

electronic periodical

# YOUTH SCIENTIFIC AND TECHNICAL BULLETIN

Bauman MSTU Publ

EL № FS-77-51038,

ISSN 2307-0609

---

УДК 004.51

## Mobile OS, localized

*Vladislav Koldobskiy, student*

*Russia, 105005, Moscow, Bauman Moscow State Technical University  
Automated control and data processing systems*

*Scientific adviser: Elena Rumyantseva, docent*

*Russia, 105005, Moscow, Bauman Moscow State Technical University  
[bauman@bmstu.ru](mailto:bauman@bmstu.ru)*

The world of mobile technologies is one of the fiercest competitive markets ever. In the quest to stand out from the pack, a lot of device manufacturers make software modifications, both big and small. But there are also users who want to improve the experience on their own and create an aftermarket operating system... for mobile devices.

In this article I'll describe some development stages of our latest translation platform and how did we end up being there.

To start off, let's describe what is CyanogenMod in general.

In short, this is a modification of Google Android OS, similar to what vendors do, but in a different way. What most manufacturers' software is missing is the ability to customize little details on how it looks, feels and acts. This is what CyanogenMod's version is about — making the device work exactly the way you want it to. Not just that, we also do care about your privacy, bringing Secure Messaging — a transparent SMS encryption that works within all CyanogenMod users, and Privacy Guard — a protected sandbox for apps you don't trust very much.

Last but not best, CM is open to new contributions. That means you can submit your own idea, fix, improvement, or anything else you feel should be there. Looking ahead, we have over 3000 contributors that improve the project in many different ways.

With all this in mind, how many people are there who actually use our software?

To answer this question, there is a statistics feature built into CyanogenMod that sends a “check-in” to our servers when device turns on.

## Statistics

Devices that have not checked in within the last 90 days are periodically removed from the database. Statistics may be delayed by up to one minute.

### Total Installs

Type	Total
Official Installs	6,529,495
Unofficial Installs	5,177,108
<b>Total Installs</b>	<b>11,706,603</b>
Last 24 Hours	18,230

Fig. 1. Installation statistics

So we have almost 12 million active users. Plus, there are also a lot of people who don’t go online very often but still use CyanogenMod. Our “total downloads” counter shows over 45 million downloads.

These people are from different countries, so we should make CM “speak” other languages too. That’s why we started the translation project.

### Understanding the structure

Since Android is modular, we have different parts that needs our attention.

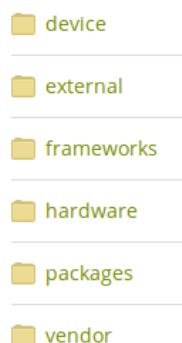


Fig. 2. Directory structure of translatable parts

I'll describe three “fundamental” parts of them, namely, “device”, “frameworks”, and “packages”.

First off, let's take “frameworks” directory. Aside from a lot of “non-interface-related” parts, there is a “base framework” package. This is the thing that forms the whole Android interface, everything you can see on the screen. While it may look like this is the biggest part to translate, it's not. We didn't start translation from scratch (after all, it's Google's OS), so compared to other parts there's not very much to translate — the only concern is that the text left untranslated will be seen throughout the whole OS.

What's a lot bigger in size is “packages” directory — a place where all applications shipped with the OS are located, like Messaging, Settings, Phone, Email, Camera, and everything else that is on “out of the box” device.

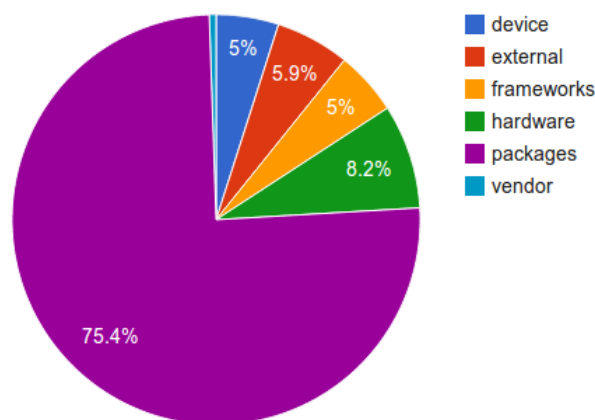


Fig. 3. Distribution of CM-added translatable text

As for “device” directory, it contains device-specific or manufacturer-specific additions or extensions. This is something that shouldn't be on every device, but needed for, let's say, ASUS-made keyboard dock or color calibration on a Samsung device.

So what does interface translation consist of?

All Android projects' structure can be divided in two parts. That's “source”, containing all the Java source code, and “resources”, containing images, XML-structured menu data, and strings, also XML-structured. The latter is what makes the whole interface (application, or the whole framework) translated into different language.

```
<!-- Blacklist preferences -->
<string name="blacklist_title">Blacklist</string>
<string name="blacklist_summary_disabled">Disabled</string>
<string name="blacklist_summary">Incoming messages from phone numbers in the blacklist will be blocked</string>
```

Fig.4. XML-structured interface strings

Specifically, we need to have similar files, with same string IDs, but containing our translations. It sounds simple, but there are many things that should be kept in mind before doing this.

### Problems of an interface translation

By looking at these lines of a XML code, you almost can't tell where they are used, is there a length limit, or how does it look among other elements.

So the first problem with interface translation is that translators work "blindfold" with the product. When doing a literature, or a document translation, you can always take a look at the text and see if the overall style is fine. This is not the case here, because strings can be used (and re-used) in different places of an application.

The second problem comes from the fact that there is a wide variety of different devices with different screen sizes. Hence, translated text that looks fine on a tablet, could be cut off on a small phone's display, leaving user wonder what did we mean.

Third one is more a recommendation than a problem. We absolutely should comply with typographic rules that apply to printed books. This includes using proper quotation marks, “en”- and “em” dashes, etc. Often, this requires a bit of work, but the result is rather pleasant to the eye.

### How did we do translations

When the whole CyanogenMod thing started, we didn't think about translations in first place.

That's why we decided to accept translations using the same system we're using for code submissions, Gerrit — a web-based application that enables discussions on “pending” patches, doing code review and other things like that. While this works fairly well for code, it doesn't really work for translations, mainly because of its requirements for translators. People who did translations were forced to install Linux on the computer, download all Android source code, which is about 40

GBytes in size, and learn how to use all these things. This is rather tough for not very tech-savvy persons. Most terrible thing was review process taking a lot of time, sometimes lasting for months. There just should be a better way to organize the process.

### **It's time to move on**

We started with the goal of simplifying the translation process for new people. To eliminate the problems translators are having, our new system should be cloud-based, i.e. available from any Internet connected device, and have an automatic import feature so no one would be forced to wait until the new translation is approved.

All in all, this comes to a technique called Continuous Localization. Similar to Continuous Integration systems that pulls new code, checks it against common failures and notifies the crew if anything's wrong, Continuous Localization allows translators to be notified when new translation is required. This makes the whole process a lot faster, removing the need to have complete sources on the computer and making a more "international-friendly" release.

The choice of cloud-based translation services is not very rich, so we decided to start with Transifex platform.

Although Transifex is relatively easy to configure, it has a huge amount of bugs with complex XML constructions and attributes. We've been trying to work around them for a couple of months but finally gave up. We realized we should try something else.

Luckily, there's another translation platform, Crowdin, which is maintained by a small company from Ukraine, hence, being a lot more helpful with issues we had initially.

So what are main benefits of Crowdin in place of using Transifex?

Almost no workarounds were required for the system to work properly, plus we also got features we could only dream about before.

One of these features is screenshot tagging functionality that completely removes the problem of translators working "blindfold". This feature allows us, project maintainers, to upload screenshots and tag where the string for translation is exactly located. That way, translators will always know what exactly is being translated.

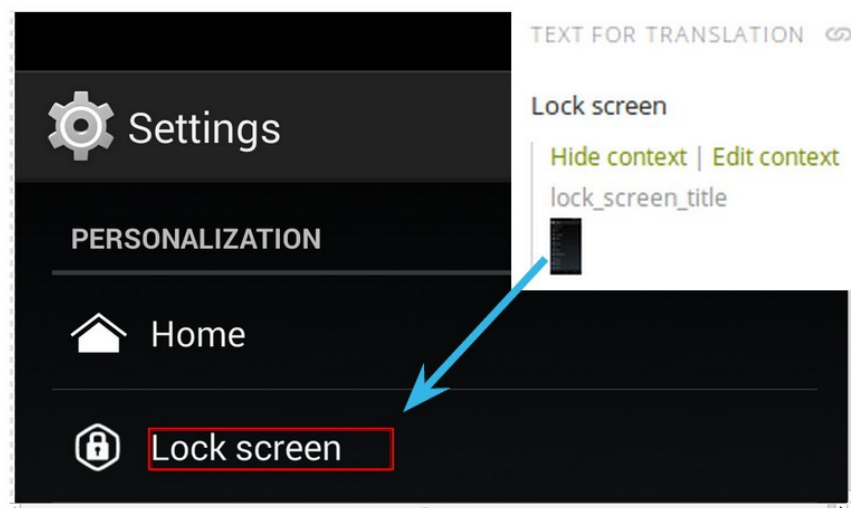


Fig. 5. Crowdin Screenshot tagging feature

Submitting translations also got a lot easier with Machine pre-translation. This is a feature that shows translation suggestions from different machine translation engines.

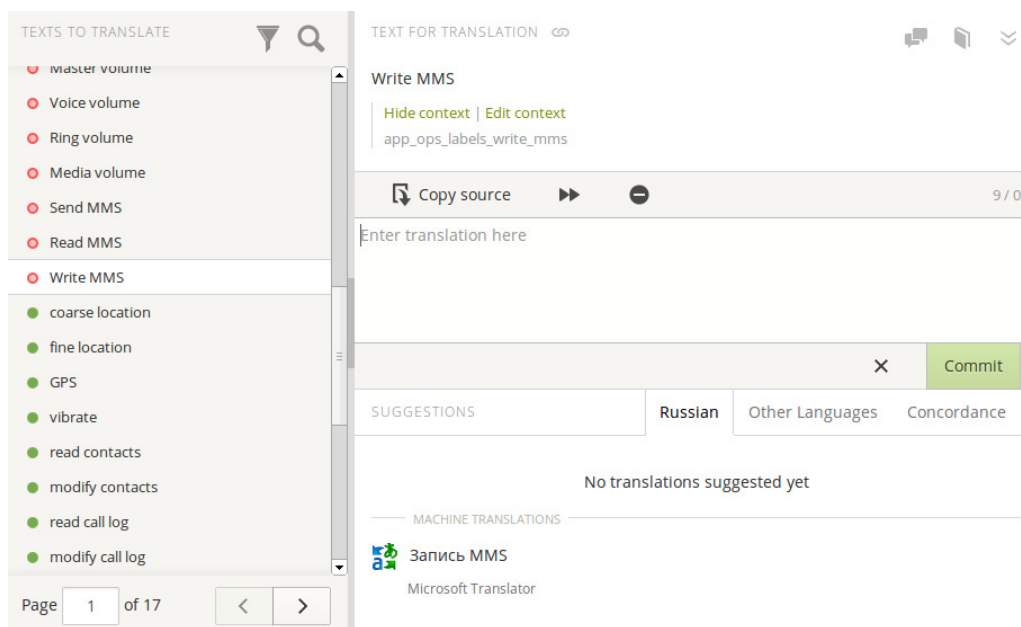


Fig. 6. Crowdin Machine pre-translation feature

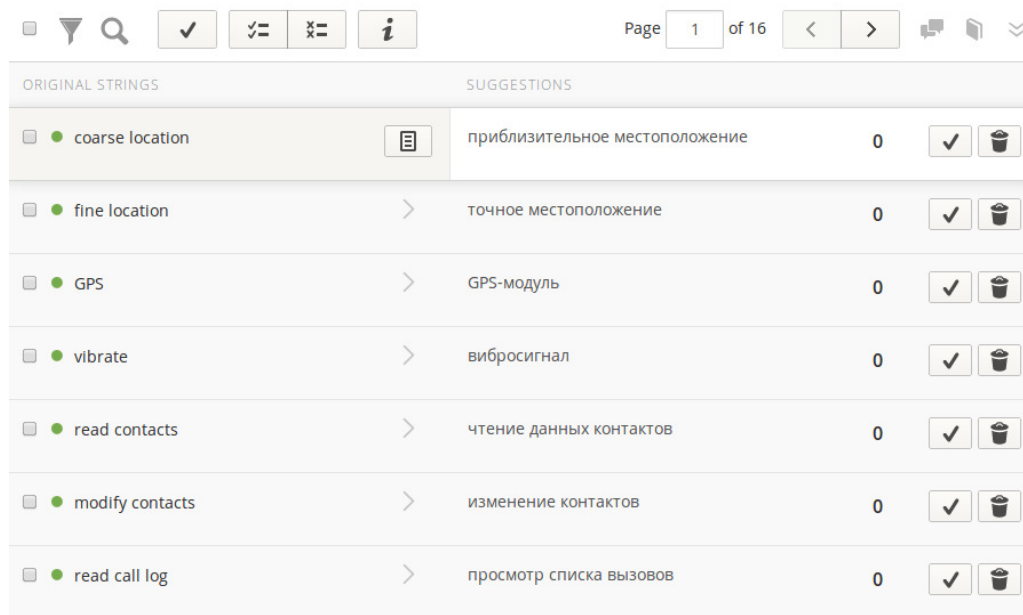
Sometimes, machine translation is actually correct, so the phrase is just a click away from being translated. Also, all translations are automatically saved into a database called “Translation

Memory”. That way, translating the phrase in one project means that same phrases will be automatically translated in others.

Now, how to make a proofreader’s life easier?

Simple enough, our new translation platform allows translators to upvote or downvote translation suggestions, thus, rearranging the list and moving the most correct suggestion to the top of the list.

From the dedicated interface, a person with Proofreader authority can mark translation as “reviewed”, making it go into the next build of CyanogenMod, remove translation if it’s wrong, or submit own suggestion.



The screenshot shows the Crowdin Proofreader interface. At the top, there is a navigation bar with icons for a menu, filter, search, and a status bar indicating 'Page 1 of 16'. Below this is a table with two main columns: 'ORIGINAL STRINGS' and 'SUGGESTIONS'. The 'ORIGINAL STRINGS' column contains a list of English phrases, each preceded by a green dot and a checkbox. The 'SUGGESTIONS' column contains Russian translations, a vote count (all are 0), and icons for upvoting (checkmark) and downvoting (trash can). The first row is highlighted in yellow.

ORIGINAL STRINGS	SUGGESTIONS
<input type="checkbox"/> coarse location	приблизительное местоположение 0 <input type="checkbox"/> <input type="checkbox"/>
<input type="checkbox"/> fine location	точное местоположение 0 <input type="checkbox"/> <input type="checkbox"/>
<input type="checkbox"/> GPS	GPS-модуль 0 <input type="checkbox"/> <input type="checkbox"/>
<input type="checkbox"/> vibrate	вибросигнал 0 <input type="checkbox"/> <input type="checkbox"/>
<input type="checkbox"/> read contacts	чтение данных контактов 0 <input type="checkbox"/> <input type="checkbox"/>
<input type="checkbox"/> modify contacts	изменение контактов 0 <input type="checkbox"/> <input type="checkbox"/>
<input type="checkbox"/> read call log	просмотр списка вызовов 0 <input type="checkbox"/> <input type="checkbox"/>

Fig. 7. Crowdin Proofreader interface

### Availability and conclusion

Finally, let’s give an overview on what our release scheme looks like.

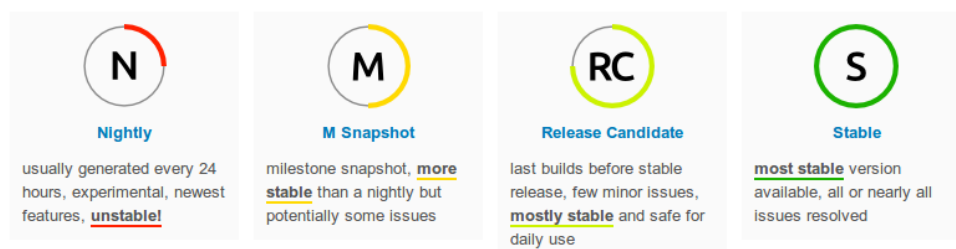


Fig. 8. CyanogenMod release scheme

To prevent stability issues, all translation platform changes have been incorporated into CM nightlies starting from April 3rd. These changes will also go into “M”, “RC”, and “S” builds. Updates are being pushed to all devices running CyanogenMod 11.

Overall, we've come a long way, from not thinking about translations at all to a point where we have an open translation platform that is reliable and easy to use.

### References

1. Writing Style. Android Developers. Available at: <http://developer.android.com/design/style/writing.html>, accessed 10.04.2014.
2. Localizing with Resources. Android Developers. Available at: <http://developer.android.com/guide/topics/resources/localization.html>, accessed 10.04.2014.
3. CyanogenMod | Android Community Operating System. Available at: <http://www.cyanogenmod.org/>, accessed 11.04.2014.
4. CyanogenMod 11 Translations - Crowdin. Available at: <https://crowdin.net/project/cyanogenmod>, accessed 11.04.2014.