

## CISM Model Release Sequence & Version Control

Pre-release. Model developers may choose the degree to which they use the CISM Software Version Control (SVC) -- CVS or SVN -- as a development tool. Those not using SVC routinely are strongly encouraged to make occasional “updates” to SVC, with assistance as desired from the CISM software engineer (CSE). Only model developers and the CSE have access to SVC; each model’s lead developer will decide who has access to that project.

Version freeze. When a version is ready to be frozen for release and validation, it is tagged in SVC. Naming guidelines are provided below under “Tagging Convention”.

Pre-validation distribution. A *pre-validation distribution*, built from the tagged SVC version, is created in a version-specific location. The distribution is a self-contained source for the necessary elements to create an installation of that model version. The exact content of the distribution is a matter of judgment from case to case. As a minimum it should include a notes file that documents conditions under which successful builds have been tested (e.g. compiler version, operating system, build tools). It may contain shared libraries and build tools.

As part of the distribution, the developer provides a test suite with “pass” criteria for others, e.g. the validation team, to perform basic checks of their installation and initial test runs. The developer also provides a checklist of steps for installation and verification.

Before releasing the pre-validation distribution, the validation team and CSE beta test the code to shake out fatal flaws or bugs. Iterating with the developer the pre-validation distribution is updated if changes are required. Finalization of the pre-validation distribution constitutes “delivery” of the model.

CISM team members may withdraw models from the pre-validation distribution for legitimate CISM purposes, such as validation and scientific use, but may not further distribute any model without permission from the developer(s).

The distribution directories are under the control of the CSE.

Pre-Validation Tests. The validation team performs the following tasks with code from the pre-validation distribution: (1) verifies with test suite; (2) makes initial validation runs; (3) generates user manual; (4) returns completed checklist to developer. After these steps the code may be distributed outside of CISM (e.g. to CCMC) with agreement of the developer(s).

Validation. The validation team does formal validation, writes the validation report, and updates the user manual as required.

Post-Validation distribution. The developer creates a post-validation tag in SVC, freezing any changes (bug fixes, not continuing development) as needed to accurately represent the as-validated version. The CSE creates a post-validation distribution that includes the frozen code, user manual, and validation report.

Tagging Convention. The following format should be used by developers when tagging:

*Project[-Codename]-Version-Stage*

Where *Project* is the name of the project (e.g. WSA), *Codename* is an optional name for this specific version (e.g. Icarus), *Version* being in the format:

*major\_minor[\_revision[\_build]]* (e.g. 2\_8), and where the required *Stage* is one of the following development stages: *devel*, *prevalid*, or *valid*. The stages are described in greater detail as follows:

*devel*: The 'devel' tag signifies that a version is at the pre-release stage of development. This is an optional stage, but may be useful for internal project development by tagging the progress of revisions (e.g. WSA-2\_7\_34\_1-devel, WSA-2\_7\_34\_2-devel, etc).

*prevalid*: When a project enters the pre-validation stage of development it is tagged with 'prevalid' by the project developers or the CSE (e.g. WSA-2\_7\_35-prevalid).

*valid*: After the validation stage is completed the project is tagged as 'valid' by the CSE in coordination with the validation team and developer (e.g. WSA-2\_7\_38-valid).

Model Data File Versioning. Beginning 2006, all delivered codes must provide in their output data files information necessary to identify the model version and to capture all input parameters for each run (i.e., to uniquely identify how the data files was produced).