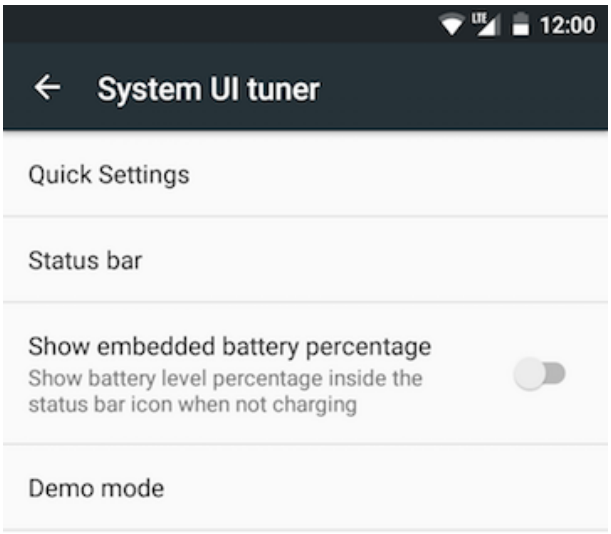
Android M: Tweaking System UI Tuner

Back to Blog Home

Everything described in this post is based off the Android 'M' Developer Preview, Release 2. As new releases (and the source code) are made available, behaviors are subject to change and the contents of this post may change along with them.

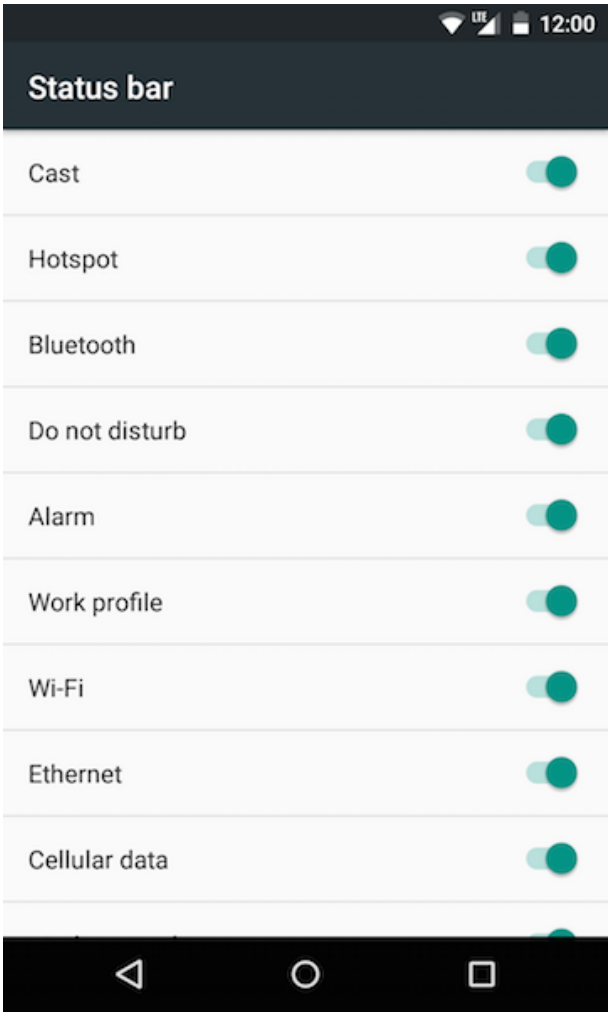


By now, you've probably already seen (or at least heard about) the new System UI Tuner developer option in Android 'M'. Beginning with Preview 2, this power tool has two new tricks up its sleeve:

1. Allows control over which status bar icons are displayed
2. Includes a one-click "demo" mode to set the entire status bar to a pre-set default (static) state

The best part is, both of these features are externally accessible. We can fairly easily control them from both the device shell and using public APIs in an application.

Status Icon Control



With the UI found under **System UI tuner -> Status bar**, developers can easily turn off most of the primary status icons (except for notifications, battery level, and clock). Fortunately for us, these changes are persisted into `Settings.Secure` under the key `icon_blacklist`. This is what the value looks like with all the icons disabled:

```
$ adb shell settings get secure icon_blacklist
ethernet,mobile,airplane,managed_profile,zen,cast,wifi,alarm_clock,hotspot,bluetooth
```

It's a comma-separated list of all the icon names. Below is a mapping of those names to their Settings description:

Name	Settings Description
cast	Cast
hotspot	Hotspot
bluetooth	Bluetooth
zen	Do not disturb
alarm_clock	Alarm
managed_profile	Work profile
wifi	Wi-Fi
ethernet	Ehternet
mobile	Cellular Data
airplane	Airplane Mode

The SystemUI application observes changes to `icon_blacklist`, so we can programmatically control the value from the shell. Here are some working examples:

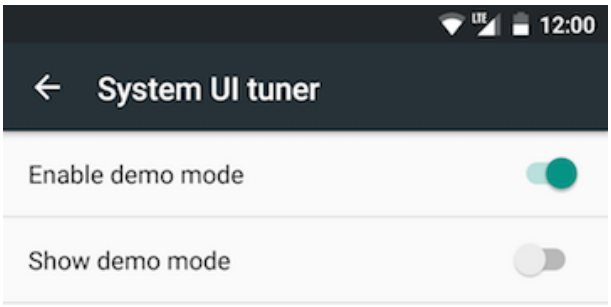
```
# Enable all icons (remove the setting)
$ adb shell settings delete secure icon_blacklist

# Just disable Bluetooth icon
$ adb shell settings put secure icon_blacklist bluetooth

# Disable Bluetoot and Wi-Fi icons
$ adb shell settings put secure icon_blacklist bluetooth,wifi
```

Since this value lives in `Settings.Secure`, it cannot be written by third party application code (only read).

Demo Mode - More than Meets the Eye



On the surface, demo mode looks shows as a one-click solution to quickly set the status bar to a pre-defined "clean" state. It doesn't offer any configuration options for controlling the appearance of the status bar items...or does it?

While demo mode in and of itself cannot be configured, it turns out that the mechanism used to trigger these changes inside of SystemUI is unprotected broadcast intents. This means that, if we can figure out what intents SystemUI expects, we can configure our own "demo mode" from the shell or some application code.

SystemUI listens for broadcasts with the `com.android.systemui.demo` action string, and forwards them onto the various demo mode handlers. Demo mode is controlled with a series of "commands" that are passed along in the broadcasts as extras. Here is a list of the current top-level commands:

- enter
- exit
- clock
- battery
- volume
- status
- network
- notifications
- bars

The `enter` and `exit` commands are used to turn demo mode on and off (the equivalent of the "Show demo mode" UI command, while the remaining commands configure the status bar if demo mode is enabled. From the shell, we could do the following:

```
# Enable demo mode
$ adb shell am broadcast -a com.android.systemui.demo --es command enter

# Disable demo mode
$ adb shell am broadcast -a com.android.systemui.demo --es command exit
```

Similarly, the following application code snippets can toggle demo mode:

```
// Enable demo mode
Intent intent = new Intent("com.android.systemui.demo");
intent.putExtra("command", "enter");
sendBroadcast(intent);

// Disable demo mode
Intent intent = new Intent("com.android.systemui.demo");
intent.putExtra("command", "exit");
sendBroadcast(intent);
```

Let's look in more details at each of the available commands...

clock

This command is used to set the value of the clock displayed—the default demo mode value is "5:20". It supports the following extra parameters:

Name	Description	Values
hhmm	Value of the clock hours/minutes	"hhmm" string, such as "1200" for 12:00

Here is an example of setting the clock to 12:00, instead of 5:20, from the shell and application code:

```
$ adb shell am broadcast -a com.android.systemui.demo --es command clock \
    --es hhmm 1200
```

```
Intent intent = new Intent("com.android.systemui.demo");
intent.putExtra("command", "clock");
intent.putExtra("hhmm", "1200");
sendBroadcast(intent);
```

battery

This command is used to control the display of the battery level icon. It supports the following extra parameters:

Name	Description	Values
level	Battery level	0-100%
plugged	Connected to AC	true/false

The following example sets the battery level to 50% and removes the "charging" symbol:

```
$ adb shell am broadcast -a com.android.systemui.demo --es command battery \
    --es level 50 \
    --es plugged false
```

```
Intent intent = new Intent("com.android.systemui.demo");
intent.putExtra("command", "battery");
intent.putExtra("level", "50");
intent.putExtra("plugged", "false");
sendBroadcast(intent);
```

volume

This command is listed but not currently used in the preview. It's handler is a no-op.

network

This command controls the display of the network icons (Wi-Fi, mobile). If supports the following extra parameters:

Name	Description	Values
mobile	Display the mobile network icon	hide/show
wifi	Display the wifi network icon	hide/show
airplane	Display the airplane mode icon	hide/show
sims	Set number of SIM card slot icons	1+
nosim	Display the No SIM icon	hide/show

When the `mobile` or `wifi` commands are supplied, additional sub-commands will control their appearance:

Name	Description	Values
level	Set signal strength	-1 = No service, 0 = Lowest, 4 = Full Bars
datatype	Mobile network icon type	1x, 3g, 4g, e, g, h, lte, or roam
fully	Show the associated network as connected	true/false

The following example sets the mobile data icon to show connected Wi-Fi and LTE with 75% signal:

```
$ adb shell am broadcast -a com.android.systemui.demo --es command network \
    --es mobile show \
    --es fully true \
    --es level 3 \
    --es datatype lte
$ adb shell am broadcast -a com.android.systemui.demo --es command network \
    --es wifi show \
    --es fully true \
    --es level 3 \
```

```
Intent intent = new Intent("com.android.systemui.demo");
intent.putExtra("command", "network");
intent.putExtra("mobile", "show");
intent.putExtra("fully", "true");
intent.putExtra("level", "3");
intent.putExtra("datatype", "lte");
sendBroadcast(intent);

intent.removeExtra("mobile");
intent.removeExtra("datatype");
intent.putExtra("wifi", "show");
sendBroadcast(intent);
```

status

This command is used to control the remaining icons displayed in the status bar. It supports the following extra parameters:

Name	Description	Values
volume	Display vibrate icon	vibrate/none
bluetooth	Display Bluetooth icon	connected/disconnected/none
location	Display location icon	hide/show
alarm	Display alarm clock icon	hide/show
zen	Display priority notifications icon	important/none
mute	Display silence icon	hide/show
speakerphone	Display speakerphone icon	hide/show
managed_profile	Display work profile icon	hide/show
cast	Display cast icon	hide/show
hotspot	Display hotpost icon	connected/none

The current preview does not support this command—it ignores any incoming requests. This may get fixed in a future preview. The default behavior of demo mode is to hide all these icons.

notifications

This command controls the visibility of the notification icons. It supports the following extra parameters:

Name	Description	Values
visible	Whether or not to display notification icons	true/false

The following example hides the notification icons from status bar:

```
$ adb shell am broadcast -a com.android.systemui.demo --es command notifications \
    --es visible false
```

```
Intent intent = new Intent("com.android.systemui.demo");
intent.putExtra("command", "notifications");
intent.putExtra("visible", "false");
sendBroadcast(intent);
```

bars

This command controls the coloring of the status bar and soft navigation bar. It supports the following extra parameters:

Name	Description	Values
mode	Display mode of status bar and navigation bar backgrounds	opaque, translucent, semi-transparent, transparent, warning

The `warning` value sets the bars to the orange state used in battery saver mode.
The following example forces the status and naviation bars to have an opaque black background:

```
$ adb shell am broadcast -a com.android.systemui.demo --es command bars \
    --es mode opaque
```

```
Intent intent = new Intent("com.android.systemui.demo");
intent.putExtra("command", "bars");
intent.putExtra("mode", "opaque");
sendBroadcast(intent);
```

Enabling External Control

Before any of the above commands will work, the "Enable demo mode" switch in **System UI Tuner -> Demo mode** must be toggled on. This process can be done by the user, or we can automate it from the shell:

```
$ adb shell put settings global sysui_demo_allowed 1
```

In the current preview, demo mode doesn't have to actually be enabled for external broadcasts to work. It just has to have been enabled at least once in the past. This may change in future releases.
This part cannot be done from application code because applications do not have the rights to write into the `Settings.Global` table.

Custom Demo Example

Tying a handful of these commands together, we can create an application method that builds the status bar just the way we want it for our demos:

```
public void onEnableClick(View v) {
    //Enter demo mode
    Intent intent = new Intent("com.android.systemui.demo");
    intent.putExtra("command", "enter");
    sendBroadcast(intent);

    //Set the clock to 12:00
    intent.putExtra("command", "clock");
    intent.putExtra("hhmm", "1200");
    sendBroadcast(intent);

    //Show the LTE icon at 75% strength
    intent.putExtra("command", "network");
    intent.putExtra("mobile", "show");
    intent.putExtra("fully", "true");
    intent.putExtra("level", "3");
    intent.putExtra("datatype", "lte");
    sendBroadcast(intent);

    //Show the Wifi icon at 75% strength
    intent.removeExtra("mobile");
    intent.removeExtra("datatype");
    intent.putExtra("wifi", "show");
    sendBroadcast(intent);

    //Show the battery unplugged at 50%
    intent.putExtra("command", "battery");
    intent.putExtra("level", "50");
    intent.putExtra("plugged", "false");
    sendBroadcast(intent);

    //Hide all notifications
    intent.putExtra("command", "notifications");
    intent.putExtra("visible", "false");
    sendBroadcast(intent);

    //Make the status bar a fixed black background
    intent.putExtra("command", "bars");
    intent.putExtra("mode", "opaque");
    sendBroadcast(intent);
}

public void onDisableClick(View v) {
    //Exit demo mode
    Intent intent = new Intent("com.android.systemui.demo");
    intent.putExtra("command", "exit");
    sendBroadcast(intent);
}
```

Hopefully in future versions, this control will become more formal and accessible. But until then, enjoy tweaking your status bars to look perfect for those app screenshots!

Want to learn more about Android development? Check out one of our upcoming Android training courses!