OpenMRS Platform Release Process

On this page

- Summary
- Prerequisites
- Prior to Release
- Release Types
    - Alpha Release
    - Beta Releases
        - Testing a Beta Release
    - Release Candidates
    - Final Release
    - Maintenance Release
- Release steps
- Troubleshooting (when not using CI for releasing)
- Tips

Summary

Here's a quick summary of being Platform release manager (from Darius Jazayeri):

1. Pick some tickets and kick others out of the release
2. Keep an eye on those tickets, looking for unassigned & those assigned with no movement, occasionally pinging folks to see what's needed to move things along.
3. For tickets awaiting to be reviewed and whose reviews are small, you can go ahead and review them yourself.
4. Pull together the release (an hour or two of time)
    - Preliminary testing
    - Releasing an alpha release
5. Oversee subsequent releases as described on the Release Process page.

Prerequisites

1. OpenMRS ID
2. GitHub account with write privileges to openmrs-core and openmrs-distro-platform (all /dev/4 and /dev/5, but can be granted temporarily on request)
3. JIRA privileges for administering the OpenMRS Core project
4. Bamboo account,

If you need GitHub/JIRA privileges, Bamboo account, open a ticket to request it.

Prior to Release

Prior to any release make an initial decision and post to Talk on what will be included in the release.

In most cases, this should be fairly clear from the Road Map; however, some projects may be a little behind schedule and a decision made whether they can be properly resourced/sped up to meet the release timeline or need to be bumped to a later release. You do not need to do all of the work of making these decision (the community should do that), but you should help ensure that the discussion occurs and you are the final arbiter for these decisions.

Features that have been committed, but are not going to be in the release need to be removed from the code.

If necessary, hold a Release Prioritization Meeting to prioritize tickets and finalize, which will be included in the release.

As release manager, try as much as possible to have the release scheduled for any working day except Friday. This will effectively publicize the the release to the community in time.

Release Types

Alpha Release

Before an alpha release, all New Feature tickets should either be Closed or at least in a Post Commit state. The same goes for Blocker tickets.

An alpha is a feature-complete release, but is not yet verified to be bug free.

An alpha version is indicated by appending "-alpha" to the version e.g. "2.2.0-alpha".

A release is ready to go from alpha to beta when preliminary testing has been completed and any bugs squashed (a minimum of rudimentary testing, ideally as many steps in the complete testing list as possible). If large bugs are found and squashed, "alpha.2", "alpha.3", etc. may be released to aid testers as needed.

Releasing additional alpha releases is up to the release manager. Going from alpha to beta does not require community discussion and is up to the release manager, but should be done only after the release manager has confirmed with testers that all known bugs have been squashed.

Beta Releases

A beta is a release that is ready to be tested by a larger group of people. Obvious bugs found in the alpha have been found and fixed.

A beta version is indicated by appending "-beta" to the version e.g. "2.2.0-beta".

Be sure to examine all outstanding tickets for the release milestone. By the time a beta release is being considered, all of the tickets assigned to the release milestone should have a clear trajectory (either be moved to a later release or have plans in place to address the issue by the full release).

Testing a Beta Release

- Rudimentary testing should have already been done when creating the beta version.
- Recruit volunteers to help test the beta version, including at least one implementation. Ideally, testers would run through all items in the list of application-level tests "More Complete Testing Process" on the Testing Releases page. It may help to copy this list to a shared document (e.g., Google doc or Wave) and have testers strike out tests that have been completed.
- Consider adding to the More Complete Testing Process list to cover any bugs discovered at the application level (that may not be completely covered by unit testing) which are not currently included in that list.

Going from beta to a release candidate or full release requires that testing have been completed and the system formally exercised by at least one (ideally two or more) implementation.

The decision to go from beta to release candidate or final release should be led by the release manager but agreed by the developer community (via mailing list or dev call).

Release Candidates

A release candidate is only needed when non-trivial changes were required during the beta phase. If the beta release was tested and no significant changes were needed, then you can proceed directly to a full release.

A release candidate version is indicated by appending "-rc" to the version e.g. "2.2.0-rc".

Going from release candidate to full release requires that testing have been completed and the system formally exercised by at least two or more implementations. The decision to go from release candidate to final release should be led by the release manager but agreed by the developer community (via Talk).

Final Release

A true "Release" is deemed tested and worthy of production environments.

Review all existing tickets. Be certain that all tickets assigned to this release milestone have been addressed and that there aren't any straggling tickets for this milestone or previous milestones.

Maintenance Release

A maintenance release contains bug fixes and security patches for use between major releases, e.g., from 1.8.0 to 1.8.1. There is no branch to create. You just pick up where you left off in the current minor version series release branch.

Release steps

There's a number of steps to follow when doing an alpha/beta/rc/final/maintenance release. Please also refer to instructions for different types of releases (alpha/beta/rc/final/maintenance), which you can find in the next paragraphs.

If it is not a maintenance release, the release process should start from creating a maintenance branch.

1. Create a new maintenance branch for openmrs-core and openmrs-distro-platform.
    a. Go to https://github.com/openmrs/openmrs-core.

    b. Click the "branch" dropdown.
    c. Type the new maintenance branch name e.g. "2.2.x".
    d. Click "Create branch '2.2.x' from master".
    e. Go to https://github.com/openmrs/openmrs-distro-platform.
    f. Repeat the steps from b to d.
2. Create a new CI plan for openmrs-core and openmrs-distro-platform.
    a. Go to https://ci.openmrs.org.
    b. Click the "Create" dropdown at the top of the page and choose "Clone an existing plan".
    c. Select "OpenMRS Core Master" as plan to clone and give it e.g. "OpenMRS Core 2.2.x" name.
    d. Edit the newly created plan by going to the "Repository" tab and change the branch of openmrs-core repo to the newly created one e.g. "2.2.x".
    e. Click the "Create" dropdown at the top of the page and choose "Clone an existing plan".
    f. Select "OpenMRS Platform Master" as plan to clone and give it e.g. "OpenMRS Platform 2.2.x" name.
    g. Edit the newly created plan by going to the "Repository" tab and change the branch of openmrs-platform repo to the newly created one e.g. "2.2.x".
3. Announce the upcoming Unsupported Releases (EOL) on Talk under the Software category for the line of two releases back. (e.g. if you're releasing 1.7.0, announce EOL for 1.4.x)

Steps to follow for all releases:

1. Write a post that announces the release date and modules to be included on Talk under the Software category. Ask, if there is a need to release modules in order to include latest changes before the platform release.
2. Verify that there are no unreviewed open tickets against the version to be released. You can use this query https://goo.gl/WXqrI3 (adjust the fixVersion).
    a. Review/close tickets that are in Post Commit Review.
    b. Look carefully at In Progress tickets to see, if some of them have not been committed to a branch (back ported) already. You need to review/close such tickets as well. You can easily check, if any code was committed by searching for an issue id on a GitHub comparison page at https://github.com/openmrs/openmrs-core/compare/2.0.4...2.0.x (adjust the versions).
    c. Move tickets to the next version that are In Progress or in Pre-Commit Review stage (use Manage Versions admin page for bulk moving of tickets, see the 4.a. step below).
3. Create a new version for the next release. Using the Manage Versions in https://tickets.openmrs.org/plugins/servlet/project-config/TRUNK.
4. Release the current version using the above admin page.
    a. When you release a version it will allow you to move non-closed tickets to the next version.
5. Download the latest CIEL for OpenMRS 1.9.x (use 1.9.x regardless of the release version) as described here. Upload the version to our maven repository to be used in standalone by running (adjust the version and the file parameters to match the downloaded version of CIEL):

```
mvn deploy:deploy-file -DgroupId=org.openmrs.contrib -DartifactId=ciel-dictionary -Dversion=1.9.9-
20170409 -Dpackaging=zip -Dfile=openmrs_concepts_1.9.9_20170409.sql.zip -DrepositoryId=openmrs-repo-
contrib -Durl=https://mavenrepo.openmrs.org/nexus/content/repositories/contrib
```

6. Update the CIEL version in standalone's pom.
7. Check if the branch to be released is green at https://ci.openmrs.org/browse/TRUNK and https://ci.openmrs.org/browse/OP-OPM/branches
8. Do some last minute rudimentary testing as described in Testing Releases on snapshot artifacts published by CI.
9. Release openmrs-core from CI by running the *Release* stage of the latest build for the corresponding branch at https://ci.openmrs.org/browse/TRUNK e.g. https://ci.openmrs.org/browse/TRUNK-OC2/latest (click the Play icon next to the Release stage on the left menu)
10. Once the release build turns green, release the corresponding platform branch from CI at https://ci.openmrs.org/browse/OP-OPM/branches. Click on the green branch build number and run the Release stage. Override the following vatiables; "**maven.release.version**" variable to make sure it matches the release version of core (for example 2.3.0-alpha, 2.3.0-beta or 2.3.0), "**maven.development.version**" to the correct SNAPSHOT if the version being release is a prerelease (for example 2.3.0-alpha, 2.3.0-alpha.2..., 2.3.0-beta etc.), "**build.release.type**" to "**prereleases**" or "**releases**" depending on whether the version to release is a prerelease or full release respectively so that the released war(s) and read-me(s) are uploaded to the correct release folders on sourceforge. Remember to update UAT Platform server instance by running the UAT Platform Deployment plan.
11. In the file listing page on Sourceforge, open the recently released version, click the info "i" button next to the standalone to expand the details of the uploaded ZIP file. For "Download Button", type "OpenMRS Platform 2.0.5 Standalone". Under "Default Download For:" check all the option boxes. Click "Save" to save the settings.
12. Create the release notes page as a child of Platform Release Notes and include JIRA filters showing all closed tickets (e.g. Platform Release Notes 1.11.1)
    - Use Finding Database Changes Between Releases to put the db changes into the release notes page.
    - Edit the Platform Release Notes page to include as content the new release notes page you just created, and updated it to give the new version number.
    - Update the Releases page to include the new platform release, (replacing the prior version in the release line with this release)
    - Update and modify the Downloads wiki page with the latest release notes page.
13. Create a help desk case with https://help.openmrs.org/customer/portal/emails/new **at a minimum of one working day prior to the scheduled release providing download links. *DO NOT* *mark Impact as anything other than "Routine."***
    Use the following template for description (**replace 2.0.5** with the correct version, **the release date & time** and **the JIRA url**, which you can determine from https://issues.openmrs.org/browse/TRUNK/?selectedTab=com.atlassian.jira.jira-projects-plugin:versions-panel):

```
In preparation for the release of OpenMRS Platform 2.0.5 on 21/04/2017 10:00 UTC please do:
1. Update http://openmrs.org/help/report-a-bug/ to allow bug reports for Platform 2.0.5.
(Do not include the rest, if it is not the latest LTE branch release)
2. Update http://openmrs.org/download/ to point to Platform 2.0.5 standalone at http://sourceforge.net
/projects/openmrs/files/releases/OpenMRS_Platform_2.0.5/openmrs-standalone-2.0.5.zip/download
3. Update http://openmrs.org/download/ to point to Platform 2.0.5 war at http://sourceforge.net/projects
/openmrs/files/releases/OpenMRS_Platform_2.0.5/openmrs.war/download
```

14. Create a new case at OpenMRS helpdesk requesting to update the OpenMRS.org home page and downloads page to the latest release. Please provide links to the downloads.
15. Write a post that announces the released on Talk under Software category.
16. With the exception of pre-releases, you need to release the version in jira, you will probably need admin privileges to do this, you can ask for them through the help desk or ask somebody with the privileges to help you release it in JIRA.
    a. Go to JIRA
    b. Click the **Projects** menu item and select **OpenMRS Core (TRUNK)** from the dropdown.
    c. You should see a menu on the left panel on the page that gets loaded, select **Releases.**
    d. In the **Actions** column click the row matching the version you're releasing and select **Release** from the little pop up menu.
    e. Set the release date and click the **Release** button, note that you might have to move any unresolved issues to the next release.

Troubleshooting (when not using CI for releasing)

- When running: `mvn release:prepare` You may also get an error message in validating the generated JavaDoc files, which you can resolve by adding `-Darguments="-Dmaven.javadoc.skip=true"` to skip the validation process. However this does not work with Java 8

- When errors occur while running: `mvn release:prepare` you may want to rollback the release process to fix them, then use `mvn release:rollback`. However this has to be followed by the manual deletion of the tag that was created in the local git environment.
- There are also times when you may get this error message: [ERROR] Failed to execute goal org.apache.maven.plugins:maven-release-plugin:2.1:prepare (default-cli) on project openmrs: Can't release project due to non released dependencies : This is caused by having SNAPSHOT dependencies (outside the current build) which should be released before you can release the project. Their released artifacts should be deployed to Nexus (through release process or manually if necessary). Then the pom.xml should be updated with the release versions and the trunk pom.xml should be bumped to their next SNAPSHOT versions, if development is continuing.

- When running: `mvn clean deploy --batch-mode` You may get an error message like this: Failed to execute goal org.apache.maven.plugins:maven-deploy-plugin:2.5:deploy (default-deploy) on project openmrs: Failed to deploy artifacts: Could not transfer artifact ..... Failed to transfer file ..... Return code is: 401 -> [Help 1]. Solving this requires you to have a nexus repository account, under the openmrs-repo-releases section, in your maven settings.xml file which should be in your MAVEN_HOME folder. If this file does not already exist, you need to create one. See the example maven settings.xml file attached: settings-example-mvnrepo.xml.

- When running: `mvn clean deploy --batch-mode` You may also get an error message like: Failed to transfer file: ..... Return code is: 400 The reason is that the Maven nexus release repo does not allow redeployment by default. To enable that, you could logon nexus and select the release repo, then change the configuration to allow redeployment there. But be aware that any projects using the release artifact will ONLY download the artifact once and thus the artifact you redeployed may not be able to be consumed by the projects, which may cause a lot of troubles. The solution is to manually remove the cached release artifact under .m2/repository so that Maven can download the new one. Because of the disadvantage for the above approach, i just logged onto the nexus repository and manually deleted the successfully deployed artifacts. Then i was able to deploy again.

- When running: `mvn clean deploy --batch-mode` You may also get an error message like: [ERROR] Java heap space -> [Help 1]. To solve this, increase the memory for maven. On my mac, i just typed this on the same shell terminal where i was deploying from: export MAVEN_OPTS="-Xmx512m -Xms128m -XX:MaxPermSize=512m"

- When you are deploying a module to nexus repository using: mvn clean deploy command You may get an error message like [ERROR] Failed to execute goal org.apache.maven.plugins:maven-deploy-plugin:2.7:deploy (default-deploy) on project...Return code is: 401, ReasonPhrase:Unauthorized. To solve this you need to add a server for the module repo by adding the following with nexus repository credentials to the list of servers in the file /M2_HOME/settings.xml .

```
<server>
   <id>openmrs-repo-modules</id>
   <username>username</username>
   <password>password</password>
</server>
```

Tips

- When doing release testing, creating standalone distributions can be a very quick and easy way.

On this page

- Summary
- Prerequisites
- Prior to Release
- Release Types
    - Alpha Release
    - Beta Releases
        - Testing a Beta Release
    - Release Candidates
    - Final Release
    - Maintenance Release
- Release steps
- Troubleshooting (when not using CI for releasing)
- Tips

Summary

Here's a quick summary of being Platform release manager (from Darius Jazayeri):

1. Pick some tickets and kick others out of the release
2. Keep an eye on those tickets, looking for unassigned & those assigned with no movement, occasionally pinging folks to see what's needed to move things along.
3. For tickets awaiting to be reviewed and whose reviews are small, you can go ahead and review them yourself.
4. Pull together the release (an hour or two of time)
    - Preliminary testing
    - Releasing an alpha release
5. Oversee subsequent releases as described on the Release Process page.

Prerequisites

1. OpenMRS ID
2. GitHub account with write privileges to openmrs-core and openmrs-distro-platform (all /dev/4 and /dev/5, but can be granted temporarily on request)
3. JIRA privileges for administering the OpenMRS Core project
4. Bamboo account,

If you need GitHub/JIRA privileges, Bamboo account, open a ticket to request it**.**

Prior to Release

Prior to any release make an initial decision and post to Talk on what will be included in the release.

In most cases, this should be fairly clear from the Road Map; however, some projects may be a little behind schedule and a decision made whether they can be properly resourced/sped up to meet the release timeline or need to be bumped to a later release. You do not need to do all of the work of making these decision (the community should do that), but you should help ensure that the discussion occurs and you are the final arbiter for these decisions.

Features that have been committed, but are not going to be in the release need to be removed from the code.

If necessary, hold a Release Prioritization Meeting to prioritize tickets and finalize, which will be included in the release.

Try as much as possible to shedule

Release Types

Alpha Release

Before an alpha release, all New Feature tickets should either be Closed or at least in a Post Commit state. The same goes for Blocker tickets.

An alpha is a feature-complete release, but is not yet verified to be bug free.

An alpha version is indicated by appending "-alpha" to the version e.g. "2.2.0-alpha".

A release is ready to go from alpha to beta when preliminary testing has been completed and any bugs squashed (a minimum of rudimentary testing, ideally as many steps in the complete testing list as possible). If large bugs are found and squashed, "alpha.2", "alpha.3", etc. may be released to aid testers as needed.

Releasing additional alpha releases is up to the release manager. Going from alpha to beta does not require community discussion and is up to the release manager, but should be done only after the release manager has confirmed with testers that all known bugs have been squashed.

Beta Releases

A beta is a release that is ready to be tested by a larger group of people. Obvious bugs found in the alpha have been found and fixed.

A beta version is indicated by appending "-beta" to the version e.g. "2.2.0-beta".

Be sure to examine all outstanding tickets for the release milestone. By the time a beta release is being considered, all of the tickets assigned to the release milestone should have a clear trajectory (either be moved to a later release or have plans in place to address the issue by the full release).

Testing a Beta Release

- Rudimentary testing should have already been done when creating the beta version.
- Recruit volunteers to help test the beta version, including at least one implementation. Ideally, testers would run through all items in the list of application-level tests "More Complete Testing Process" on the Testing Releases page. It may help to copy this list to a shared document (e.g., Google doc or Wave) and have testers strike out tests that have been completed.
- Consider adding to the More Complete Testing Process list to cover any bugs discovered at the application level (that may not be completely covered by unit testing) which are not currently included in that list.

Going from beta to a release candidate or full release requires that testing have been completed and the system formally exercised by at least one (ideally two or more) implementation.

The decision to go from beta to release candidate or final release should be led by the release manager but agreed by the developer community (via mailing list or dev call).

Release Candidates

A release candidate is only needed when non-trivial changes were required during the beta phase. If the beta release was tested and no significant changes were needed, then you can proceed directly to a full release.

A release candidate version is indicated by appending "-rc" to the version e.g. "2.2.0-rc".

Going from release candidate to full release requires that testing have been completed and the system formally exercised by at least two or more implementations. The decision to go from release candidate to final release should be led by the release manager but agreed by the developer community (via Talk).

Final Release

A true "Release" is deemed tested and worthy of production environments.

Review all existing tickets. Be certain that all tickets assigned to this release milestone have been addressed and that there aren't any straggling tickets for this milestone or previous milestones.

Maintenance Release

A maintenance release contains bug fixes and security patches for use between major releases, e.g., from 1.8.0 to 1.8.1. There is no branch to create. You just pick up where you left off in the current minor version series release branch.

Release steps

There's a number of steps to follow when doing an alpha/beta/rc/final/maintenance release. Please also refer to instructions for different types of releases (alpha/beta/rc/final/maintenance), which you can find in the next paragraphs.

If it is not a maintenance release, the release process should start from creating a maintenance branch.

1. Create a new maintenance branch for openmrs-core and openmrs-distro-platform.
    a. Go to https://github.com/openmrs/openmrs-core.
    b. Click the "branch" dropdown.
    c. Type the new maintenance branch name e.g. "2.2.x".
    d. Click "Create branch '2.2.x' from master".

    e. Go to https://github.com/openmrs/openmrs-distro-platform.
    f. Repeat the steps from b to d.
2. Create a new CI plan for openmrs-core and openmrs-distro-platform.
    a. Go to https://ci.openmrs.org.
    b. Click the "Create" dropdown at the top of the page and choose "Clone an existing plan".
    c. Select "OpenMRS Core Master" as plan to clone and give it e.g. "OpenMRS Core 2.2.x" name.
    d. Edit the newly created plan by going to the "Repository" tab and change the branch of openmrs-core repo to the newly created one e.g. "2.2.x".
    e. Click the "Create" dropdown at the top of the page and choose "Clone an existing plan".
    f. Select "OpenMRS Platform Master" as plan to clone and give it e.g. "OpenMRS Platform 2.2.x" name.
    g. Edit the newly created plan by going to the "Repository" tab and change the branch of openmrs-platform repo to the newly created one e.g. "2.2.x".
3. Announce the upcoming Unsupported Releases (EOL) on Talk under the Software category for the line of two releases back. (e.g. if you're releasing 1.7.0, announce EOL for 1.4.x)

Steps to follow for all releases:

1. Write a post that announces the release date and modules to be included on Talk under the Software category. Ask, if there is a need to release modules in order to include latest changes before the platform release.
2. Verify that there are no unreviewed open tickets against the version to be released. You can use this query https://goo.gl/WXqrI3 (adjust the fixVersion).
    a. Review/close tickets that are in Post Commit Review.
    b. Look carefully at In Progress tickets to see, if some of them have not been committed to a branch (back ported) already. You need to review/close such tickets as well. You can easily check, if any code was committed by searching for an issue id on a GitHub comparison page at https://github.com/openmrs/openmrs-core/compare/2.0.4...2.0.x (adjust the versions).
    c. Move tickets to the next version that are In Progress or in Pre-Commit Review stage (use Manage Versions admin page for bulk moving of tickets, see the 4.a. step below).
3. Create a new version for the next release. Using the Manage Versions in https://tickets.openmrs.org/plugins/servlet/project-config/TRUNK.
4. Release the current version using the above admin page.
    a. When you release a version it will allow you to move non-closed tickets to the next version.
5. Download the latest CIEL for OpenMRS 1.9.x (use 1.9.x regardless of the release version) as described here. Upload the version to our maven repository to be used in standalone by running (adjust the version and the file parameters to match the downloaded version of CIEL):

```
mvn deploy:deploy-file -DgroupId=org.openmrs.contrib -DartifactId=ciel-dictionary -Dversion=1.9.9-
20170409 -Dpackaging=zip -Dfile=openmrs_concepts_1.9.9_20170409.sql.zip -DrepositoryId=openmrs-repo-
contrib -Durl=https://mavenrepo.openmrs.org/nexus/content/repositories/contrib
```

6. Update the CIEL version in standalone's pom.
7. Check if the branch to be released is green at https://ci.openmrs.org/browse/TRUNK and https://ci.openmrs.org/browse/OP-OPM/branches
8. Do some last minute rudimentary testing as described in Testing Releases on snapshot artifacts published by CI.
9. Release openmrs-core from CI by running the *Release* stage of the latest build for the corresponding branch at https://ci.openmrs.org/browse/TRUNK e.g. https://ci.openmrs.org/browse/TRUNK-OC2/latest (click the Play icon next to the Release stage on the left menu)
10. Once the release build turns green, release the corresponding platform branch from CI at https://ci.openmrs.org/browse/OP-OPM/branches. Click on the green branch build number and run the Release stage. Override the following vatiables; "**maven.release.version**" variable to make sure it matches the release version of core (for example 2.3.0-alpha, 2.3.0-beta or 2.3.0), "**maven.development.version**" to the correct SNAPSHOT if the version being release is a prerelease (for example 2.3.0-alpha, 2.3.0-alpha.2..., 2.3.0-beta etc.), "**build.release.type**" to "**prereleases**" or "**releases**" depending on whether the version to release is a prerelease or full release respectively so that the released war(s) and read-me(s) are uploaded to the correct release folders on sourceforge.
11. In the file listing page on Sourceforge, open the recently released version, click the info "i" button next to the standalone to expand the details of the uploaded ZIP file. For "Download Button", type "OpenMRS Platform 2.0.5 Standalone". Under "Default Download For:" check all the option boxes. Click "Save" to save the settings.
12. Create the release notes page as a child of Platform Release Notes and include JIRA filters showing all closed tickets (e.g. Platform Release Notes 1.11.1)
    - Use Finding Database Changes Between Releases to put the db changes into the release notes page.
    - Edit the Platform Release Notes page to include as content the new release notes page you just created, and updated it to give the new version number.
    - Update the Releases page to include the new platform release, (replacing the prior version in the release line with this release)
    - Update and modify the Downloads wiki page with the latest release notes page.
13. Create a help desk case with https://help.openmrs.org/customer/portal/emails/new **at a minimum of one working day prior to the scheduled release providing download links. *DO NOT* *mark Impact as anything other than "Routine."***
    Use the following template for description (**replace 2.0.5** with the correct version, **the release date & time** and **the JIRA url**, which you can determine from https://issues.openmrs.org/browse/TRUNK/?selectedTab=com.atlassian.jira.jira-projects-plugin:versions-panel):

```
In preparation for the release of OpenMRS Platform 2.0.5 on 21/04/2017 10:00 UTC please do:
1. Update http://openmrs.org/help/report-a-bug/ to allow bug reports for Platform 2.0.5.
(Do not include the rest, if it is not the latest LTE branch release)
2. Update http://openmrs.org/download/ to point to Platform 2.0.5 standalone at http://sourceforge.net
/projects/openmrs/files/releases/OpenMRS_Platform_2.0.5/openmrs-standalone-2.0.5.zip/download
3. Update http://openmrs.org/download/ to point to Platform 2.0.5 war at http://sourceforge.net/projects
/openmrs/files/releases/OpenMRS_Platform_2.0.5/openmrs.war/download
```

14. Create a new case at OpenMRS helpdesk requesting to update the OpenMRS.org home page and downloads page to the latest release. Please provide links to the downloads.
15. Write a post that announces the released on Talk under Software category.

16. With the exception of pre-releases, you need to release the version in jira, you will probably need admin privileges to do this, you can ask for them through the help desk or ask somebody with the privileges to help you release it in JIRA.
    a. Go to JIRA
    b. Click the **Projects** menu item and select **OpenMRS Core (TRUNK)** from the dropdown.
    c. You should see a menu on the left panel on the page that gets loaded, select **Releases.**
    d. In the **Actions** column click the row matching the version you're releasing and select **Release** from the little pop up menu.
    e. Set the release date and click the **Release** button, note that you might have to move any unresolved issues to the next release.

Troubleshooting (when not using CI for releasing)

- When running: `mvn release:prepare` You may also get an error message in validating the generated JavaDoc files, which you can resolve by adding `-Darguments="-Dmaven.javadoc.skip=true"` to skip the validation process. However this does not work with Java 8
- When errors occur while running: `mvn release:prepare` you may want to rollback the release process to fix them, then use `mvn release:rollback`. However this has to be followed by the manual deletion of the tag that was created in the local git environment.
- There are also times when you may get this error message: [ERROR] Failed to execute goal org.apache.maven.plugins:maven-release-plugin:2.1:prepare (default-cli) on project openmrs: Can't release project due to non released dependencies : This is caused by having SNAPSHOT dependencies (outside the current build) which should be released before you can release the project. Their released artifacts should be deployed to Nexus (through release process or manually if necessary). Then the pom.xml should be updated with the release versions and the trunk pom.xml should be bumped to their next SNAPSHOT versions, if development is continuing.
- When running: `mvn clean deploy --batch-mode` You may get an error message like this: Failed to execute goal org.apache.maven.plugins:maven-deploy-plugin:2.5:deploy (default-deploy) on project openmrs: Failed to deploy artifacts: Could not transfer artifact ..... Failed to transfer file ..... Return code is: 401 -> [Help 1]. Solving this requires you to have a nexus repository account, under the openmrs-repo-releases section, in your maven settings.xml file which should be in your MAVEN_HOME folder. If this file does not already exist, you need to create one. See the example maven settings.xml file attached: settings-example-mvnrepo.xml.
- When running: `mvn clean deploy --batch-mode` You may also get an error message like: Failed to transfer file: ..... Return code is: 400 The reason is that the Maven nexus release repo does not allow redeployment by default. To enable that, you could logon nexus and select the release repo, then change the configuration to allow redeployment there. But be aware that any projects using the release artifact will ONLY download the artifact once and thus the artifact you redeployed may not be able to be consumed by the projects, which may cause a lot of troubles. The solution is to manually remove the cached release artifact under .m2/repository so that Maven can download the new one. Because of the disadvantage for the above approach, i just logged onto the nexus repository and manually deleted the successfully deployed artifacts. Then i was able to deploy again.
- When running: `mvn clean deploy --batch-mode` You may also get an error message like: [ERROR] Java heap space -> [Help 1]. To solve this, increase the memory for maven. On my mac, i just typed this on the same shell terminal where i was deploying from: export MAVEN_OPTS="-Xmx512m -Xms128m -XX:MaxPermSize=512m"
- When you are deploying a module to nexus repository using: mvn clean deploy command You may get an error message like [ERROR] Failed to execute goal org.apache.maven.plugins:maven-deploy-plugin:2.7:deploy (default-deploy) on project...Return code is: 401, ReasonPhrase:Unauthorized. To solve this you need to add a server for the module repo by adding the following with nexus repository credentials to the list of servers in the file /M2_HOME/settings.xml .

```
<server>
   <id>openmrs-repo-modules</id>
   <username>username</username>
   <password>password</password>
</server>
```

Tips

- When doing release testing, creating standalone distributions can be a very quick and easy way.