XFree86® Version Numbering Schemes

The XFree86 Project, Inc

6 February 2005

Abstract

The version numbering schemes used by XFree86 have changed from time to time. The schemes used since version 3.3 are explained here.

1. Development, Branches, and Releases

XFree86 development happens on the trunk of the CVS repository with releases cut from this development trunk at the end of each release cycle. A maintenance branch is then created sometime around each release, which can be used for important post-release fixes and security updates.

XFree86 release cycles occur approximately every 12 months and coincide with the calendar year. For example 4.0 was released in 2000; 4.1 in 2001; 4.2 in 2002; 4.3 in 2003; 4.4 in 2004 and 4.5 in 2005. So it is safe to say that 4.6 will be released early in 2006.

The development phase typically ends 1-3 months before the release date and is marked by the start of a **feature freeze** . The timing of releases and the length of the release cycles may vary according to the resources available to The XFree86 Project, an entirely volunteer organisation.

XFree86 consists of full source code tarballs, plus full binary distributions for a broad range of supported platforms. Update/bugfix releases may be made from time to time as required and as resource availability allows. Such update releases typically consist of source code patches plus binary updates that may be layered on top of the previous release.

The current release is 4.6.0, and the next release will be 4.7.0. No update release is scheduled, but if one is needed it will be version 4.6.1.

Aside from releases, snapshots of the development trunk are tagged in the CVS repository at regular intervals, normally every two weeks. Each snapshot has an identifiable sequential version number. Further information about the latest snapshot can be found at the XFree86 snapshots page <URL:http://www.xfree86.org/develsnaps/>.

2. Current Version Numbering Scheme

Starting with the main development branch after 4.0.2, the XFree86 versions are numbered according to the scheme outlined here. Both the 4.0.2 stable branch and the 3.3.x legacy branch continue to use the previous scheme, which is outlined in the sections below.

The version numbering format is `M.m.P.s`, where `M` is the major version number, `m` is the minor version number, `P` is the patch level, and `s` is the snapshot number. Full releases have `P` set to

zero, and it is incremented for each subsequent bug fix release on the post-release stable branch. The snapshot number `s` is present only for between-release snapshots of the development and stable branches.

2.1 Development Branch

Immediately after forming a release stable branch, the patch level number for the main development branch is bumped to 99, and the snapshot number is reset. The snapshot number is incremented for each tagged development snapshot. The CVS tag for snapshots is "`xf-M_m_P_s`". When the development branch enters the feature freeze, the snapshot number is bumped to 901, and the snapshot number is incremented sequentially until the release is finalised. Snapshots made during the feature freeze are called "release candidates," and are numbered by the snapshot number minus 900. An example of this is snapshot number 901 would be release candidate 1 (RC1).

When the release is finalised, the minor version is incremented, the patch level is reset to zero, and the snapshot number removed. Somewhere around the time of the release, a maintenance branch is created. This branch is usually called "`xf-M_m-branch`".

Here is an example which shows the version number sequence for the development leading up to version 4.4.0:

`4.3.99.1`
: The first snapshot of the pre-4.4 development trunk.

`4.3.99.8`
: The eighth snapshot of the pre-4.4 development trunk.

`4.3.99.901`
: The start of the 4.4 feature freeze, and first 4.4.0 release candidate.

`4.3.99.903`
: The third 4.4.0 release candidate (RC3).

`4.4.0`
: The 4.4.0 release.

`4.4.99.1`
: The first pre-4.5 development snapshot.

2.2 Maintenance Branch

After a full release, the maintenance branch for the release will be be created to hold important bug fixes and updates. Snapshots and release candidates may be tagged on maintenance branches from time to time. When an update release is cut, the patch level value (`P`) is incremented and the snapshot number dropped.

Here is an example which shows the version number sequence for the 4.2.x maintenance branch.

`4.2.0.1`
: The first snapshot on the 4.2.x maintenance branch.

`4.2.1`
: The 4.2.1 update release.

3. Version Numbering Scheme for XFree86 4.0.x.

The version numbering format for XFree86 4.0.x releases is `M.m.nx`, where `M` is the major version number (4), `m` is the minor version number (0), `n` is the sub-minor version number, and `x` is a letter. Full release versions up to and including 4.0.2 were 4.0, 4.0.1, and 4.0.2. Between-release snapshots are indicated by including `x`, a lower case letter. For example, the first post-4.0.1

snapshot was 4.0.1a. Release candidates have been indicated by setting `x` to a one or two letter combination with the first letter being "Z". For example, 4.0.1Z was the first 4.0.2 release candidate.

The next 4.0.x release will be an update release, not a full release. These update releases will be indicated by incrementing the sub-minor version number. So, the first post-4.0.2 update release will be 4.0.3. Between-release snapshots will continue to be indicated with a lower case letter, so the first pre-4.0.3 snapshot will be 4.0.2a.

The following example illustrates the release sequence from 4.0 through to the post-4.0.2 update releases.

`4.0`
: The 4.0 release.

`4.0a`
: The first post-4.0 development snapshot.

`4.0f`
: The sixth post-4.0 development snapshot.

`4.0Z`
: The 4.0.1 release candidate.

`4.0.1`
: The 4.0.1 release.

`4.0.1a`
: The first post-4.0.1 development snapshot.

`4.0.1f`
: The sixth post-4.0.1 development snapshot.

`4.0Z`
: The first 4.0.2 release candidate.

`4.0Zb`
: The third 4.0.2 release candidate.

`4.0.2`
: The 4.0.2 release.

`4.0.2a`
: The first pre-4.0.3 snapshot/release candidate.

`4.0.2c`
: The third pre-4.0.3 snapshot/release candidate.

`4.0.3`
: The 4.0.3 update release.

`4.0.3a`
: The first pre-4.0.4 snapshot/release candidate.

`4.0.4`
: The 4.0.4 update release.

4. Pre-4.0 Development Versions

This section is included mostly for historical reasons.

The development leading up to 4.0 started from version 3.2A, but much of it happened on a separate development branch. The "new design" work on that development branch was first folded

into the main development branch at version 3.9N. Up until the XFree86 CVS was made publicly available, all versions containing one or more letters were internal development snapshots. The internal development snapshots continued through the following sequence: 3.9N, 3.9Na, ..., 3.9Nz, 3.9P, 3.9Pa, ..., 3.9Py, 3.9.15, 3.9.15a, ..., 3.9.16, 3.9.16a, ..., 3.9.17, 3.9.17a, ..., 3.9.18, 3.9.18a, ..., 4.0. The 3.9.15, 3.9.16, etc versions were public pre-4.0 beta releases.

5. Version Numbering Scheme for XFree86 3.3.x.

The version numbering format for XFree86 3.3.x releases is `M.m.nx`, where `M` is the major version number (3), `m` is the minor version number (3), `n` is the sub-minor version number, and `x` is a letter. Between-release snapshots are indicated by including `x`, a lower case letter. An exception to this scheme was the 3.3.3.1 release, which was an update to the 3.3.3 release.

`3.3`
: The 3.3 release.

`3.3a`
: The first post-3.3 development snapshot.

`3.3.1`
: The 3.3.1 release.

`3.3.1a`
: The first post-3.3.1 development snapshot.

`3.3.2`
: The 3.3.2 release.

`3.3.2a`
: The first post-3.3.2 development snapshot.

`3.3.3`
: The 3.3.3 release.

`3.3.3a`
: The first post-3.3.3 development snapshot.

`3.3.3.1`
: The 3.3.3.1 release.

`3.3.3.1a`
: The first post-3.3.3.1 development snapshot.

`3.3.4`
: The 3.3.4 release.

`3.3.4a`
: The first post-3.3.4 snapshot.

`3.3.5`
: The 3.3.5 release.

`3.3.5a`
: The first post-3.3.5 snapshot.

`3.3.6`
: The 3.3.6 release.

`3.3.6a`
: The first post-3.3.6 snapshot.

6. Finding the XFree86 X Server Version From a Client

The XFree86 X servers report a `VendorRelease` value that matches the XFree86 version number. There have been some cases of releases where this value wasn't set correctly. The rules for interpreting this value as well as the known exceptions are outlined here.

For 3.3.x versions, the `VendorRelease` value is `Mmnp`. That is, version `M.m.n.p` has `VendorRelease` set to `M * 1000 + m * 100 + n * 10 + p`. Exceptions to this are: The value wasn't incremented for the 3.3.3.1 release, and for the 3.3.4 and 3.3.5 releases the value was incorrectly set to `Mmn` (`M * 100 + m * 10 + n`). This was corrected for the 3.3.6 release.

For versions 3.9.15 to 4.0.x, the `VendorRelease` value is `Mmnn`. That is, version `M.m.n` has `VendorRelease` set to `M * 1000 + m * 100 + n`. There have been no exceptions to this rule.

For post-4.0.2 development and release versions using the new numbering scheme, the `VendorRelease` value is `MMmmPPsss`. That is, version `M.m.P.s` has `VendorRelease` set to `M * 10000000 + m * 100000 + P * 1000 + s`. Note: 4.0.3 and any other 4.0.x releases will continue with the `Mmnn` scheme.

The following is a code fragment taken from `xdpyinfo.c` that shows how the `VendorRelease` information can be interpreted.

```
if (strstr(ServerVendor(dpy), "XFree86")) {
    int vendrel = VendorRelease(dpy);

    printf("XFree86 version: ");
    if (vendrel < 336) {
        /*
         * vendrel was set incorrectly for 3.3.4 and 3.3.5, so handle
         * those cases here.
         */
        printf("%d.%d.%d", vendrel / 100,
                           (vendrel / 10) % 10,
                            vendrel       % 10);
    } else if (vendrel < 3900) {
        /* 3.3.x versions, other than the exceptions handled above */
        printf("%d.%d", vendrel / 1000,
                        (vendrel /  100) % 10);
        if (((vendrel / 10) % 10) || (vendrel % 10)) {
            printf(".%d", (vendrel / 10) % 10);
            if (vendrel % 10) {
                printf(".%d", vendrel % 10);
            }
        }
    } else if (vendrel < 40000000) {
        /* 4.0.x versions */
        printf("%d.%d", vendrel / 1000,
                        (vendrel /   10) % 10);
        if (vendrel % 10) {
            printf(".%d", vendrel % 10);
        }
    } else {
        /* post-4.0.x */
        printf("%d.%d.%d", vendrel / 10000000,
                           (vendrel /   100000) % 100,
                           (vendrel /     1000) % 100);
        if (vendrel % 1000) {
            printf(".%d", vendrel % 1000);
        }
    }
}
```

CONTENTS

1. Development, Branches, and Releases 1
2. Current Version Numbering Scheme 1
   2.1 Development Branch 2
   2.2 Maintenance Branch 2
3. Version Numbering Scheme for XFree86 4.0.x. 2
4. Pre-4.0 Development Versions 3
5. Version Numbering Scheme for XFree86 3.3.x. 4
6. Finding the XFree86 X Server Version From a Client 5

$XFree86: xc/programs/Xserver/hw/xfree86/doc/sgml/Versions.sgml,v 1.8 2005/03/10 02:35:08 dawes Exp $