XFree86 Version Numbering Schemes

The XFree86 Project, Inc

16 January 2002

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. Releases, Development Streams and Branches

As of the release of version 4.0.2 in December 2000, XFree86 has three release branches. First is trunk of the CVS repository. This is the main development stream, where all new work and work for future releases is done.

Second is the stable bugfix branch for the latest full release (4.2.1). It is created around the time of the release. The branch for this one is called "`xf-4_2-branch`". Fixes for bugs found in the release will be added to this branch (as well as the trunk), and updates to this release (if any) will be cut from this branch. Similar stable branches are present for previous full releases.

Finally there is the 3.3.x legacy branch, which is called "`xf-3_3-branch`". While this branch is not actively being maintained, it does include some important post-3.3.6 bug fixes and security updates. Relevant security updates in particular are usually back-ported to this branch.

XFree86 is planning to make full releases from the main development stream approximately every six months, in late May and November of each year. The feature freezes for these releases will be 1 April and 1 October respectively. These are target dates, not a binding commitment. How effectively these dates can be met will depend to a large degree on the resource available to XFree86. Full releases consist of full source code tarballs, plus full binary distributions for a range of supported platforms. Update/bugfix releases will be made on an as-required basis, depending also on the availability of resources. Update/bugfix releases will not be full releases, and will consist of source code patches, plus binary updates to be layered on top of the previous full release.

The next full release will be version 4.3.0, tentatively scheduled for late late 2002. There is no scheduled update release. If there is one, the version will be 4.2.2.

Aside from actual releases, snapshots of the active release branches are tagged in the CVS repository from time to time. Each such snapshot has an identifiable version number.

2. Current (new) 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 feature freeze, the snapshot number may be bumped to 900, and a stable branch may be created for the next full release. The branch is called "`xf-M_m-branch`". The snapshot number is incremented from there until the release is finalised. Each of these snapshots is a "release candidate". When the release is finalised, the minor version is incremented, the patch level is set to zero, and the snapshot number removed.

Here's an example which shows the version number sequence for the development leading up to version 4.1.0:

`4.0.99.1`
: The first snapshot of the pre-4.1 development branch.

`4.0.99.23`
: The twenty-third snapshot of the pre-4.1 development branch.

`4.0.99.900`
: The start of the 4.1 feature freeze, which marks the creation of the "`xf-4_1-branch`" branch. That branch is the "stable" branch for the 4.1.x releases.

`4.0.99.903`
: The third 4.1.0 release candidate.

`4.1.0`
: The 4.1.0 release.

`4.1.99.1`
: The first pre-4.2 development snapshot, which is the first main branch snapshot after creating the 4.1 stable branch.

2.2 Stable Branch

After a full release, the stable branch for the release will be maintained with bug fixes and important updates until the next full release. All snapshots on this branch are considered "release candidates", so the first is indicated by setting `s` to 901. The snapshot number is then incremented for each subsequent release candidate until the update release if finalised. The patch level value (`P`) is incremented for each update release.

Here's an example which shows the version number sequence for the 4.1.x stable branch.

`4.0.99.900`
: The start of the 4.1 feature freeze, which marks the creation of the "`xf-4_1-branch`" branch. That branch is the "stable" branch for the 4.1.x releases.

`4.0.99.903`
: The third 4.1.0 release candidate.

`4.1.0`
: The 4.1.0 release.

`4.1.0.901`
The first pre 4.1.1 snapshot.

`4.1.0.903`
The third pre 4.1.1 snapshot, also known as the third 4.1.1 release candidate.

`4.1.1`
The 4.1.1 release.

`4.1.1.901`
The first pre 4.1.2 snapshot.

`4.1.2`
The 4.1.2 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. Releases, Development Streams and Branches 1
2. Current (new) Version Numbering Scheme 1
   2.1 Development Branch 2
   2.2 Stable Branch 2
3. Version Numbering Scheme for XFree86 4.0.x. 3
4. Pre-4.0 Development Versions 4
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 1.2 2002/01/16 20:38:45 dawes Exp $