[MeasureScreenHeight]
Measure=Plugin
Plugin=SysInfo
SysInfoType=SCREEN_HEIGHT
SysInfoData=1

[MeterOSVersion]
Meter=String
MeasureName=MeasureOSVersion
FontSize=12
FontColor=255,255,255,255
StringStyle=Bold
AntiAlias=1
Text=OS Version: %1

[MeterScreenSize]
Meter=String
MeasureName=MeasureScreenWidth
MeasureName2=MeasureScreenHeight
Y=2R
FontSize=12
FontColor=255,255,255,255
StringStyle=Bold
```

```
AntiAlias=1

Text=Screen Size: %1 x %2
```

Select all

Virtual Desktops plugin [home]

`Plugin=VirtualDesktops` gives access to information from different Virtual Desktop Managers (VDMs) for Windows and can be used to trigger desktop switches.

Options

General measure options

> All general measure options are valid.

VDManager Valid values: **Dexpot** or **VirtuaWin**

> Must be set to the name of a supported VDM. Only Dexpot and VirtuaWin are currently supported.

VDMeasureType

> Specifies what information to return. These options are supported by all VDMs:
>
> - `VDMActive`: Returns 1 if the VDM is running, 0 otherwise.
> - `CurrentDesktop`: The number of the active desktop.
> - `SwitchDesktop`: Used with the `!CommandMeasure` bang to switch the active desktop.
> - `DesktopCount`: The total number of virtual desktops.

VDDesktopCount

> When `VDMeasureType=DesktopCount`, this setting can be set to either X or Y to return the number of columns or rows, respectively, in a desktop grid layout. See the example below for more information.

Bangs

`!CommandMeasure "DexpotSwitchDesktop" "2"` where *DexpotSwitchDesktop* is the section name containing `VDMeasureType=SwitchDesktop`. See the example below for more information.

Example

```
[VirtuaWinCurrentDesktop]

Measure=Plugin

Plugin=VirtualDesktops

VDManager=VirtuaWin

VDMeasureType=CurrentDesktop
```

```
[VirtuaWinDesktopCountX]
Measure=Plugin
Plugin=VirtualDesktops
VDManager=VirtuaWin
VDMeasureType=DesktopCount
VDDesktopCount=X


[DexpotSwitchDesktop]
Measure=Plugin
Plugin=VirtualDesktops
VDManager=Dexpot
VDMeasureType=SwitchDesktop
```

Select all

Dexpot Options `VDManager=Dexpot`

`VDOnActivate`

A bang that is executed when Dexpot starts. Used with `VDMeasureType=VDMActive`.

`VDOnDeactivate`

A bang that is executed when Dexpot closes. Used with `VDMeasureType=VDMActive`.

`VDOnChange`

A bang that is executed when the current desktop changes (when used with `VDMeasureType=CurrentDesktop`), or when the number of desktops changes (when used with `VDMeasureType=DesktopCount`).

`VDMeasureType`

Additional measure types are supported by Dexpot. These are:

- `Command`: Can be used to send commands to Dexpot through bands. Possible commands are the Dexpot command line options, listed here.
- `DesktopName`: The name of the active desktop. (Can be forced to a specific desktop by using VDDesktop.)

- `DesktopWallpaper`: The file path of the active desktop. (Can be forced to a specific desktop by using `VDDesktop`.)
- `Screenshot`: Writes a `.bmp` screenshot of the current desktop to the file specified by `VDOutputFile` after each desktop switch. (Can be forced to a specific desktop by using `VDDesktop`.)

VDDesktop

Set to the number of the desktop which should be used in place of the active one for the `DesktopName`, `DesktopWallpaper` and `Screenshot` MeasureTypes.

Options specific to `VDMeasureType=Screenshot`

VDOutputFile

The file to save the screenshot to.

VDWidth

Defines the width of the screenshot in pixels. Will autoscale if only height is given.

VDHeight

Defines the height of the screenshot in pixels. Will autoscale if only width is given.

VDRefreshOnUpdate

If `VDRefreshOnUpdate=1` and `VDDesktop` denotes the currently active desktop, creates a new screenshot every time the measure is updated.

Bangs

- `!CommandMeasure "DexpotCommand" "-next"`: Switches to the next desktop
- `!CommandMeasure "DexpotCommand" "-prev"`: Switches to the previous desktop.
- `!CommandMeasure "DexpotCommand" "-V"`: Open full-screen preview.
- `!CommandMeasure "DexpotCommand" "-d"`: Open window catalog.

DexpotCommand is the section name containing `VDMeasureType=Command`. See the example below for more information.

Example

```
[Screenshot]

Measure=Plugin

Plugin=VirtualDesktops
```

```
VDManager=Dexpot

VDMeasureType=Screenshot

VDDesktop=1

VDOutputFile=#CURRENTPATH#Desktop1.bmp

VDWidth=320


[DexpotCommand]

Measure=Plugin

Plugin=VirtualDesktops

VDManager=Dexpot

VDMeasureType=Command
```

Select all

WebParser plugin [home]

`Plugin=WebParser` reads and parses information from web pages.

The plugin uses Perl Compatible Regular Expressions to extract information from any web page or local file.

Usage

WebParser measures take the form:

```
[MeasureParent]
Measure=Plugin
Plugin=WebParser
URL=http://SomeSite.com
RegExp="(?siU)&lt;Item&gt;(.*)&lt;/Item&gt;.*&lt;Item&gt;(.*)&lt;/Item&gt;"
```

Select all

This example creates two StringIndex values in what is referred to as the "parent" WebParser measure. The information is generally used in subsequent "child" WebParser measures:

```
[MeasureChild1]
Measure=Plugin
Plugin=WebParser
URL=[MeasureParent]
StringIndex=1

[MeasureChild2]
Measure=Plugin
Plugin=WebParser
URL=[MeasureParent]
StringIndex=2
```

Select all

The values of the two child measures are now the information parsed into StringIndexes 1 and 2 by the parentmeasure. These can then be used with MeasureName and other options in meters.

Note: More information and examples for WebParser can be found at WebParser Tutorial and RSS/Atom Feed Tutorial.

Options

General measure options

All general measure options are valid.

URL

URL to the site or file to be downloaded and parsed. If the name of another WebParser measure is used, e.g.`URL=[SomeMeasure]`, then the value of the parent measure is used, generally by referring to a specificStringIndex number.

WebParser **cannot** use cookies or other session-based authentication, so it **cannot** be used to retrieve information from web sites requiring a login. However, Webparser **can** be used on sites which support HTTP authentication. E.g. `http://myname:mypassword@somesite.com`.

WebParser can read and parse local files on your computer by using the `file://` URI scheme. E.g.`URL=file://#CURRENTPATH#SomeFile.txt`.

If you want to use the current value of a measure in a dynamic way as a Section Variable, rather than as a reference to a "parent" WebParser measure, you must prefix the name of the measure with the `&` character.

```
URL=http://SomeSite.com\[&WebMeasure]
```

RegExp

The Perl compatible regular expression used in parsing.

FinishAction

A bang or action that is executed when the page has been downloaded and the parsing is done. This option is only valid on measures that connect to a site or file with `URL`, and not on child measures.

StringIndex

Defines which captured string from the `RegExp` this measure returns. This option is generally used in a childmeasure to determine which of the caputured values in a parent measure to use.

StringIndex2

The second string index is used when using a `RegExp` in a measure that uses data from another WebParser measure (i.e. the `URL` points to a parent measure. In this case the `StringIndex` defines

the index of the result of the parent measure's `RegExp` and the `StringIndex2` defines the index of this measure's `RegExp` (i.e. it defines the string that the measure returns).

More information on using StringIndex2 can be found here.

Note: If the `RegExp` is not defined in this measure, the `StringIndex2` has no effect.

`UpdateRate`Default: `600`

The rate in milliseconds determining how often the webpage is downloaded. This is relative to the config's mainUpdate rate **and** any UpdateDivider on the measure. So the formula would be `Update` X `UpdateDivider` X`UpdateRate` = "how often the measure connects to the site".

Notes: Some caution should be used in determining how often to connect to a site with WebParser. Excessively accessing a site can cause your computer to be seen as an "attack" and result in being blocked. The`UpdateRate` option defaults to `600` as a safety measure. This should not be changed unless there is some reason to connect more or less often to the site.

In order to override the UpdateRate set on a WebParser measure, to have it connect to the site and download the data "right now", the !CommandMeasure bang must be used, with the name of the "parent" measure as the first parameter, and "Update" as the second.

```
LeftMouseUpAction=[!SetOption WebMeasure URL "http://SomeNewSite.com"][!CommandMeasure WebMeasure Update]
```

`DecodeCharacterReference`Default: `0`

Automatically decodes HTML Character References. This will eliminate the need to use a `Substitute`statement to translate character references like `&quot;`, `&amp;`, `&lt;`, and `&gt;` to the actual character. Valid values are:

- `0`: Does nothing (default).
- `1`: Decodes both numeric character references and character entity references.
- `2`: Decodes only numeric character references.
- `3`: Decodes only character entity references.

`Debug`Default: `0`

Logs DEBUG messages to the Rainmeter log or to a file. Valid values are:

- `0`: Does not log DEBUG messages from WebParser.
- `1`: Logs DEBUG message to the log. Rainmeter must also be in Debug mode.
- `2`: Saves the downloaded webpage to `WebParserDump.txt` in the current skin folder. This can be useful since some web servers send different information depending which client requests it. Remember to remove this from your config once you have it working correctly.

Hint: Determining StringIndex values to use in a child measure can be done by setting `Debug=1` on a measure having the `RegExp` option, which will display matched strings and StringIndex numbers in the Rainmeter log

`Debug2File`

If the `Debug` option is set to `2`, this option can be set to the path and name of the file to use for the downloaded webpage instead of `WebParserDump.txt` in the current skin folder.

Note: The folder for the file must already exist.

`Download`Default: `0`

If `Download=1`, the `URL` is downloaded to Window's `TEMP` folder and the name to the file is returned as string value. The measure can then be used with MeasureName on an Image meter to download images from a site and display them.

`DownloadFile`

If the `Download` option is set to `1`, this option defines a relative path and file name where the downloaded file will be saved instead of in Windows `TEMP`.

A folder `DownloadFile` will be created in the current folder, and the defined relative path and file name will be created under that. It is not possible to specify an absolute path.

Note: This file is **not** a temporary file so it is **not** deleted after unloading a skin or exiting Rainmeter.

`ErrorString`

The value of the measure will be set to the string defined in this option if the `RegExp` results in a regular expression parsing error.

`ForceReload`Default: `0`

WebParser reads the resource only if it has been modified since last read. This can be overridden with`ForceReload=1`.

`ProxyServer`Default: `/auto`

Proxy server to use with the plugin. The following settings are valid:

- **/auto**
  This will use the proxy settings contained in the options for Internet Explorer. (default)
- **/none**
  This will make a direct connection, and will not use any proxy setting.

- **ServerName:Port**
  This will connect to the proxy server hostname or ip address and port defined. Port is often optional with proxy servers.

This option can also be set in the Rainmeter.data file. If set there, it will be used as the global setting for all WebParser measures unless overridden in an individual measure(s).

Note: The plugin doesn't support any authentication, so **only** use proxy settings that do not require it.

Examples: `ProxyServer=/none`, `ProxyServer=192.168.1.1:8080`, `ProxyServer=ProxyHostname.net`

CodePageDefault: `0`

Defines the code page of the downloaded URL=http:// web page or external file read with URL=file://.

Most web sites on the web today are encoded with the Unicode UTF-8 standard. This is the default for WebParser, and it will seamlessly handle the site. No CodePage option is needed.

However, there may be some older web sites that are encoded in a language / character set specific way. On a web site, the encoding used can generally be determined by viewing the raw HTML source and checking the "charset" meta value in the "head" section of the page. (i.e. meta charset="UTF-8")

Some Examples are:

- `CodePage=1200` : Unicode UTF-16 LE (Little Endian)
- `CodePage=1251` : ANSI Cyrillic; Cyrillic (Windows)
- `CodePage=1252` : ANSI Latin 1; Western European (Windows)
- `CodePage=28605` : ISO 8859-15 Latin 9
- `CodePage=65001` : Unicode UTF-8

In addition, there are times when an external local file to be parsed with URL=file:// will be encoded in other than the ANSI (really ASCII plus "extended ASCII" specific to the locale of the computer) encoding used as the default in most Windows-based text editors. Primarily this will be in Unicode UFT-16 LE. In this case, the`CodePage=1200` option must be used to tell WebParer how to interpret the resource being read.

Codepage definitions and more information can be found at Windows code pages.

Additional general help with Unicode encoding in Rainmeter can be found at Character Encoding in Rainmeter.

WebParser and Dynamic Variables

Dynamic variables can be used with the WebParser plugin. There are some things specific to WebParser that should be kept in mind when doing things in a dynamic way in WebParser measures:

WebParser uses UpdateRate to determine how often the plugin should actually access the site or file. While you can dynamically change any option on a WebParser measure, the plugin will not use the changes and access the site again until the next UpdateRate is reached. Just using !Update or !UpdateMeasure will NOT override the UpdateRate.

In order to have a dynamic change make WebParser parse the site "right now", you will use the!CommandMeasure bang with the "parent" WebParser measure as the first parameter, and "Update" as the second.

```
LeftMouseUpAction=[!SetOption WebMeasure URL "http://SomeNewSite.com"][!CommandMeasure WebMeasure Update]
```

If you dynamically change an option on a "child" WebParser measure that depends on a "parent" measure, (like StringIndex for instance) you MUST use !CommandMeasure with "Update", targeting the "parent" WebParser measure. The values of child WebParser measures are a function of the parent measure, and are only updated when the parent is. You should never use !CommandMeasure on a "child" measure.

If you want to use the current value of a measure in a dynamic way as a Section Variable, rather than as a reference to a "parent" WebParser measure, you must prefix the name of the measure with the `&` character.

```
URL=http://SomeSite.com\[&WebMeasure]
```

Examples

Retrieve the site title, first item and link from Slashdot's RSS feed.

```
[Rainmeter]

Update=1000

DynamicWindowSize=1

BackgroundMode=2

SolidColor=0,0,0,255


[MeasureRSSParent]

Measure=Plugin

Plugin=WebParser

URL=http://slashdot.org/slashdot.rdf
```

```
RegExp="(?siU)&lt;title&gt;(.*)&lt;/title&gt;.*&lt;item&gt;.*&lt;title&gt;(.*)&lt;/title&gt;.*
&lt;link&gt;(.*)&lt;/link&gt;"


[MeasureRSSTitle]

Measure=Plugin

Plugin=WebParser

URL=[MeasureRSSParent]

StringIndex=1


[MeasureRSSItemTitle]

Measure=Plugin

Plugin=WebParser

URL=[MeasureRSSParent]

StringIndex=2


[MeasureRSSItemLink]

Measure=Plugin

Plugin=WebParser

URL=[MeasureRSSParent]

StringIndex=3


[MeterRSSTitle]

Meter=String

MeasureName=MeasureRSSTitle

FontSize=14

FontColor=222,255,227,255

StringStyle=Bold

AntiAlias=1


[MeterRSSItemTitle]

Meter=String
```

```
MeasureName=MeasureRSSItemTitle

Y=2R

FontSize=11

FontColor=255,255,255,255

StringStyle=Bold

AntiAlias=1

LeftMouseUpAction=["[MeasureRSSItemLink]"]

DynamicVariables=1
```

Select all

Retrieve the title, download and display an image for the first item in the Customize.org RSS feed.

```
[Rainmeter]

Update=1000

DynamicWindowSize=1


[MeasureCustoParent]

Measure=Plugin

Plugin=WebParser.dll

URL=http://customize.org/feeds/submissions

RegExp="(?siU).*&lt;item&gt;.*&lt;title&gt;(.*)&lt;/title&gt;.*&lt;description&gt;.*src="(.*)"
.*&lt;/description&gt;.*&lt;link&gt;(.*)&lt;/link&gt;"


[MeasureTitle]

Measure=Plugin

Plugin=WebParser.dll

URL=[MeasureCustoParent]

StringIndex=1


[MeasureImage]

Measure=Plugin

Plugin=WebParser.dll
```

```
URL=[MeasureCustoParent]

StringIndex=2

Download=1


[MeasureLink]

Measure=Plugin

Plugin=WebParser.dll

URL=[MeasureCustoParent]

StringIndex=3


[MeterTitle]

Meter=String

MeasureName=MeasureTitle

FontSize=12

FontColor=255,255,255,255

SolidColor=0,0,0,255

AntiAlias=1


[MeterImage]

Meter=Image

MeasureName=MeasureImage

Y=2R

W=80

H=60

PreserveAspectRatio=2

LeftMouseUpAction=["[MeasureLink]"]

DynamicVariables=1
```

Select all

WiFiStatus plugin [home]

`Plugin=WiFiStatus` can be used to display various attributes of visible wireless networks in your area.

Windows XP SP2 users need to install the KB918977 hotfix.

Options

General measure options

All general measure options are valid.

`WiFiInfoType`

This defines what is measured. This option is **required**. Valid values are:

- `SSID`: The broadcast name for your current connection. When trying to connect, you may see a 'connecting...' or 'authorizing...' next to the name, but only if your update speed is around 2 seconds.
- `Quality`: Percentage value of the maximum dBm signal strength for your current connection.
- `Encryption`: The cipher algorithm being used for your current connection. Possible values are: `NONE`,`WEP40`, `TKIP`, `AES`, `WEP104`, `WPA_GROUP`, and `WEP`. An unknown algorithm will return `???`.
- `AUTH`: The authentication algorithm being used for your current connection. Possible values are: `Open`,`Shared`, `WPA-NONE`, `WPA-Enterprise`, `WPA-Personal`, `WPA2-Enterprise`, and `WPA2-Personal`. An unknown algorithm will return `???`.
- `PHY`: The supported bands for your connection (**not** your adapter). Possible values are `802.11a`,`802.11b`, `802.11g`, `802.11n`, `DSSS`, `FHSSS`, and `IR-Band`. An unknown band will return `???`. **Note:**The `PHY` option is only supported in Windows Vista and higher.
- `List`: Returns a list of all visible networks. The list is automatically separated by line-breaks `\n`, so each network appears on a separate line. It will also display the `PHY`, `Encryption`, and `AUTH` algorithms next the the network `SSID`.

`WiFiIntfID`Default: `0`

Only use this if you have more than 1 wireless interface adapter (which may include any bluetooth or infrared receivers). If you do not get any response from the measure with this value at `0`, then try other values starting from `1`, then `2` and so on.

`WiFiListStyle`Default: `0`

This allows you to control what information appears along with the names of all visible networks when`WiFiInfoType=List`. Valid values are:

- `0`: SSID.
- `1`: SSID @band.
- `2`: SSID (Encryption:Authentication).
- `3`: SSID @band (Encryption:Authentication).

WiFiListLimitDefault: 5

Allows you to control the number of networks that are will be displayed when used with `WiFiInfoType=List`.

Note: The list is sorted in descending order based on the signal quality of each network (i.e. strongest first, weakest last).

Example

```
[MeasureSSID]
Measure=Plugin
Plugin=WiFiStatus
WiFiInfoType=SSID
WiFiIntfID=0

[MeasureNetworks]
Measure=Plugin
Plugin=WiFiStatus
WiFiInfoType=LIST
WiFiIntfID=0
WiFiListStyle=3
```

Select all

Win7Audio plugin [home]

`Plugin=Win7AudioPlugin` controls the sound device and volume for Windows Vista and newer. Previous versions of Windows are not supported.

Returns the name of the current sound device (when used in a String meter) and the percentage (0-100) of current volume level (when used in a meter which requires a percentage).

Options

General measure options

> All general measure options are valid.

Usage:

```
[MeasureWin7Audio]

Measure=Plugin

Plugin=Win7AudioPlugin
```

Select all

Bangs

To control the sound device or volume, use the !CommandMeasure bang. In the examples below`MeasureWin7Audio` is the section name for the Win7Audio measure. Your section name may differ.

```
!CommandMeasure "MeasureWin7Audio" "ToggleNext"
```

> Jump to the first if last device is active.

```
!CommandMeasure "MeasureWin7Audio" "TogglePrevious"
```

> Jump to the last if first device is active.

```
!CommandMeasure "MeasureWin7Audio" "SetOutputIndex index"
```

> Set a specific device with *`index`*. This depends on your system setup and number of output devices.

```
!CommandMeasure "MeasureWin7Audio" "ToggleMute"
```

> Toggle the mute state.

```
!CommandMeasure "MeasureWin7Audio" "SetVolume x"
```

> Set volume to `x` (between 0 and 100). This disables mute.

```
!CommandMeasure "MeasureWin7Audio" "ChangeVolume x"
```

> Change the volume by `x` percent. Use negative numbers to lower volume. This disables mute.

Example

```
[Rainmeter]
BackgroundMode=2
SolidColor=0,0,0,150
Update=1000

; Returns the name of the current sound device and percent volume level
[MeasureWin7Audio]
Measure=Plugin
Plugin=Win7AudioPlugin

; Display the current sound device name
[MeterDeviceName]
Meter=String
MeasureName=MeasureWin7Audio
X=5
Y=0
W=200
H=20
FontFace=Trebuchet MS
FontSize=11
FontColor=255,255,255,255
StringAlign=LEFT
AntiAlias=1
```

```
ClipString=1


; Toggle to the next installed sound device (wraps back to first)
[MeterChangeDevice]
Meter=String
FontFace=Trebuchet MS
FontSize=15
FontColor=255,255,255,255
AntiAlias=1
X=220
Y=-3
Text="+"
LeftMouseUpAction=!CommandMeasure "MeasureWin7Audio" "ToggleNext"


; Change the volume down by 10%
[MeterVolumeDown]
Meter=String
FontFace=Trebuchet MS
FontSize=15
FontColor=255,255,255,255
AntiAlias=1
X=0
Y=20
Text="&lt;&lt;"
LeftMouseUpAction=!CommandMeasure "MeasureWin7Audio" "ChangeVolume -10"


[MeasureVolBarBackground]
Measure=Calc
Formula=1
[MeterVolBarBackground]
```

```
Meter=Bar
MeasureName=MeasureVolBarBackground
BarOrientation=Horizontal
BarColor=150,150,150,255
W=60
H=6
X=25
Y=29

; Displays the current volume percentage on a Bar meter
[MeterCurrentVolumeBar]
Meter=Bar
MeasureName=MeasureWin7Audio
BarOrientation=Horizontal
W=60
H=6
X=25
Y=29

; Change the volume up by 10%
[MeterVolumeUp]
Meter=String
FontFace=Trebuchet MS
FontSize=15
FontColor=255,255,255,255
AntiAlias=1
X=83
Y=20
Text="&gt;&gt;"
LeftMouseUpAction=!CommandMeasure "MeasureWin7Audio" "ChangeVolume 10"
```

```
[MeasureVolPercent]
Measure=Calc
Formula=MeasureWin7Audio

; Displays the percentage volume level as text
[MeterVolPercent]
Meter=String
MeasureName=MeasureVolPercent
X=55
Y=40
W=110
H=15
FontFace=Trebuchet MS
FontSize=10
FontColor=255,255,255,255
StringAlign=Center
AntiAlias=1
Text=%1%

; Toggles "mute" for the current device
[MeterVolToggleMute]
Meter=String
X=5
Y=60
FontFace=Trebuchet MS
FontSize=10
FontColor=255,255,255,255
StringAlign=Left
AntiAlias=1
```

```
Text=MUTE
LeftMouseUpAction=!CommandMeasure "MeasureWin7Audio" "ToggleMute"


; Set volume for current device to 50%
[MeterVolSet50%]
Meter=String
X=78
Y=60
FontFace=Trebuchet MS
FontSize=10
FontColor=255,255,255,255
StringAlign=Left
AntiAlias=1
Text=50%
LeftMouseUpAction=!CommandMeasure "MeasureWin7Audio" "SetVolume 50"
```

Select all

WindowMessage plugin [home]

`Plugin=WindowMessagePlugin` can be used to send and receive information from other applications. It can send window messages to other applications and show the result. For example, the plugin can be used to control WinAmp or some similar media players.

Options

General measure options

All general measure options are valid.

WindowName

The name of the window. This is used to identify the window. It's not necessary to set this if the `WindowClass` is set.

WindowClass

The class of the window. This is used to identify the window. It's not necessary to set this if the `WindowName` is set.

WindowMessage

This is the message to be send to the window. You need to define 3 parameters to where the first one is the `message` and the next ones are `wParam` and `lParam`. **Both wParam and lParam are unsigned decimal integers.** The measure returns the value returned by the SendMessage API function. If the `WindowMessage` is not given, the measure returns the window's current title.

Bangs

Messages can also be sent on demand with the !CommandMeasure bang. The arguments are similar that of the `WindowMessage` option. `!CommandMeasure "MeasureName" "SendMessage Msg wParam lParam"`

Examples

The following example returns the name of the song that is playing in WinAmp.

```
[MeasureWinamp]
Measure=Plugin
Plugin=WindowMessagePlugin
WindowClass=Winamp v1.x
Substitute="[Paused]":"","- Winamp":""
```

Select all

Check if Winamp is playing, or not. The following example returns `0` if it's not playing, `1` if it is playing, and `3` if it's paused.

```
[MeasureWinampPlaying]
Measure=Plugin
```

```
Plugin=WindowMessagePlugin
WindowClass=Winamp v1.x
WindowMessage=1024 0 104
```

Select all

Show song progress. In the following example, the `[MeasureWinampDuration]` will return the current song position as a value between `0` and `1`. Then you can use the Bar meter to display the value.

```
[MeasureWinampFull]
Measure=Plugin
Plugin=WindowMessagePlugin
WindowClass=Winamp v1.x
WindowMessage=1024 1 105

[MeasureWinampCurr]
Measure=Plugin
Plugin=WindowMessagePlugin
WindowClass=Winamp v1.x
WindowMessage=1024 0 105

[MeasureWinampDuration]
Measure=Calc
Formula=(MeasureWinampCurr&lt;=0)?0:(MeasureWinampCurr/(MeasureWinampFull*1000+1))
```

Select all

Options [home]

While Rainmeter has hundreds of options in total for all the measures, meters, and plugins, many of them can be categorized into types. For example, some options represent colors while others are used to specify a path.

Color options

Color options such as `SolidColor` and `FontColor` should use the RGBA (red-green-blue-alpha) notation in either the hexadecimal or decimal form. The color results from a mixing of the red, green and blue components of the option, with the transparency (alpha channel) of the element set by the fourth component.

Decimal colors are specified as `RRR,GGG,BBB,AAA`, where `RRR`, `GGG`, `BBB`, and `AAA` are decimal numbers from 0 to 255. Formulas can also be used in place of the numbers.

Hexadecimal colors are specified as `RRGGBBAA`, where `RR`, `GG`, `BB`, and `AA` are hexadecimal numbers from 00 to FF.

Note: The alpha component is optional. As with the other components it is a value from 0 to 255 (00 to FF), with 0 being completely invisible (and the meter will not react to the mouse) to 255, which is completely opaque. The default is 255 (or FF).

```
; The following lines are equivalent to solid opaque red:

SolidColor=255,0,0,255

SolidColor=255,0,0

SolidColor=(200 + 55),(2 - 2),0

SolidColor=FF0000FF

SolidColor=FF0000
```

Select all

One of many ways to obtain a desired color code is Rainmeter Online Color Picker.

Path options

Path options specify the relative or absolute path of either a file or a folder (depending on the option). For example, the path of an image file is expected with `ImageName`.

```
; Relative paths (relative to current folder):

ImageName=lolcat.png

ImageName=..\lolcat.png


; Absolute path:

ImageName=C:\lolcats\lolcat.png
```

Select all

Number options

Number options specify either a number or a formula enclosed in parentheses.

```
; The following lines are equivalent:

FontSize=42

FontSize=(40 + 2)

FontSize=(2 &gt; 1 ? 42 : 666)
```

Select all

Action options

Options in a skin which cause Rainmeter to take some action. Actions may be triggered by user input such asmouse actions, or system events and changing values within the skin, such as if actions.

Normally the value of an action option is going to be one or more bangs, or the execution of external commmands in Windows.

```
; Single bang:

LeftMouseUpAction=!HideMeter SomeMeter

LeftMouseUpAction=[!HideMeter SomeMeter]


; Multiple bangs:

LeftMouseUpAction=[!HideMeter SomeMeter][!HideMeter SomeOtherMeter]
```

Select all

For bangs that take parameters, the arguments should be separated by a space. Parameters that contain spaces must use quotes around the parameter.

```
; These two lines are equivalent:

LeftMouseUpAction=[!HideMeter SomeMeter]

LeftMouseUpAction=[!HideMeter "SomeMeter"]


; These two are NOT equivalent. The first line will cause an error due to the spaces

; in the parameters while the second will properly set the value of SomeVariable

; to: I think, therefore I am

LeftMouseUpAction=[!SetVariable SomeVariable I think, therefore I am]

LeftMouseUpAction=[!SetVariable SomeVariable "I think, therefore I am"]


; These two may or may not be equivalent depending on the value of the #ImageFile#

; variable. If #ImageFile# contains spaces, the first line will fail. In uncertain

; cases, it is always best to use quotes as in the second line.

LeftMouseUpAction=[!SetWallpaper #ImageFile#]

LeftMouseUpAction=[!SetWallpaper "#ImageFile#"]
```

Select all

External Windows commands can be executed by specifying the path to the executable and any parameters. Enclose any values with spaces in quotes.

```
LeftMouseUpAction=["C:\Windows\Notepad.exe" MyFile.txt]

; Runs Notepad.exe and loads the file MyFile.txt.


LeftMouseUpAction=["http://rainmeter.net/forums"]

; Opens the URL in the default web browser.
```

Select all

A parameter containing quotes should be surrounded by a pair of triple quotes.

```
; The first line will fail due to extra quotes. The second line will properly log

; the string: Quotes are "fun"!

LeftMouseUpAction=[!Log "Quotes are "fun"!"]

LeftMouseUpAction=[!Log """Quotes are "fun"!"""]
```

Select all

When using `#VarName#` or `[MeasureName]` in an action option, the current value of the variable or measure will be used. To have the literal string "#VarName#" or "[MeasureName]" used, in a !SetOption value for instance, use the "*" char as in `#*VarName*#` or `[*MeasureName*]` to tell Rainmeter you do not want the value resolved, but rather used as a literal string.

```
LeftMouseUpAction=!SetOption SomeMeter FontSize #VarName#

; The FontSize option for SomeMeter will be set to the current value of the variable.


LeftMouseUpAction=!SetOption SomeMeter FontSize #*VarName*#

; The FontSize option for SomeMeter will be set to the the string "#VarName#"
```

Select all

Regular expression options

These options use Perl Compatible Regular Expressions (PCRE) to match specific parts of a plaintext string. Regular expressions are used when the structure of a piece of data is known, but the content is not.

In Rainmeter, regular expressions are most prominently used by the WebParser plugin to interpret (or "parse") web-based data sources, but they can also be used to modify the value of any measure using Substitute options.

The PCRE library is maintained and documented by its own creators, not by any members of the Rainmeter development team. For more about using regular expressions, see the following resources:

- Documentation.
- Tutorial.
- Cheat sheet.

Formats that are commonly parsed with regular expressions include HTML, XML, RSS, Atom, JSON and CSV.

Bangs [home]

Bangs are commands that control various aspects of skins and Rainmeter. They are executed when used withaction options in the skin.

- Operating System bangs
- Application bangs
- Option and Variable bangs
- Skin bangs
- Meter bangs
- Measure bangs

Bangs can also be used from the Windows command line as a parameter to the Rainmeter.exe executable. Example: `"C:\Program Files\Rainmeter\Rainmeter.exe" !RefreshApp`.

Note: Many bangs have a `Config` parameter. Unless otherwise specified, valid values are the config name of a currently loaded skin to be acted upon or `*` (asterisk) to act on all currently loaded skins. When optional and not supplied, the parameter defaults to the current config. If executing a bang with a "config" parameter from the Windows command line, the parameter is always required.

Operating System bangs

!SetClip Parameter: **String**

Copies the specified string to the Windows clipboard. If the string contains spaces, use quotes around the parameter.

- `String` (required): The string to be copied to the clipboard.

Example: `!SetClip "This is copied to the clipboard!"`

!SetWallpaper Parameters: **File**, *Position*

Sets the Windows desktop background to the specified image file.

- `File` (required): The file to be set as the desktop background. The following formats are supported: `.bmp`, `.jpg`, `.png`, `.gif`, and `.tiff`. If the filename contains spaces, you must enclose the filename in quotes.
- `Position` (optional): Sets the position of the wallpaper. Valid values are: `Center`, `Tile`, `Stretch`. Windows 7 (and above) also have the values: `Fit` and `Fill`.

Example: `!SetWallpaper "Some Image.png" Center`

Application bangs

!About Parameter: *TabName*

Opens the About window.

- `TabName` (optional): Name of the tab to open. Valid values are: `Log` (default), `Skins`, `Plugins`, and `Version`. The value `Measures` is deprecated and will open the `Skins` tab.

Example: `!About Skins`

!Manage Parameters: *TabName*, *Config*, *File*

Opens the Manage window.

- `TabName` (optional): Name of the tab to open. Valid values are: `Skins` (default), `Layouts`, and `Settings`.
- `Config` (optional): Config name. If specified the list on the left will jump to and select the named config. If Config is specified, then TabName is required.
- `File` (optional): A skin .ini file in a named config. If specified the list on the left will jump to and select the named skin .ini file. If File is specified, then TabName and Config are required.

Example: `!Manage Skins "illustro\Clock" "Clock.ini"`

`!TrayMenu`

Opens the Rainmeter context menu at the current mouse location.

`!Log`Parameters: **`String`**, *`ErrorType`*

Writes a message to the log.

- `String` (required): The string to be written to the log. If the string contains spaces, enclose the string in quotes.
- `ErrorType` (optional): Specifies the type of error. Valid values are: `Notice` (default), `Error`, `Warning`, and `Debug`.

Example: `!Log "There was an error!" Error`

`!ResetStats`

Resets the network statistics.

`!LoadLayout`Parameters: **`LayoutName`**

Loads the named layout. A layout can also be loaded from the Windows command line:

Example: `"C:\Program Files\Rainmeter\Rainmeter.exe" !LoadLayout "My Saved Layout"`

`!RefreshApp`

Does a full refresh of all skins and reloads the list of configs and Rainmeter.ini settings. This is the same as "Refresh All" from the system tray context menu. The main difference from `!Refresh *` is that the skins folder is rescanned.

!Quit

Quits Rainmeter.

PlayParameter: **SoundFile**

Plays the given sound file **once**.

- `SoundFile` (required): File to be played. Must be a .wav file.

Example: `Play "SomeFile.wav"`

PlayLoopParameter: **SoundFile**

Plays the given sound file in a loop.

- `SoundFile` (required): File to be played. Must be a .wav file.

Example: `PlayLoop "SomeFile.wav"`

PlayStop

Stops the currently playing sound.

Option and Variable bangs

!SetOption Parameters: **Meter/Measure, Option, Value,** *Config*

Sets an option (ie. FontSize, Text, Formula, etc.) of a meter or measure.

- `Meter/Measure` (required): Name of the meter or measure section.
- `Option` (required): Name of the option to be changed.
- `Value` (required): New value to be set.
- `Config` (optional)

Details at !SetOption Guide.

Example: `!SetOption SomeStringMeter Text "New Text"`

!SetVariable Parameters: **Variable, Value,** *Config*

Sets a new value for a variable. The meter or measure where the variable is used must enableDynamicVariables. !SetVariable can create a new variable in memory, even if it is not pre-defined in the[Variables] section.

- `Variable` (required): Name of the variable.
- `Value` (required): New value to be set.
- `Config` (optional)

Example: `!SetVariable SomeVariable "New value!"`

!WriteKeyValue Parameters: **Section, Key, Value,** *FilePath*

Permanently writes a `Key=Value` pair below a section in a INI formatted file (such as .ini or .inc files). A skin must be refreshed for a new value written to the skin's .ini or .inc files to be re-read and used.

- `Section` (required): If the section does not exist in the file, a new section will be written at the end of the file.
- `Key` (required): If the key does not exist under the section, a new key will be written at the end of the section.
- `Value` (required): Value to be written. Any previous value will be overwritten.
- `FilePath` (optional): If not specified, the current skin file is used. The file must exist and must be located under either `#SKINSPATH#` or `#SETTINGSPATH#`.

Example: `!WriteKeyValue Variables MyFontName Arial "#@#Variables.inc"`

Option and Variable group bangs:

!SetOptionGroup Parameters: **Group, Option, Value,** *Config*

Sets an option (ie. FontSize, Text, Formula, etc.) of meters and measures in the specified group.

- `Group` (required): Name of the group.
- `Option` (required): Name of the option to be changed.
- `Value` (required): New value to be set.

- `Config` (optional)

Example: `!SetOptionGroup StringGroup Text "New text!"`

`!SetVariableGroup`Parameters: **`Variable, Value, Group`**

Changes the value of a variable in the configs of the specified group.

- `Variable` (required): Name of the variable.
- `Value` (required): New value to be set.
- `Group` (required): Name of the group.

Example: `!SetVariableGroup MyFontName "Arial" ConfigGroup`

Skin bangs

`!Show`, **`!Hide`**, **`!Toggle`**Parameter: *`Config`*

Shows or hides an active skin.

- `Config` (optional)

Example: `!Toggle "illustro\Clock"`

`!ShowFade`, **`!HideFade`**, **`!ToggleFade`**Parameter: *`Config`*

Shows or hides an active skin with a fade effect.

- `Config` (optional)

`!ShowBlur`, **`!HideBlur`**, **`!ToggleBlur`**Parameter: *`Config`*

Shows or hides the blur behind a skin.

- `Config` (optional)

`!AddBlur`, **`!RemoveBlur`**Parameter: **`Region`**, *`Config`*

Adds or removes a blur region to/from existing blur areas.

- `Region` (required): Region to add or remove a blur effect.
- `Config` (optional)

!Move Parameters: **`X, Y,`** ***`Config`***

Sets the position of a skin.

- `X` (required): New X position.
- `Y` (required): New Y position.
- `Config` (optional)

Example: `!Move "100" "100"`

!ActivateConfig Parameters: **`Config,`** ***`File`***

Activates a skin.

- `Config` (required): The config to be activated.
- `File` (optional): If not specified, the next .ini file variant in the config folder is activated.

Example: `!ActivateConfig "illustro\Clock" "Clock.ini"`

!DeactivateConfig Parameter: ***`Config`***

Deactivates a skin.

- `Config` (required): The config to be deactivated.

Example: `!DeactivateConfig "illustro\Clock"`

!ToggleConfig Parameters: **`Config, File`**

Activates or deactivates a skin.

- `Config` (required): The config to be activated or deactivated.

- `File` (required): The .ini file to be activated or deactivated.

!Update Parameter: *Config*

Overrides the setting of the Update option in [Rainmeter] and immediately updates the entire skin, all measuresand meters. This does not override any UpdateDivider options on measures or meters.

- `Config` (optional)

!Redraw Parameter: *Config*

Overrides the setting of the Update option in [Rainmeter] and immediately redraws all visible elements of the entire skin. Meters will use the last value obtained for any measures or variables referenced.

- `Config` (optional)

!Refresh Parameter: *Config*

Reads the skin file and recreates the skin.

- `Config` (optional)

!SkinMenu Parameter: *Config*

Opens the skin context menu at the current mouse position.

- `Config` (optional)

!SetTransparency Parameters: **Alpha**, *Config*

Sets the transparency of a skin.

- `Alpha` (required): From 0 (invisible) to 255 (opaque).
- `Config` (optional)

Example: `!SetTransparency "128" "illustro\Clock"`

!ZPos Parameters: **Position**, *Config*

Sets the Z-position of a skin.

- `Position` (required): `-2` for On desktop, `-1` for Bottom, `0` for Normal, `1` for On top, or `2` for Always on top.
- `Config` (optional)

Example: `!ZPos "2" "illustro\Clock"`

`!Draggable`Parameters: **`Setting`**, *`Config`*

Sets the Draggable option of a skin.

- `Setting` (required): `0` to disable, `1` to enable or `-1` to toggle.
- `Config` (optional)

`!KeepOnScreen`Parameters: **`Setting`**, *`Config`*

Sets the KeepOnScreen option of a skin.

- `Setting` (required): `0` to disable, `1` to enable or `-1` to toggle.
- `Config` (optional)

`!ClickThrough`Parameters: **`Setting`**, *`Config`*

Sets the ClickThrough option of a skin.

- `Setting` (required): `0` to disable, `1` to enable or `-1` to toggle.
- `Config` (optional)

`!SnapEdges`Parameters: **`Setting`**, *`Config`*

Sets the SnapEdges option of a skin.

- `Setting` (required): `0` to disable, `1` to enable or `-1` to toggle.
- `Config` (optional)

Skin group bangs:

`!ShowGroup`, **`!HideGroup`**, **`!ToggleGroup`**Parameter: **`Group`**

Shows or hides a group of active skins.

- `Group` (required): Group to show or hide.

Example: `!ShowGroup "SomeGroup"`

`!ShowFadeGroup`, **`!HideFadeGroup`**, **`!ToggleFadeGroup`**Parameter: **`Group`**

Shows or hides a group of active skins with a fade effect.

- `Group` (required): Group to show or hide with a fade effect.

`!DeactivateConfigGroup`Parameter: **`Group`**

Deactivates a group of skins.

- `Group` (required): Group to deactivate.

`!Updategroup`Parameter: **`Group`**

Overrides the setting of the Update option in [Rainmeter] and immediately updates the skins in the specified group, all measures and meters. This does not override any UpdateDivider options on measures or meters.

- `Group` (required): Group to redraw.

`!RedrawGroup`Parameter: **`Group`**

Overrides the setting of the Update option in [Rainmeter] and immediately redraws all visible elements of the specified group of skins. Meters will use the last value obtained for any measures or variables referenced.

- `Group` (required): Group to redraw.

`!RefreshGroup`Parameter: **`Group`**

Reads the skin file and recreates a group of skins.

- `Group` (required): Group to refresh.

!SetTransparencyGroup Parameters: **Alpha, Group**

Sets the transparency of the configs in the specified group.

- `Alpha` (required): From 0 (invisible) to 255 (opaque).
- `Group` (required): Name of the the group.

Example: `!SetTransparencyGroup "128" "SuiteName"`

!DraggableGroup Parameters: **Setting, Group**

Sets the Draggable option for each config in the specified group.

- `Setting` (required): `0` to disable, `1` to enable or `-1` to toggle.
- `Group` (required): Name of the group.

!ZPosGroup Parameters: **Position, Group**

Sets the Z-position of the configs in the specified group.

- `Position` (required): `-2` for On desktop, `-1` for Bottom, `0` for Normal, `1` for On top, or `2` for Always on top.
- `Group` (required): Name of the group.

!KeepOnScreenGroup Parameters: **Setting, Group**

Sets the KeepOnScreen setting for the specified group.

- `Setting` (required): `0` to disable, `1` to enable or `-1` to toggle.
- `Group` (required): Name of the group.

!SnapEdgesGroup Parameters: **Setting, Group**

Sets the SnapEdges setting for the specified group.

- `Setting` (required): `0` to disable, `1` to enable or `-1` to toggle.
- `Group` (required): Name of the group.

Meter bangs

!ShowMeter, **!HideMeter**, **!ToggleMeter**Parameters: **Meter,** *Config*

Shows or hides the given meter.

- `Meter` (required): Name of the meter to show or hide.
- `Config` (optional)

Example: `!ToggleMeter "MyMeter"`

!UpdateMeter Parameters: **Meter**, *Config*

Overrides the setting of the Update option in [Rainmeter] or any UpdateDivider on the meter, and immediately updates the meter, obtaining new current values for any measures or variables referenced. Note that the meter is not redrawn with any new values until the next update, or if a !Redraw bang is used.

- `Meter` (required): Name of the meter to update. Use `*` to update all meters.
- `Config` (optional)

Example: `!UpdateMeter "MyMeter"`

!MoveMeter Parameters: **X**, **Y**, **Meter**, *Config*

Moves the given meter (not the window) to the specified position. The new position is relative to the top left corner of the skin.

- `X` (required): New X position.
- `Y` (required): New Y position.
- `Meter` (required): Name of the meter to move.
- `Config` (optional)

Example: `!MoveMeter 15 10 "MyMeter"`

Meter group bangs:

!ShowMeterGroup, **!HideMeterGroup**, **!ToggleMeterGroup**Parameters: **Group**, *Config*

Shows or hides a group of meters.

- `Group` (required): Group to show or hide.

- `Config` (optional).

!UpdateMeterGroup Parameters: **Group**, *Config*

Overrides the setting of the Update option in [Rainmeter] or any UpdateDivider on the meters in the specified group, and immediately updates the meters, obtaining new current values for any measures or variables referenced. Note that the meters are not redrawn with any new values until the next update, or if a!RedrawGroup bang is used.

- `Group` (required): Name of the meter group to update.
- `Config` (optional)

Measure bangs

!EnableMeasure, **!DisableMeasure**, **!ToggleMeasure**Parameters: **Measure**, *Config*

Enables or disables the given measure.

- `Measure` (required): Name of the measure.
- `Config` (optional)

This behavior can also be controlled with the Disabled general measure option. Numeric values in measures will be set to zero when the measure is disabled.

Example: `!ToggleMeasure "CPUMeasure"`

!PauseMeasure, **!UnpauseMeasure**, **!TogglePauseMeasure**Parameters: **Measure**, *Config*

Pauses or unpauses updating the given measure.

- `Measure` (required): Name of the measure.
- `Config` (optional)

This is similar to !DisableMeasure / !EnableMeasure, however the numeric value of the measure will contain the most recent value, rather than being set to zero.

Example: `!TogglePauseMeasure "CPUMeasure"`

!UpdateMeasure Parameters: **Measure**, *Config*

Overrides the setting of the Update option in [Rainmeter] or any UpdateDivider on the measure, and immediately updates the measure.

- `Measure` (required): Name of the measure. Use `*` to update all measures.
- `Config` (optional)

Example: `!UpdateMeasure "CPUMeasure"`

`!CommandMeasure`Parameters: **`Measure, Arguments,`** *`Config`*

Sends a command to the given measure.

- `Measure` (required): Name of the measure.
- `Arguments` (required): Arguments to send to the measure.
- `Config` (optional)

Example: `!CommandMeasure "NowPlayingParent" "Previous"`

Measure group bangs:

`!EnableMeasureGroup`, **`!DisableMeasureGroup`**, **`!ToggleMeasureGroup`**Parameters: **`Group,`** *`Config`*

Enables or disables the given measures in the specified group.

- `Group` (required): Name of the group.
- `Config` (optional)

`!PauseMeasureGroup`, **`!UnpauseMeasureGroup`**, **`!TogglePauseMeasureGroup`**Parameters: **`Group,`** *`Config`*

Pauses or unpauses updating the given measure in the specified group.

- `Group` (required): Name of the group.
- `Config` (optional)

`!UpdateMeasureGroup`Parameters: **`Group,`** *`Config`*

Overrides the setting of the Update option in [Rainmeter] or any UpdateDivider on the measures in the specified group, and immediately updates the measures.

- `Group` (required): Name of the group.

- `Config` (optional)

Deprecated bangs

`!Execute`

The !Execute bang, used both to indicate multiple bangs in one statement and to execute external applications has been made optional. Do not use it in new skins.

`!Rainmeter...`

Bangs names can optionally be preceded by the "Rainmeter" keyword. For example, !RainmeterShowMeter and !ShowMeter are both equal. New skins should not use the Rainmeter prefix.

`!PluginBang`Parameters: **`Measure`**, *`Arguments`*, *`Config`*

This bang has been deprecated in favor of !CommandMeasure and should not be used in any new skins.

Variables [home]

A **variable** is a string of text that is associated with a short, memorable name. Variable names can be used in mostoptions throughout the skin, in place of the associated text.

Variables are helpful when a certain string is repeated many times throughout the skin. By referencing the variable name instead of the full string, the amount of "redundant" code is reduced, as is the overall size of the skin. Variables also simplify the task of changing values. A variable string can be modified in a single location, and without recreating the entire option in which it is used.

[Variables] section

Each skin may have a special section called **[Variables]**. Each option in [Variables] defines a variable name. Unlike most other option names in Rainmeter, variable names are not limited to a specific list. They can have any valid name, as long as they do not conflict with Rainmeter's built-in variables.

```
[Variables]

Foo=This is a string!

Bar=So is this!
```

Select all

Using Variables

Variable names are referenced in other options by surrounding the name with pound signs (`#`), as in `#Foo#`. When the skin is loaded, Rainmeter replaces the variable reference with the corresponding string. Variables are inserted "literally," which means they can be mixed with other variables or regular text.

```
[MeterFoo]

Meter=String

Text=The value of my "Foo" variable is: #Foo#

FontColor=#Red#,#Green#,#Blue#,#Alpha#


[MeterBar]
```

```
Meter=String

Text=The value of my "Bar" variable is: #Bar#

FontColor=#Red#,#Green#,#Blue#,#Alpha#
```

Select all

Variables can also be used to define other variables:

```
[Variables]

Foo=rainmeter

Bar=http://www.#Foo#.net/
```

Select all

Changing Variables

The !WriteKeyValue bang can be used to rewrite values in [Variables]. The skin must be refreshed to apply changes made by !WriteKeyValue.

`!WriteKeyValue "Variables" "Foo" "This is a new string!"`

Alternatively, the !SetVariable bang can be used to change variable values dynamically. !SetVariable can be used to create a variable that does not already exist in the [Variables] section.

`!SetVariable "Foo" "This is a new string!"`

Dynamic Variables

A variable whose value changes while the skin is active is called a **dynamic variable**. Dynamic variables allow a skin to store, retrieve and display ever-changing information without refreshing the skin.

Dynamic variables can only be used in a measure or meter with the option `DynamicVariables=1`. Any `[MeasureName]` used in a bang as a Section Variable is automatically dynamic, and `DynamicVariables=1` is not required.

In addition, there are some plugins which do not support dynamic variables at this time. The following are sections where dynamic variables cannot be used:

- [Rainmeter]
- [Variables]
- [Metadata]
- iTunes
- MediaKey
- Power
- VirtualDesktops
- 3rd-party plugins are not guaranteed to support dynamic variables.

When a meter or measure has dynamic variables enabled, Rainmeter uses marginally more processor power. This difference is negligible for most skins, but it may be noticeable in very large, complex skins with many dynamic variables, or on image meters with very large images. For this reason, it is recommended to use dynamic variables only when necessary, and use alternatives such as !SetOption in other cases.

Built-In Variables

Rainmeter automatically creates a number of helpful variables for each skin. These variables do not need to be defined in the [Variables] section. Some built-in variables are dynamic.

For a complete list, see Built-In Variables.

Section Variables

Some measure and meter properties can also be referenced as dynamic variables by using the section name in brackets (`[]`), as in `[SectionName]`. These **section variables** can also be modified by certain parameters, as in `[SectionName:P1,P2]`.

For more, see Section Variables.

Event Variables

Some action options allow the use of special variables that are related to the specific event that triggers the action. These variables are referenced by surrounding the name with dollar signs (`$`), as in `$Foo$`. Event variables are evaluated at the time the action is triggered, used to execute the action, and then immediately discarded. For example, when using a command with the InputText plugin, the event variable `$UserInput$` refers to the string that was entered into a text input box by the user.

Environment Variables

Windows environment variables can be referenced as variables in Rainmeter by using the variable name contained between percent signs (`%`), such as `%APPDATA%` or `%SystemDrive%`. To see a complete list of environment variables available on your system, open a Windows command prompt (cmd.exe) and type `set`.

Escaping Variables

Rainmeter always attempts to replace `#Name#` or `[Name]` with a value, if "Name" refers to an existing variable, meter or measure. To prevent a variable reference from being replaced, place asterisks (`*`) inside the containers, as in:

- `#*Foo*#`
- `[*Bar*]`

These are replaced with `#Foo#` and `#Bar#`, respectively. The !SetOption bang can use these escape characters to protect variable references when setting options on dynamic meters or measures.

Built-In Variables [home]

Rainmeter automatically creates a number of helpful variables for each skin. These variables do not need to be defined in the [Variables] section and cannot be directly modified by actions in a skin. Some built-in variables aredynamic.

- Path variables
- Skin variables
- Miscellaneous variables
- Monitor variables

Note: All path variables already contain a trailing slash "\".

Path variables

#PROGRAMDRIVE#Example: `C:` or `\\server\Users\`

> Drive or server Rainmeter is located on.

#PROGRAMPATH#Example: `C:\Program Files\Rainmeter\`

> Path to the program folder containing Rainmeter.exe.

#SETTINGSPATH#Example: `C:\Users\YourName\Appdata\Roaming\Rainmeter\`

> Path to the folder containing Rainmeter.ini and other settings files and folders.

#SKINSPATH#Example: `C:\Users\YourName\My Documents\Rainmeter\Skins\`

> Path to the skins folder.

#PLUGINSPATH#Example: `C:\Program Files\Rainmeter\Plugins\`

> Path to the built-in plugins folder.

#ADDONSPATH#Example: `C:\Users\YourName\Appdata\Roaming\Rainmeter\Addons\`

Path to the addons folder.

Note: `#ADDONSPATH#` should be avoided when possible. Addons should be kept in the @Resources folderinstead.

Skin variables

#@#Example: `C:\Users\YourName\Documents\Rainmeter\Skins\illustro\@Resources\`

Path to the @Resources folder for the current skin.

#CURRENTPATH#Example: `C:\Users\YourName\Documents\Rainmeter\Skins\illustro\Clock\`

Path to the folder containing the current skin file.

#CURRENTFILE#Example: `Clock.ini`

File name of the current skin.

#ROOTCONFIGPATH#Example: `C:\Users\YourName\Documents\Rainmeter\Skins\illustro\`

Path to Root config - Highest-level folder under the skins folder for the current skin.

#ROOTCONFIG#Example: `illustro`

Name of Root config - Highest-level folder under the skins folder for the current skin.

#CURRENTCONFIG#Example: `illustro\Clock`

Config name of current skin.

#CURRENTCONFIGX#, **#CURRENTCONFIGY#**, **#CURRENTCONFIGWIDTH#**, **#CURRENTCONFIGHEIGHT#**

Position and size of the current skin.

Note: These variables are dynamic.

Miscellaneous variables

#CRLF#

Creates a new line (carriage return / linefeed) where used.

#CURRENTSECTION#

The name of the section in which the variable is used. Provides a blank string if used in another context, such asGetVariable in a Lua script.

Monitor variables

Note: These variables are dynamic.

#WORKAREAX#, **#WORKAREAY#**, **#WORKAREAWIDTH#**, **#WORKAREAHEIGHT#**

Work area position and size of the current monitor.

#SCREENAREAX#, **#SCREENAREAY#**, **#SCREENAREAWIDTH#**, **#SCREENAREAHEIGHT#**

Screen area position and size of the current monitor.

#PWORKAREAX#, **#PWORKAREAY#**, **#PWORKAREAWIDTH#**, **#PWORKAREAHEIGHT#**

Work area position and size of the primary monitor.

#PSCREENAREAX#, **#PSCREENAREAY#**, **#PSCREENAREAWIDTH#**, **#PSCREENAREAHEIGHT#**

Screen area position and size of the primary monitor.

#WORKAREAX@*N*#, **#WORKAREAY@*N*#**, **#WORKAREAWIDTH@*N*#**, **#WORKAREAHEIGHT@*N*#**

Work area position and size of the *N*th monitor.

#SCREENAREAX@*N*#, **#SCREENAREAY@*N*#**, **#SCREENAREAWIDTH@*N*#**, **#SCREENAREAHEIGHT@*N*#**

Screen area position and size of the *N*th monitor.

#VSCREENAREAX#, **#VSCREENAREAY#**, **#VSCREENAREAWIDTH#**, **#VSCREENAREAHEIGHT#**

Position and size of the virtual screen.

Section Variables [home]

Measures and meters can be referenced as variables. These are called **section variables**, and they can provide several kinds of information about the meter or measure.

Usage

A meter or measure is referenced as a section variable by placing the section name in brackets (`[]`). The value provided by a section variable can be changed by adding parameters after a colon (`:`). Multiple parameters are separated by commas (`,`). Spaces are allowed after a comma.

```
[SectionName:P1,P2]
```

Normal or built-in variables take priority over section variables, which means they can be used within section variables. For example: `[#Foo#:#Bar#]`. (The reverse, such as `#[Foo][Bar]#`, is not valid.)

Dynamic Variables

Section variables are always dynamic. `DynamicVariables=1` is only needed in the section where the variable is being used, not the section that is being referenced.

Meter Parameters

None.

> Section variables for meters have no value without a parameter.

`:X`, **`:Y`**Example: `[MeterName:X]`

> The current X or Y position of the meter.
>
> **Note:** This provides the "real" X or Y value, which is the position of the top-left corner of the meter area. This is always an integer, even if the meter has no X and Y options set, or if the options use formulas, variables or relative positions.
>
> This also means that for a string meter with StringAlign options, the value of `[MeterName:X]` and `[MeterName:Y]` may not be the same as the option values.

`:W`, **`:H`**Example: `[MeterName:W]`

The current width or height of the meter.

Note: This provides the "real" W or H value. This is always an integer, even if the meter has no W and Hoptions set, or if the options use formulas or variables.

Measure Parameters

None.Example: `[MeasureName]`

If no parameters are given, the measure's string value is provided.

:Example: `[MeasureName:]`

If a blank parameter is given, the measure's number value is provided, with up to ten decimal places of precision.

:nExample: `[MeasureName:10]`

The measure's number value, with the number of **decimal places** given.

Multiple Parameters: This parameter may be combined with Scale and Percentual.

:/n Example: `[MeasureName:/1024]`

The measure's number value, **scaled** by the divisor given.

Multiple Parameters: This parameter may be combined with Decimals and Percentual.

:%Example: `[MeasureName:%]`

The measure's number value, as a **percentual** value.

Multiple Parameters: This parameter may be combined with Decimals and Scale.

:MinValue :MaxValue Example: `[MeasureName:MaxValue]`

The measure's MinValue or MaxValue number value.

Multiple Parameters: This parameter may be combined with Decimals and Scale.

Mouse Variables [home]

Mouse variables are a special function to return the X and Y position of the mouse cursor when a mouse action takes place.

The position is relative to the meter that has the mouse click action, or the skin if used in the [Rainmeter] section.

There are two variants of the function.

- **$MouseX$** and **$MouseY$**
  Contains the current X and Y position of the mouse in pixels relative to the meter or skin.
- **$MouseX:%$** and **$MouseY:%$**
  Contains the current X and Y position of the mouse as a percentage relative to the meter or skin.

Usage

The variables are only created and used in the context of a mouse click action. Primarily, they will be used as a parameter to a Bang. For instance:

```
LeftMouseUpAction=[!SetOption SomeMeter X $MouseX$][!UpdateMeter *][!Redraw]
```

```
LeftMouseUpAction=!CommandMeasure ScriptMeasure GetRGB($MouseX$,$MouseY$)
```

Notes: The values returned are not the mouse position on the screen, but are pixels or a percentage relative to the meter or skin with the mouse action. They are also not general purpose variables, and when used outside the context of a mouse action on a meter or the skin, will not contain a value.

$MouseX$

X position of the mouse cursor as a number of pixels relative to the meter or skin with the mouse action.

$MouseY$

Y position of the mouse cursor as a number of pixels relative to the meter or skin with the mouse action.

$MouseX:%$

X position of the mouse cursor as a percentage relative to the meter or skin with the mouse action.

`$MouseY:%$`

> Y position of the mouse cursor as a percentage relative to the meter or skin with the mouse action.

Example

```
[Rainmeter]

LeftMouseUpAction=[!SetOptionGroup Coordinates Text "Click the square!"][!UpdateMeterGroup Coordinates][!Redraw]


[Background]

Meter=Image

SolidColor=0,0,150

W=150

H=225


[CoordinateA]

Meter=String

FontColor=255,255,255

Text=Click the square!

Group=Coordinates


[RedSquare]

Meter=Image

SolidColor=255,0,0

X=25

Y=30

W=100

H=100

LeftMouseUpAction=[!SetOption CoordinateA Text "($MouseX$, $MouseY$)"][!UpdateMeter CoordinateA][!Redraw]

MouseActionCursorName=Cross


[CoordinateB]
```

```
Meter=String

FontColor=255,255,255

Text=Click the square!

Group=Coordinates

Y=20R


[GreenSquare]

Meter=Image

SolidColor=0,255,0

X=25

Y=180

W=100

H=25

LeftMouseUpAction=[!SetOption CoordinateB Text "X = $MouseX:%$%, Y = $MouseY:%$%"][!UpdateMeter CoordinateB][!Redraw]

MouseActionCursorName=Cross
```

Select all

Groups [home]

Skins, meters, and measures can be categorized into groups to allow easier control with group bangs. For example, the `!HideMeterGroup` bang may be used to hide multiple meters in a single bang (compared to `!HideMeter` statements for each meter).

Options

`Group`

Defines the group(s). Multiple groups can be specified using `|` as the delimiter.

- Skin groups are defined under the [Rainmeter] section of a skin or the individual [ConfigName] sections of Rainmeter.ini, and are used by the skin group bangs.
- Meter groups are defined in the meter section and are used by the meter group bangs.
- Measure groups are defined in the Measure section and are used by the measure group bangs.

Example: `Group=SomeGroup` or `Group=FirstGroup | SecondGroup | ThirdGroup`

Mouse Actions [home]

Mouse actions are action options used on any visible part of the skin. The action is triggered by specific mouse events.

Usage

- On any meter. The target area detected by the mouse will be any non-tranparent areas of the meter, or any part of the meter which has a non-transparent meter or skin background behind it.
- In the [Rainmeter] section of the skin if Background is defined.

Note: Actions defined for a meter will override actions defined in the `[Rainmeter]` section.

Mouse Click Options

Note: Mouse Click Options may be overridden by holding down **CTRL** while clicking.

`LeftMouseDownAction`

Action to execute when the left mouse button is pressed.

Note: This disables dragging the skin. See LeftMouseUpAction.

`RightMouseDownAction`

Action to execute when the right mouse button is pressed.

Note: This disables the skin context menu.

`MiddleMouseDownAction`

Action to execute when the middle mouse button is pressed.

`LeftMouseUpAction`

Action to execute when the left mouse button is released.

Note: This will generally be the desired option for a left mouse click. LeftMouseDownAction should be avoided unless there is a specific need to trap the downward press, as it will disable the ability to drag the skin.

`RightMouseUpAction`

Action to execute when the right mouse button is released.

Note: This disables the skin context menu.

`MiddleMouseUpAction`

Action to execute when the middle mouse button is released.

`LeftMouseDoubleClickAction`

Action to execute when the left mouse button is double clicked.

Note: If `LeftMouseDownAction` or `LeftMouseUpAction` is also set, both will be executed.

`RightMouseDoubleClickAction`

Action to execute when the right mouse button is double clicked.

Note: This disables the skin context menu.

`MiddleMouseDoubleClickAction`

Action to execute when the middle mouse button is double clicked.

`X1MouseDownAction, X2MouseDownAction`

Action to execute when a supported extra mouse button is pressed.

Note: Some manufacturers and/or software have the ability to change what each extra button can do, therefore these actions may not work for everyone. It is not recommended to distribute skins that rely on these actions.

`X1MouseUpAction, X2MouseUpAction`

Action to execute when a supported extra mouse button is released.

`X1MouseDoubleClickAction, X2MouseDoubleClickAction`

Action to execute when a supported extra mouse button is double clicked.

Mouse Hover Options

`MouseOverAction`

Action to execute when the mouse cursor is moved over the meter or skin.

`MouseLeaveAction`

Action to execute when the mouse cursor leaves the meter or skin.

Mouse Wheel Scroll Options

Actions to be taken when the mouse scroll wheel is rotated while the cursor is over a skin or meter containing the option.

`MouseScrollDownAction`

Action to execute when the mouse wheel is rotated down.

MouseScrollUpAction

Action to execute when the mouse wheel is rotated up.

MouseScrollLeftAction

Action to execute when the mouse wheel is tilted or rotated to the left. *Not all mice have this capability.*

MouseScrollRightAction

Action to execute when the mouse wheel is tilted or rotated to the right. *Not all mice have this capability.*

Mouse Cursor Options

MouseActionCursor Default: 1

When set to 1 (default) on a meter with a mouse action, a pointer cursor will be shown when hovering the mouse over the meter.

The default for the entire skin can be set to 0 by putting `MouseActionCursor=0` in the [Rainmeter] section of the skin. This can then be overridden on a meter-by-meter basis with `MouseActionCursor=1`.

Note: If you have a meter with a mouse action, and there is a meter on top of it, you will need to set `MouseActionCursor=0` on the foreground meter (even if it does not have a mouse action).

MouseActionCursorName

If a custom cursor file (i.e. .cur or .ani) is found in @Resources\Cursors in the root config folder of a skin, it will be automatically loaded and available as an alternative for the standard "hand" pointer for meters or skins with a mouse action.

The option can be used with `MouseActionCursorName=MyCustomCursor.cur` in any meter or the [Rainmeter] section. No path is used in the option. In addition, several built-in Windows cursors may be used without needing any file name or extension. `Hand, Text, Help, Busy, Cross, Pen`

Lua Scripting [home]

Rainmeter has the ability to load and execute scripts in **Lua**, a functional programming language. Rainmeter includes the Lua 5.1 standard libraries, which encompass a variety of powerful features.

A **script** refers to a set of Lua functions that is associated with a script measure. These functions may be executed when the skin loads, when it updates, or on command.

This page details the Rainmeter-specific modifications and new functions that have been added to Rainmeter's built-in Lua environment. More documentation for Lua itself is available at:

- Lua 5.1 Reference Manual
- Lua Tutorial
- *Programming in Lua* by Roberto Ierusalimschy

The rest of this page assumes a basic knowledge of the Lua language.

Script Measure

The script measure is used to load a Lua script from a file and interface with the script. The script file must be a text file, and typically has the extension `.lua`.

Much like plugin measures, each script measure creates a separate **instance** of its script. This means that a skin can have any number of scripts loaded simultaneously—even from the same script file. (The order in which scripts are executed is determined by the measure order.) "Global" variables are not shared between instances.

```
[MeasureName]

Measure=Script

ScriptFile=MyScript.lua
```

Select all

Options

In addition to general measure options and ScriptFile, scripts also allow user-defined options. These options may have any name and value, and may be changed with !SetOption. The script can read and use its own option values using the SELF object functions. This allows the same script file to be used with different parameters depending on the context.

```
[MeasureName]

Measure=Script

ScriptFile=MyScript.lua
```

```
MyOption=Hello, world!
```

Select all

Dynamic Variables

Dynamic variables are generally not needed with script measures. This is because functions are provided to get the current values of variables, measures and options within Lua. If these functions are used in the Update function, they will return the current values at the time the function is called.

!CommandMeasure

The !CommandMeasure bang can be used to execute Lua code in the context of a particular script instance:

```
!CommandMeasure "MyScriptMeasure" "MyFunction()"
```

Multiple statements may be separated by semicolons (`;`). All statements are global.

```
!CommandMeasure "MyScriptMeasure" "a = b; print(SKIN:ParseFormula('2+2'))"
```

All statements must be passed as a single parameter in the bang. Because single-quotes (`'`) and double-quotes (`"`) are both valid string containers in Lua, while only double-quotes are recognized in Rainmeter, single quotes are recommended when passing strings with !CommandMeasure.

Initialize

If the **Initialize** function is defined in any script, the function is called one time (without parameters) when the skin is activated or refreshed. This happens even if the script measure is disabled. If the script file is changed by a!SetOption bang, the new script's Initialize function is called as well.

```
function Initialize()

        MyVariable = 'Hello, world!'

end
```

Select all

Actions that are needed to "set up" the script, such as declaring global variables, should be done in the Initialize function.

Update

If the **Update** function is defined in any script, the function is called (without parameters) whenever the script measure updates. The script measure responds normally to the Disabled option, the UpdateDivider option, and allmeasure bangs.

```
function Update()
```

```
        MyVariable = 'Hello, world!'

        return MyVariable
end
```

Select all

The Update function's **return value** determines what values are provided by the script measure. Strings and numbers in Lua are analogous to string values and number values in Rainmeter measures. Examples:

- `return`
  Provides *0* as the number value, and '' (blank) as the string value. The same is true if no `return` is stated.
- `return 99` and `return '99'`
  Provides *99* as the number value, and '*99*' as the string value.
- `return 'Ninety-Nine'`
  Provides *0* as the number value (because the string cannot be converted to a number), and '*Ninety-Nine*' as the string value.

The values provided by the script measure can be used in the same way as other measure values. (**Note:** the values only update when the measure itself updates. Calling `Update()` within Lua does not update the measure values.)

Log

Lua errors are logged in the About window. The print function may also be used to write strings to the log. This can provide helpful feedback when writing or troubleshooting a script.

```
print('The current value of MyVariable is: ' .. MyVariable)
```

Functions

Lua functions are provided to identify a meter, a measure, or the current skin as a Lua **object**. Additional functions are provided for manipulating each type of object in specific ways.

SKIN object

The `SKIN` object is created automatically.

`GetMeasure`Parameter: **`MeasureName`**

> Creates an object for the named measure. Returns `nil` if the measure is not found.
>
> **Example:** `MyMeasure = SKIN:GetMeasure('MeasureName')`

`GetMeter`Parameter: **`MeterName`**

Creates an object for the named meter. Returns `nil` if the meter is not found.

Example: `MyMeter = SKIN:GetMeter('MeterName')`

GetVariableParameter: **MeterName**, *Default*

Returns the current value of the named variable as a string. If the variable does not exist, returns the given default value, or `nil` if no default is given.

Example: `MyVariable = SKIN:GetVariable('VariableName', 'n/a')`

BangParameters: **Bang** or **BangName**, *BangArg1*, *BangArg2*, *BangArgN*

Executes a bang. The bang can be constructed in one of two ways:

1. The entire bang as a single string.

   Example: `SKIN:Bang('!SetOption "MyMeter" "Text" "Hello, world!"')`

   Multiple bangs are valid using this method.

   Example: `SKIN:Bang('[!UpdateMeter "MyMeter"][!Redraw]')`

2. Each bang parameter as a separate parameter in the function.

   Example: `SKIN:Bang('!SetOption', 'MyMeter', 'Text', 'Hello, world!')`

MakePathAbsoluteParameter: **File/Folder**

Converts a relative filepath into an absolute filepath, in the same manner as a path option.

Example: `MyPath = SKIN:MakePathAbsolute('MyImage.png')`

ReplaceVariablesParameter: **String**

Returns the given string, with all valid variable values properly replaced. Section variables are valid.

Example: `MyString = SKIN:ReplaceVariables('The value of MyVariable is #MyVariable#.')`

ParseFormulaParameter: **FormulaString**

If the given string is a valid formula, evaluates the formula and returns the result as the number. Otherwise, returns `nil`.

Example: `MyNumber = SKIN:ParseFormula('2+2')`

Measure object

A `Measure` object is created using GetMeasure. It is linked to a specific measure in the skin.

Example: `MyMeasure = SKIN:GetMeasure('MeasureName')`

`GetValue`

Returns the current number value of the measure.

Example: `MyMeasureValue = MyMeasure:GetValue()`

`GetStringValue`

Returns the current string value of the measure.

Example: `MyMeasureValue = MyMeasure:GetStringValue()`

`GetOption`Parameters: **`OptionName`**, *`Default`*

Returns the current value of the named option as a string. If the option does not exist, returns the given default value, or `''` if no default is given.

Example: `MyGroup = MyMeasure:GetOption('Group', 'None')`

`GetNumberOption`Parameters: **`OptionName`**, *`Default`*

Returns the current value of the named option as a number. If the option does not exist, or is not a valid number or formula, returns the given default value, or `0` if no default is given.

Example: `MyUpdateDivider = MyMeasure:GetNumberOption(UpdateDivider, 1)`

`GetName`

Returns the measure's name as a string.

Example: `MyMeasureName = MyMeasure:GetName()`

`GetMinValue`

Returns the current minimum value of the measure as a number.

Example: `MyMeasureMin = MyMeasure:GetMinValue()`

`GetMaxValue`

Returns the current maximum value of the measure as a number.

Example: `MyMeasureMax = MyMeasure:GetMaxValue()`

`GetRelativeValue`

Returns the measure's current number percentage value as a number, scaled from `0.0` to `1.0`.

Example: `MyMeasureValue = MyMeasure:GetRelativeValue()`

`GetValueRange`

Returns the current value range of the measure as a number.

Example: `MyMeasureRange = MyMeasure:GetValueRange()`

`Disable`

Disables the measure.

Example: `MyMeasure:Disable()`

`Enable`

Enables the measure.

Example: `MyMeasure:Enable()`

SELF object

The `SELF` object is created automatically. `SELF` is a measure object linked to the current script measure. Allmeasure object functions are valid for `SELF`.

```
MyScriptMeasureName = SELF:GetName()
```

Meter object

A `Meter` object is created using GetMeter. It is linked to a specific meter in the skin.

Example: `MyMeter = SKIN:GetMeter('MeterName')`

`GetOption`Parameters: **`OptionName`**, *`Default`*

Returns the current value of the named option as a string. If the option does not exist, returns the given default value, or `''` if no default is given.

Example: `MySolidColor = MyMeter:GetOption('SolidColor', '000000')`

GetName

Returns the measure's name as a string.

Example: `MyMeterName = MyMeter:GetName()`

GetXParameters: *Absolute*

Returns the current X position of the meter as a number. If the optional *Absolute* parameter is `true`, returns the absolute (or "real") X position.

Example: `MyX = MyMeter:GetX()`

GetYParameters: *Absolute*

Returns the current Y position of the meter as a number. If the optional *Absolute* parameter is `true`, returns the absolute (or "real") Y position.

Example: `MyY = MyMeter:GetY()`

GetW

Returns the current "real" width of the meter as a number.

Example: `MyW = MyMeter:GetW()`

GetH

Returns the current "real" height of the meter as a number.

Example: `MyH = MyMeter:GetH()`

SetX

Sets the X position of the meter.

Example: `MyMeter:SetX()`

SetY

Sets the Y position of the meter.

Example: `MyMeter:SetY()`

SetW

Sets the width of the meter.

Example: `MyMeter:SetW()`

`SetH`

Sets the height of the meter.

Example: `MyMeter:SetH()`

`Hide`

Hides the meter.

Example: `MyMeter:Hide()`

`Show`

Shows the meter.

Example: `MyMeter:Show()`

Restrictions

The following Lua features are not available in Rainmeter at this time:

- The Debug library.
- External libraries, such as LuaCURL.
- The Require function.
- The Dofile function.

Deprecated Features

The following Rainmeter-specific features have been deprecated and should not be used. They are still supported, but may be removed in future versions.

- `PROPERTIES = { ... }`
  This table was previously used to read options on the script measure. Instead, use:
  `SELF:GetOption('MyOption')` or `SELF:GetNumberOption('MyOption')`
- `MyMeter:SetText('...')`
  This function was used to change a string meter's text. Instead, use:
  `SKIN:Bang('!SetOption', 'MeterName', 'Text', '...')`
- `function GetStringValue()`
  This function was used to set the script measure's value. Instead, use:
  `function Update() return ... end`