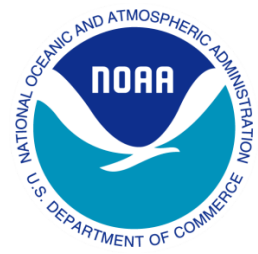




National Centers for
Environmental Prediction
NCEP Central Operations

Software Version Numbering v2.0.0



Disclaimers

The information contained in this document is subject to change without notice.

Who Should Read This Document?

This guide is designed to benefit the following professionals:

System/Subsystem Owners and Team Members

This document will help the system owners apply the rules for version numbering consistently across all operational software within NCEP Central Operations

Configuration Managers

This document will help the configuration managers enforce the rules for version numbering consistently across all operational software.

Document History

Paper copies are valid only on the day they are printed. Contact the author if you are in any doubt about the accuracy of this document.

Revision History

Revision Number	Revision Date	Summary of Changes	Author
1	5/10/2010	Initial Document	Scott Jacobs
2	6/29/2010	Add Software Units and Initial Version Numbers	Scott Jacobs
3	7/2/2010	Update Mesoscale Branch information	Geoff DiMego

4	7/2/2010	Update Global Branch information	John Ward
5	7/6/2010	Update NCO library information	Scott Jacobs
6	7/7/2010	Update Marine Branch information	Hendrik Tolman
7	7/7/2010	Re-order sections	Scott Jacobs
8	7/28/2010	Sponsor corrections and suggestions added	Scott Jacobs
9	7/29/2010	Added "Major" Systems to the software unit tables	Scott Jacobs
10	8/18/2017	Added "pre-release" version number Make generic to NCO instead of WCOSS specific Removed section 3&4	Steven Earle

Approvals

This document requires following approvals:

Name	Title
Ben Kyger	Director, NCEP Central Operations

Table of Contents

Introduction	5
Purpose of this document	5
Identification	5
Version Numbering Scheme	5
Overall Scheme	5
Major Version	6
Minor Version	7
Maintenance Version	7
Pre-Release Version	7
Summary	7

1. Introduction

1.1 Purpose of this document

This document covers the scheme used to number the software units used in production across NCEP Central Operations. This document will also identify the software units and define the boundaries of each unit.

1.2 Identification

- This document applies to all software units currently used in NCEP Central Operations (NCO) and to any future units developed or implemented by NCO.
- The numbering scheme presented in this document is an amalgamation of many schemes used in various software development fields. This document attempts to combine these into a single scheme appropriate to software development at NCEP.

2. Version Numbering Scheme

2.1 Overall Scheme

There are a wide variety of numbering schemes used in software development projects. They are all variations on the same basic concept: keep track of different versions of a piece of software. The vast majority of the version numbering schemes are sequence based. A sequence of numbers, separated by a delimiter, is used to convey the significance of the changes between releases. The

left-most value is changed for the most significant software modifications. Changes to the subsequent values indicate decreasing significance.

Many version numbering schemes are quite complex. One goal with the NCEP scheme is to keep it simple. Therefore, the NCEP scheme for production software is defined as:

```
major.minor.maintenance
```

The NCEP scheme is also defined as only using numeric values for each sequence number. Some version numbering schemes allow for the use of “a” and “b” to denote alpha and beta versions of the software. This usage makes these schemes more complex. Therefore, the use of letters in the NCEP version numbering is not allowed.

For planning purposes, the next version number of a software unit should be defined early in the development process. That is, when a requirement, or set of requirements, forces a change in the software unit, the scope of the development needed should be determined. At this time, the scope will also dictate whether the released software will be a *major*, *minor* or *maintenance* release. If, during development, it becomes apparent that the scope was not properly estimated, the version number may change in the planning documents.

During the development and pre-implementation stage, including interactions between development and NCO, a fourth numerical value must be used in order to keep track of these pre-release changes.

```
major.minor.maintenance.build
```

Because the fourth digit is for internal version tracking, it must be dropped entirely when the software is implemented into operations.

Each of the following sections defines the criteria for increasing the value of each sequence number.

2.2 Major Version

What constitutes a Major Version? Generally, use of the term *major* is very subjective. This section attempts to define the causes of incrementing the major value in the version sequence.

For the numerical prediction systems, changes to the resolution, the underlying model, the model physics or the data assimilation would necessitate incrementing the major version number. These changes have widespread effects on the entire model system being modified. The scope of the change must also play a role in determining the version number. For an entire model system, if only one subsystem is changing, the major number for the encompassing system may not change. However, if many subsystems are modified, then the major version value should increase.

Customer service should always be a focus for any software unit that creates products. Therefore, any time a software change causes a change to the output content or format, the change constitutes a major version.

2.3 Minor Version

Similar to the major value, *minor* is a subjective term. A minor version is defined as a change involving selected areas of a model or changes for IT reasons. The scope of the changes is also less far-reaching than a major update.

Data format changes to incoming data sets fall into the category of minor versions. For example, the data type is already being ingested into the models, but the data supplier announces a change to the format. The decoder would need to be changed in this case, but the changes may be of limited scope.

From a customer perspective, a minor version should present no differences in the content or format of the output.

When a major update is released, the value of the minor version should be reset to zero.

2.4 Maintenance Version

A maintenance version has the least scope. This level of update is reserved for bug fixes or changes to a single routine or file. The maintenance number will usually be increased for solutions to operational problems that cannot wait for a minor or major version.

When a major or minor update is released, the value of the maintenance version should be reset to zero.

2.5 Pre-Release Version

A pre-release version will never be seen in operations. This level of update is reserved for all changes being made prior to an official major, minor or maintenance release of software. This version number is dropped when the software is implemented into operations.

2.6 Summary

Major Version

- Resolution changes
- Underlying model replacement and/or changes
- Major scientific technology upgrade, e.g., semi-lagrangian or physics updates
- Large scope, many subsystems involved
- Update to an international standard
- Table format changes
- Customers can expect to see differences in the output

Minor Version

- Model Physics changes
-

-
- IT-related changes, modifications that do not affect the model or its output, but how the model is executed or how data moves into or out of the model
 - Medium scope, limited number of subsystems or routines
 - Table entry updates
 - Customers should expect to see no significant differences in the output or performance

Maintenance Version

- Bug fixes
- Solutions to operational problems
- Small scope, only a few files are affected by the change
- Updates to data files with an implicit dynamic nature, such as buoy files

Pre-Release Version

- All changes made during pre-operational testing
-