Variables

General

A variable is a named value which changes over time e.g. the level of the battery, the time of day.

When Tasker encounters a variable name in a text, it replaces the name with the current value of the relevant variable before carrying out the action.

The main purposes of variables are:

- *dynamic binding*: doing something with an action with data which is unknown when the task is created e.g. respond to an SMS; the sender is not known until the SMS is received.
- allow flow control within and between tasks
- record data for some future use e.g. passing data between tasks

Built-In Variables

The values of Built-In variables are updated by Tasker. Their names always use all-capital letters.

- **Airplane Mode Status**
  (dynamic)
  %AIR
  Whether Airplane Mode is **on** or **off**
- **Battery Level**
  %BATT
  Current device battery level from 0-100.
- **Bluetooth Status** (dynamic)
  %BLUE
  Whether Bluetooth is on or off.
- **Call Name / Number/ Date / Time (In)** (dynamic, monitored)
  %CNAME / %CNUM / %CDATE / %CTIME
  The caller name, number, date and time of the last call received.
  Caller number is **0** if it's unknown.
  Caller name is **?** if it's unknown (probably because the caller number was blocked) and set to the caller number if the contact couldn't be looked up. It's unavailable on Android versions prior to 2.0.
- **Call Name / Number/ Date / Time / Duration (Out)**(dynamic, monitored)
  %CONAME / %CONUM / %CODATE / %COTIME / %CODUR
  The called name, number, date and time of the last (**not** the current) outgoing call made.
  Called Name is set to the called number if the contact couldn't be looked up. It's unavailable on Android versions prior to 2.0.
- **Cell ID** (dynamic,monitored)
  %CELLID
  The current cell tower ID if known.
  If you are using a Cell Near state, note that sometimes the Cell Near state will stay active even though %CELLID reports that the tower ID is unknown or invalid; that is because Cell Near only responds to valid IDs to prevent the state becoming inactive e.g. due to a service interruption.

- **Cell Signal Strength** (dynamic,monitored)
  %CELLSIG

  The current phone signal level from 0-8 inclusive on a rougly linear scale. On some CDMA phones, the level will rise in steps of 2 (0,2,4,6,8). The value is -1 if the value is unknown or there is e.g. no service.
  There is a bug with some Android versions that the reported signal strength is not updated until the device is turned off and on.
- **Cell Service State** (dynamic,monitored)
  %CELLSRV
  The current phone service state. One of *unknown, service, noservice, emergency, nopower*.
- **Clipboard Contents**
  %CLIP
  The current contents of the system clipboard.
- **CPU Frequency**
  %CPUFREQ
  The current frequency CPU 0 is running at. See also: CPU Control.
- **CPU Governor**
  %CPUGOV
  The current governor controlling the frequency of CPU 0. See also: CPU Control.
- **Date**
  %DATE
  Current human-readable date.
- **Day of the Month**
  %DAYM
  Current Day of the Month, starting at 1.
- **Day of the Week**
  %DAYW
  Current Day of the Week starting with Sunday.
- **Display Brightness** *%BRIGHT*
  Current screen brightness, 0-255. On some devices, if the Android setting Auto Brightness is enabled, the value will always be 255.
- **Display Timeout** *%DTOUT*
  Current system screen timeout (seconds).
- **Email From / Cc / Subject / Date / Time** (dynamic)
  %EFROM / %ECC / %ESUBJ / %EDATE / %ETIME
  The From, Cc, Subject, Received Date and Received Time of the last email received by the K9 email agent.
- **Free Memory**
  %MEMF
  System free memory remaining in MB.
- **GPS Status**
  %GPS
  Whether the system GPS receiver is **on** or **off**.
- **HTTP Response Code / Data / Content Length**
  (dynamic)) *%HTTPR / %HTTPD / %HTTPL*
  Values from the last HTTP POST/GET action.
- **Keyguard Status**
  %KEYG
  Whether the Keyguard is **on** or **off**

- **Last Application**
  %LAPP
  The name of the application that was in the foreground before the current one e.g. Maps.
- **Last Photo**
  %FOTO
  The filesystem path to the last photo taken by Tasker or the standard system camera application.
- **Light Level** (dynamic,monitored)
  %LIGHT
  The last recorded light level in lux.
  Note that Android does not return a value until the light level changes, so to test the sensor is working you should put it near a bright light initially.
- **Location** (dynamic)
  %LOC
  The latitude and longitude of the last GPS fix.
  See note.
- **Location Accuracy** (dynamic)
  %LOCACC
  The accuracy in metres of the last GPS fix.
  See note.
- **Location Altitude** (dynamic)
  %LOCALT
  The altitude in metres of the last GPS fix, or 0 if unavailable.
  See note.
- **Location Speed** (dynamic)
  %LOCSPD
  The speed in metres/second at the last GPS position fix or 0 if unavailable.
  See note.
- **Location Fix Time Seconds** (dynamic)
  %LOCTMS
  The time in seconds of the last GPS fix. To get age of fix, take this away from %TIMES.
  This value is not set until an offset of the GPS time from the fixed time has been calculated (should be after the first GPS fix) because the value is meaningless until that point.
  See note.
- **Location (Net)** (dynamic)
  %LOCN
  The latitude and longitude of the last network location fix.
  See note.
- **Location Accuracy (Net)** (dynamic)
  %LOCNACC
  The accuracy in metres of the last network location fix.
  See note.
- **Location Fix Time (Net)** (dynamic)
  %LOCNTMS
  The time in seconds of the last net location fix. To get age of fix, take this away from %TIMES.
  See note.
- **Music Track** (dynamic))

%MTRACK
The current playing music track, supported for:
- Tasker actions *Music Play* and *Music Play Dir*
- Built-in Android music-player, probably not on all devices however
- Power AMP
- Phantom Music Control Pro

Priority: if both Tasker and one of the other supported apps are playing simultaneously, the non-Tasker track will be shown. If more than one of the other supported apps is playing simultaneuosly, behaviour is unspecified.
Notes:
- if you don't have a supported player, you could try Phantom Music Control Pro, which supports a lot of players and should pass the info on to Tasker
- pausing a track clears the variable, unpausing sets it again

- **Muted**
%MUTED
Whether the microphone is currently muted (**on**) or not (**off**).
- **Night Mode**
%NIGHT
The current Android Night Mode.
One of **on**, **off** or **auto**.
If **auto**, Android will decide whether it should be in Night Mode itself.
- **Notification Title** (monitored, dynamic)
%NTITLE
The title of the last notification shown in the status bar. Requires Tasker's accessibility server to be running (see Android Accessibility Settings). Notifications generated by Tasker are not shown. Notifications for some apps will not register i.e. the variable will be blank.
Not available on Cupcake.
- **Phone Number**
%PNUM
The current phone number of the device, if it's in service.
On some phones it doesn't work (Android limitation), seems related to the type of SIM.
- **Profiles Active** (dynamic)
%PACTIVE
A comma-separated list of the currently active, named profiles in activation order. Duplicate names will appear on the list only once. The list always starts and ends with a comma to make matching easier, if it's not empty.
- **Profiles Enabled** (dynamic)
%PENABLED
A comma-separated list of the currently enabled, named profiles in creation order. Duplicate names will appear on the list only once. The list always starts and ends with a comma to make matching easier, if it's not empty.
- **Roaming**
%ROAM
on if device is roaming on the current telephone network, otherwise **off**.
- **Screen** (dynamic)
%SCREEN
Whether the screen is on (value **on**) or off (value **off**).
- **Silent Mode** (dynamic)
%SILENT

The current state of silent mode: **off**, **vibrate** or **on**.

- **SIM Serial Number**
  %SIMNUM
  The serial number of the SIM card, if one is present and accessible.
  If the SIM has not been unlocked it will not be available.
- **SIM State**
  %SIMSTATE
  The current state of the SIM card.
  One of **unknown**, **absent**, **pinrequired**, **pukrequired**, **netlocked** or **ready**.
- **Speakerphone**
  %SPHONE
  Whether the speakerphone is **on** or **off**
- **Speech** (dynamic)
  %SPEECH
  The current utterance as a result of a *Say* or *Say File* action, if applicable.
- **Task Queue Seconds** (dynamic)
  %QTIME
  The number of seconds since the current task first started executing. Note that tasks can be interrupted by higher priority tasks, so this number is not necessarily the total run-time of the task.
- **Tasks Running** (dynamic)
  %TRUN
  A comma-separated list of any named tasks which are currently running. The list always starts and ends with a comma to make matching easier, if it's not empty.
- **Telephone Network** (dynamic, monitored)
  %TNET
  The current telephony network operator the device is using.
  May be unreliable on CDMA networks
- **Text From/Date/Subject/Time** (monitored)
  %SMSRF / %SMSRN / %SMSRB / %SMSRD / %MMSRS / %SMSRT
  The sender address, name, body, date and time of the last text (SMS or MMS) received.
  These variables will be empty until the first time a text is received after they have been referenced because Tasker does not monitor SMSs unless it's needed.
  Sender name is set to sender address of no contact could be looked up. It's unavailable on Android versions prior to 2.0.
  Subject will only be set if the last text was an MMS.
- **Time**
  %TIME
  Current human-readable time separated by a period e.g. 10:59
- **Time Seconds**
  %TIMES
  The current time in seconds.
  (seconds since some time in January, 1970, if you must know).
- **UI Mode** (dynamic,monitored)
  %UIMODE
  The current Android UI mode.
  One of **car**, **desk** or **normal**.
- **Uptime Seconds**
  %UPS
  The number of seconds since the device last booted.

- **Volume - Alarm/Call/DTMF/Media/Notification/Ringer/System** (dynamic)
  %VOLA / %VOLC / %VOLD / %VOLM / %VOLN / %VOLR / %VOLS
  Current audio channel volume level.
  On some devices, volume changes are not picked up dynamically, on others not when using the phone app.
- **WiFi Info**
  %WIFII
  When connected to an Access Point (AP), shows human-readable data about the AP. When not connected, show details of the most recent Wifi scan results for nearby APs.
- **WiFi Status** (dynamic)
  %WIFI
  Whether WiFi is **on** or **off**. Note: if WiFi is enabling or disabled, in fact anything but enabled, it's classed as off.
- **Wimax Status**
  %WIMAX
  Whether Wimax is **on** or **off**. Note: if Wimax is enabling or disabled, in fact anything but enabled, it's classed as off.
- **Window Label** (monitored)
  %WIN
  The label of the current window, which could be a full-screen activity or a dialog. Not set if the label is unknown.
  For some windows, the label might be that of the first item in the window e.g. a menu entry or even a button.

General Notes

Variables marked dynamic in the list above trigger changes in *Variable Value* states and *Variable Set* events whenever their value changes.

Variables marked monitored will cause the relevant monitor to startup to track their state when they are used in contexts or tasks which are used by widgets or **enabled** profiles. For instance, %CELLS used in a Flash action will cause cell location to be tracked.

Limitation: monitored variables cannot be detected in anonymous shortcuts.

Note On Location Variables

When the relevant provider (Net or GPS) of a location context is active, these variables report the values from the provider, which may be more recent than Tasker has seen if other applications are asking for location.

When the relevant provider is **not** active, these variables report the last values **seen by Tasker**, which could be the result of a Get Location action or of monitoring for a Location Context.

That means the the reported fix times could **go backwards**, if you turn off the location provider between two uses of the variables.

Location variables can also be manually updated by running the Get Location action.

User Variables

The action *Variable Set* (and several others) can be used to create new variables. Variable names have the following restrictions:

- they must start with the **%** character
- they are case-sensitive
- then must at least a further **3** alphanumeric characters
- they can also contain the underscore character (_) but not start or end with it

Global vs Local Variables

All built-in variables are *global*, meaning they are **visible anywhere** in Tasker (e.g. %WIFI)

User variables which have one or more capital letters in their name are also global (e.g. %Car)

However, user variables which have **all-lower-case** names (e.g. %fruit) are *local*, meaning they are only visible **in the task in which they are used** (or the scene in which they are used, if the task was launched from by a scene event).

In general, it's best to use local variables wherever possible because:

- you know they won't be interfered with by other tasks
- they are more efficient in several ways

Note: multiple copies of the same task running at the same time each have their own separate copy of their local variables.

Variable Lifetime

The value a variable holds should last until Tasker is uninstalled if it is not changed by any task.

Uninitialized Variables

User-variables which have not had a value assigned do not have replacements carried out e.g. in the expression *I love %fruit*, if %fruit is uninitialized, the expression remains as it is, otherwise %fruit is replaced with the value.

Exception: uninitialized variables used in mathematical expressions are replaced with 0.

Variables In Plugins

Plugin developers can tell Tasker to replace variables it finds in plugin strings with their current Tasker value. If you have a plugin which doesn't support this, send the developer this URL

> http://tasker.dinglisch.net/plugins.html

which has the relevant details.

Variable Arrays

Tasker supports pseudo-arrays.

They are especially useful when used with the For action, since you can perform a set of actions on each element in turn e.g. list a set of files then test each one.

Examples

If the four variables **%arr1, %arr2, %arr3, %arr4** hold respectively **a, b, c** and **d** then we have an array with 4 *elements*. These variables can be used just like any other, however it is also possible to access them in special ways. Here are some examples:

- **%arr(#)**
  The number of defined array elements (**4** in this case)
- **%arr(#>)**
  The index of the first defined array element, or **0** if none are defined (**1**).
- **%arr(#<)**
  The index of the last defined array element, or **0** if none are defined (**4**)
- **%arr(#?b/c)**
  A comma-separated list of the array indices (lowest to highest) with matching values, or **0** if none match (**2,3** in the example)
- **%arr(>)**
  The contents of the first defined array element (**a**)
- **%arr(<)**
  The contents of the last defined array element (**d**)
- **%arr()** or **%arr(:)**
  All of the array elements separated by commas (**a,b,c,d**)
- **%arr(2)** or just **%arr2**
  The content of the element with index 2 (**b**)
- **%arr(2:4)**
  Contents of defined elements with indices 2 to 4 (**b,c,d**)
- **%arr(:3)**
  All the defined elements with indices up to 3 (**a,b,c**)
- **%arr(3:)**
  All the defined elements with indices starting from 3 (**c,d**)
- **%arr(1:2)**
  All the defined elements with indices from 1 to 2 (**a,b**)

Notes:

- arrays will virtually always have all their elements defined so e.g. %arr(>) will be the same as %arr(1), %arr(#) will be the same as %arr(#<)
- index specifiers can themselves be variables (e.g. %arr(1:%MAX) or %arr(#?%FINDME))

Creating An Array

1. using Variable Split:
   Variable Set %arr a,b,c,d
   Variable Split %arr
   If your data has commas in it, you can separate the values with e.g. @ and specify @ also in the Variable Split action.
2. by assigning individual elements with Variable Set:
   Variable Set, %arr3, c.
3. using Array Push to add an initial element
4. some other actions also create arrays for their results e.g. List Files.

Inserting Elements

Use the Array Push action.

The *Fill Spaces* parameter might need more explanation. It is only relevant if one or more of the array elements are undefined. As an example, if we have the array elements %arr1 and %arr3 containing **apple** and **banana**:

- **Array Push %arr1, 1, pear**

leaves %arr1, %arr2 and %arr4 containing **pear**, **apple** and **banana**.
- but **Array Push %arr2, 1, pear, Fill Spaces**
  leaves %arr1, %arr2 and %arr3 containing **pear**, **apple** and **banana**.

Removing Elements

Use the Array Pop action. Note the difference between Array Pop and Variable Clear: Pop reduces the number of elements in the array, while Clear merely changes elements to undefined.

Example: if we have the array elements %arr1, %arr2, %arr3 containing **apple**,**pear** and **banana**:

- **Variable Clear %arr2**
  leaves %arr1 and %arr3 containing **apple** and **banana**.
- but **Array Pop %arr2**
  leaves %arr1 and %arr2 containing **apple** and **banana**.

Deleting An Array

Use Array Clear.

In most cases you could also use **Variable Clear %arr*** with Pattern Matching checked, but that would also delete variables called e.g. %arrTOODEETOO so Array Clear is safer.

Array Efficiency

Arrays are intended for convenience when processing high-level data, not for e.g. processing astronomical data. Doing thousands of array actions will likely take several seconds (although mostly due to the housekeeping work done by Tasker in-between each action rather than due to the array operations themselves).

In terms of storage efficiency, they are also fairly hopeless. You probably do not want to store tens of thousands of items in an array.