MODIS Science Software and Product Versioning White Paper - 1.0

Robert Wolfe, MODLAND Support
and
Mike Linda, SDST

December 14, 1998

Introduction

In the past, versioning of science software and products within the Moderate Resolution Imaging Spectroradiometer (MODIS) Science Data Processing System (SDPS) was *ad hoc*. The necessity to have a formal versioning convention is recognized throughout the MODIS Science Team. This white paper proposes a comprehensive framework for versioning MODIS science software and products.

The requirements for a versioning scheme are that it:

1. is easy to understand,
2. provide a unique label for elements in the system that may change,
3. conveys information about the significance of a science algorithm change,
4. conveys information about the version of the system.

Versioning is used in a number of different areas of the MODIS SDPS:

1. system,
2. process,
3. Product Generation Executives (PGE),
4. Earth Science Data Type (ESDT),
5. lookup table,
6. product.

These areas and the relationship between them are shown in Figure 1. In addition, a production history string will be used to convey information within a product about changes to upstream products.

Table 1 contains format of the version numbers and the location of the versioning information within a MODIS product.

Since the MODIS science software will be run both at the GSFC Data Archive and Distribution Center (DAAC) and in the MODIS Principle Investigator Science Data

Processing System (MODAPS) there are some differences in the implementation of the versioning within each system. Most of the versioning is common between these systems and this paper does not talk in detail about these differences.

This white paper does not include a detailed discussion of the impact of implementing the versioning system. Most of the current science code will have to be modified, though the changes are expected to be small and straightforward and so implementation costs are expected to be modest. A software library may be developed by the Science Data Support Team (SDST) that implements individual elements of the versioning framework. Also, once a framework has been adopted, the versioning can be phased in, with the requirement that it be included in any new or updated code delivered from the science team members and their developers.

System Version

The MODIS SDPS software is built from a large number of elements, including: individual processes, PGEs, lookup tables, etc. A hierarchy of these compatible components forms a version of the MODIS SDPS. This system will be labeled with a major and minor version number, such as Version 2.1, where 2 and 1 are the major and minor version numbers respectively. The major version number is established from direction given by the MODIS Science Team Leader. The minor version number is derived from a consensus of the MODIS Science Team Members.

The major version number is used to signify a major milestone in the MODIS software and data processing, such as 2.x is the at-launch software, or 3.x is the software for the first major reprocessing. It is also used to convey information to the Science Computing Facility (SCF) developers about major revisions of the software and to the end user about major changes in the science algorithms. For software, a change in the version will indicate a major change in functionality such as robustness, external interfaces, etc. It is expected that the major version number will not change at an interval of less than once every six months.

The minor system version number will primarily be used to document and label a unique internally-compatible version of the system. Processes and PGEs are built and installed into the system individually or in small groups. Due to the size and complexity of the system, it is unlikely that there will be a single “build” of the system at any one time, but if a unique build were possible, it would be identified as a system that has a unique minor version number. It is expected that a new minor version of the system will occur at an interval of more than three months and less than a year.

When one or more PGEs change in such a way that they cannot function in a given system, a new version of the system will be established and the system minor version

number will be incremented. The new system will be built from the new PGEs and unchanged components from the previous system.

A one-way mapping will be maintained between the system version number and versions of the elements in the system. For each version of the system this will consist of a table which contains the version of each element in the system. For elements that change during the time-span of the system version, the version of the changed element and the date of the change will also be maintained. The system version will not be carried in each element, although individual elements may inherit the major version number (see below).

Process Version

A process is an executable that generates one or more products. It is primarily built from science software provided by the individual SCFs. It is linked with a number of software libraries, including the SDP toolkit library, HDF-EOS library, operating system run-time libraries and, in some cases, other SDST and/or common discipline specific supplied libraries.

The method of identifying versions of a process will depend on the type of change being made to the process software and on the version of the system that the software is part of. The process version will consist of three parts: a major version number, a minor version number, and an update number. For instance, 2.1.3 is major version 2, minor version 1, and update 3.

The major version number will be incremented when significant changes are made to a process either in preparation for, or following, a major project milestone, i.e., when the system version number changes. This allows the resynchronization of the versions of the individual processes with the rest of the system. However, regardless of the project milestone, small or insignificant changes to the software may only result in changes to the update version number. Whenever the major version number is changed, the minor and update version numbers will be reset to zero.

The minor version will be used to keep track of changes in the science algorithms. The user of the product will be able to use this number to determine if a significant change in the science algorithm has taken place. Whenever the minor version number is changed, the update version number will be reset to zero.

The update number will track any other changes, including, minor algorithm changes that handle specific cases and do not effect a significant amount of the data being produced. Changes in error messages and minor changes in the product format also fall into this category.

The decision of whether to increment the minor or update version number is left to the individual scientist/developer. However, a large number of increments in the minor version number, i.e., a large number of changes in the science algorithm, are discouraged because they confuse the end user and make analysis of multi-date data (time series) more difficult.

The effect on the version number of a process caused by changes to software libraries, or other software used by more than a single process, will be handled on a case by case basis. Regardless, the same criteria, i.e. the impact on the science algorithm, will be used to determining whether to update the minor and update version numbers.

The developer and the SDST Configuration Management (CM) will agree on a preliminary version number at the time of delivery of the process to the SDST CM. The SDST Science Software Transfer Group (SSTG) will also provide guidance, based on compatibility with related software, in determining the proper version number. This preliminary version number may change as mini-milestones are reached during the SSTG and Test and Integration processes. The final version number will be assigned before releasing the algorithm to the processing system, DAAC, MODAPS, or to any other organization external to SDST. The intermediate version numbers allow for coordination between SDST and the code developer, especially when the developer is making changes while the software is going through the SDST process.

The process version number will be stored as a string within the process source code. This will prevent any possibility that the process version will become separated from the source. To enable the version to be easily updated, all processes will use the same convention for storing and naming the variable or parameter containing the process version. The specific implementation details for FORTRAN and C will be provided by SDST. An implementation under consideration is that the software for each process contain a separate “include” file named “version.inc” (FORTRAN) or “version.h” (C) containing only the version number and associated definitions.

Each process will write the process version to each of its products as a Product Specific Attribute (PSA) called *ProcessVersion* (“PROCESSVERSION”). Specifically, the process version will be written as a string with no prefix or suffix and no leading zeros, for instance, “2.1.3”. Each field may be multi-digit, for example, 3.12.105. Only positive integers, including zero, will be used. The process version will be implemented as a PSA so that it can be used as a criterion in searches for a particular product, perhaps to find products generated by a particular version of the software.

Some SCFs may specify an algorithm version for individual parameters within a product using a set of PSAs *VerParm_** (“VERPARM_*”), where “*” is the name of an individual parameter. This allows modifications of individual parameter algorithms to be

tracked. The relationship between these parameter version numbers and the process version number will be left to the individual SCF.

PGE Version

A PGE consists of one or more processes, control scripts, and support files. The control scripts and support files tie the processes together and to the production system. The Metadata Configuration Files (MCFs), Process Control File (PCF) templates, production rule Objection Definition Language (ODL) files and PGE profiles, which control the execution of a PGE, are considered part of the PGE for versioning purposes.

The PGE version will be similar to the process version number and will consist of three parts, the major version number, the minor version number, and an update number.

The PGE version number will be loosely tied to the versions of its processes. Whenever the major, minor, or update version number of a PGE's process changes, the corresponding version number in the PGE will change. However, the PGE version number will not be forced to inherit the process version numbers. For instance, when the process minor version number changes, the PGE's minor version number will increment, but not necessarily to the same value.

There are cases where changes can be made to PGE-specific files that do not affect any of the PGE's processes. In such cases the PGE's update version number will change.

There will be a loose relation between the PGE version number and the system version number. In general, the PGE major version number will be identical to the system's major version number. However, if a PGE does not change significantly when the system's major number is incremented, the PGE may retain the lower major version number.

Whenever the major version number is changed, the minor version number will be set to the highest minor version number (typically zero) of the PGE's process with the same major version number. Whenever the PGE's minor version number changes, the update version number will be set to zero. In this way, the combined major/minor version numbers will always be the same as or "higher" than any of the processes' combined major/minor version numbers.

The PGE version number will be read out of the PCF as a run-time parameter. A single LUN will be used for all PGEs. Each process will read the PGE version from the PCF and store it as the ECS inventory Metadata field *PGEVersion* ("PGEVERSION"). Since PCFs are instantiated at runtime, in the MODAPS system the PGE version number will come from a PCF template that is read by the PGE script and put into the PCF.

In the ECS system a PGE's production rule ODL file contains a 5-character field called the PGE version number. This version number will be maintained by the DAAC and will not necessarily be related to the PGE version number maintained by MODIS.

ESDT

There are two parts to versioning of an ESDT, the ECS CM version number and the public version number. The CM version number is used to track internally a specific set of ECS files and programs that are specific to a particular ESDT. The public version number, on the other hand, is used by the ECS system to distinguish between different sub-types of the ESDT and is mainly under the control of the science team.

The CM version number is a designator given to the ESDT by the creator of the specific version of the ESDT, currently the ECS, but eventually the DAAC. It will be used during PGE testing and when the PGE is delivered to the DAAC to match up PGEs with compatible ESDTs. The CM ESDT version will change whenever any change is made to the inventory metadata for a product, when a new service (subsetting, etc.) is needed for a product, or when the SCF requests a change to the public version number.

The public version number is used in a number of ways by the ECS system. As currently implemented it is an integer in the range 0 to 255. It is primarily used to distinguish between different "sets" of a specific product. In the current implementation of the browse and order tool, the user must select both a specific ESDT and a specific version number before a search can be performed. In the future the user may be allowed to search across multiple versions of the ESDT. The public version number may also be used to control the release of different data sets, that is, the control of whether a specific set of products is public or private will be tied to the version of the product.

The public version number is under control of the SCF, with one exception. The exception is that version numbers must be changed when any inventory metadata field is deleted or when a mandatory metadata field is added. Any changes to the archive metadata or changing a metadata field from mandatory to non-mandatory do not necessitate a change in the public version number.

At this time, there is no consensus on how the ESDT public version number will be used by the MODIS science team. Every attempt will be made to minimize changes to the public version number to minimize the confusion of the user over multiple versions of the products.

The public ESDT version number is stored in the MCF and included in the product as the ECS inventory metadata field *VersionID* ("VERSIONID").

Normally, the SCFs will not be concerned with the CM version number of the ESDTs. SDST will track this version. The implementation details depend on how ECS and the DAACs maintain the ESDTs.

Lookup Table

Each process may have one or more lookup tables that contain constants used in the science algorithm. A number of methods are currently used by the SCFs to distinguish between different versions of the files, some as part of the lookup table file name, and others based on some binary or text field contained in the lookup table. No special tools should be required to determine the version of a lookup table, whether the table is a binary or text file.

A set of lookup tables is stored in an ESDT bucket, a special type of ESDT for lookup tables. Each bucket could be versioned but the details of this are not currently known.

If the name of the lookup table contains some type of version information, the part of the file name that contains the version information should be clearly delineated and documented by the SCF for the SDST. The preferred method is to have the version information as a suffix on the lookup table name, with the first character being a "dot". For instance, "lookuptable.12" is the 12^{th} version of the lookup table. There should be some logical progression of the version numbers that allow the SDST and the processing centers to easily distinguish which table has the "higher" version number.

If multiple versions of the lookup table are used by a single execution of a process, the lookup table name must be unique for each version.

In the case where the version information is only contained within the lookup table, the version format and location within the file must be clearly documented by the SCF. This would typically be done as part of the software and/or lookup table delivery package. For binary files, the preferred method is for the version information to be contained in the file name. However, it might be necessary for the SCFs to provide a tool that will extract the version number from the file, but this is discouraged. If a version number must be located inside a binary file, then it should be extractable by running the UNIX "strings" command piped into "grep" command. Again, there should be some logical progression of the version numbers that allows the SDST and processing centers to easily distinguish a "higher" version of the lookup table.

It is recommended that there be some information in each product that clearly identifies which versions of lookup tables were used in the production of the data. In the MODAPS system, if the version of the table is in the file name, the simplest way to do this is to include the lookup table name in the *InputPointer* ("INPUTPOINTER") metadata field. However, in the ECS system, since the file name is a Universal Reference (UR) which may not contain the original lookup table file name, the information in the file name is lost and no information about the version of the lookup table is directly conveyed in the *InputPointer* metadata field. So, in general, the product should contain some other

metadata field that contains information clearly identifying versions of the input lookup tables. At this time a unified policy is not established and suggestions are welcome.

Product Version

Ideally, the product version number would be used to help the end user distinguish significant changes in the science content of the product. This would include changes to the inputs to the process, changes to the process itself, changes to the production rules, and changes to the lookup tables. However, this would not mean that all or even most changes would result in a change in the product version number, rather only those changes which would be scientifically meaningful to the end user.

The product version will be similar to the process and PGE version numbers and will consist of three parts: the major version number, the minor version number, and an update number. For instance, 2.1.3 is major version 2, minor version 1, and update 3.

The major version number will be incremented when significant changes are made to a science algorithm or lookup table either in preparation for, or following, a major project milestone, i.e., when the system version number changes. This allows the resynchronization of versions of individual products with the rest of the system. However, regardless of the project milestone, small or insignificant changes to the product may only result in changes to the update version numbers. Whenever the major version number is changed, the minor and update version numbers will be reset to zero.

The minor version will be used to keep track of changes in the science content of the product. The user of the product will be able to use this number to determine if a significant change in the science algorithm or lookup tables has taken place. Whenever the major version number is changed, the minor and update version numbers will be reset to zero

The update number will track any other changes. These may consist of minor algorithm or lookup table changes which handle specific cases and which do not effect a significant amount of the data being produced. Minor changes in the product format also fall into this category.

The decision of whether to increment the minor or update version number is left to the individual scientist/developer. However, a large number of increments in the minor version number, i.e., a large number of changes in the product science content, are discouraged because they confuse the end user and make analysis of multi-date data (time series) more difficult.

There is no practical way to fully specify or automate the process of changing product version numbers and still keep product version number meaningful. This is because of the complexity of the system, which includes loops, that feedback data from downstream processes to upstream processes. Therefore, it is each SCF's responsibility to be aware of the changes that may effect their products, including changes to upstream algorithms.

However, it is very difficult for a scientist to know beforehand, in all cases, whether a change to an upstream product will cause a scientifically meaningful change in their product.

Hopefully, by fully specifying the process, PGE, and lookup table version information elsewhere in each product, as well as the input product pointer information, it should be possible for an end-user to trace back and determine whether or not a change in the product is scientifically meaningful. Also, when used in conjunction with the production history string (below), the product version number should be very useful to the end user.

The product version number will be read out of the Process Control File (PCF) at a specific Logical Unit Number (LUN) for each product. The SDST will determine the specific LUNs used by each product. The process will read the version from the PCF and write it to each product produced as a inventory metadata field called *LocalVersionID* ("LOCALVERSIONID"). Specifically, the product version will be written as a string with no prefix or suffix and no leading zeros, for instance, "2.1.3". Each field may be multi-digit, for example, "3.12.105". Since PCFs are instantiated at runtime, the product version number will come from a PCF template that is read by the PGE script and put into the PCF.

The product version number will also appear in the inventory metadata field *LocalGranuleID* ("LOCALGRANULEID"). It will replace the current 3-digit 'vvv' version number. It will be written as the character 'v' followed by the product version number in the same form as above, except with the dots (".") will be replaced by underscores ("_"). The maximum number of digits in each part of the process version is two for the major and minor version numbers, and three for the update version number. The minimum number of characters in this field will be 6 (e.g., "v2_1_3") and the maximum 10 (e.g., "v11_13_125").

Production History String

The production history string attempts to provide the user with a way to determine if there is a significant change in the input products used to produce the current product. It is a character string that contains a concatenated list of the product versions of the current process, follow by all the inputs to the process and the inputs to any upstream processes. So, the concatenated string will automatically change when an SCF changes an upstream product version number.

The production history string does not capture all possible changes in the upstream or even the current process, including changes to ancillary data, changes to PGE production rules, etc. So, while the string does convey some important information about the current product and the input data used to produce the product, it cannot be solely relied upon for determining the production history of a product. The intent is that the history string

will be used in conjunction with the other versioning information both embedded in the product and external to the product.

The production history string will be stored in every product as an archive metadata field called *ProductionHistory* ("PRODUCTIONHISTORY"). It is a space-delimited string of ESDT names and product versions for the current product and each upstream MODIS product. The format is:

> <current product ESDT name>:<current product version>
> <upstream product 1 ESDT name>:<upstream product 1 highest version>
> <upstream product 2 ESDT name>:<upstream product 2 highest version>
> etc.

For example:

> MOD09:2.2.3 MOD35_L2:2.1.0 MOD02QKM:2.2.3 …

There will only be a single field for each unique input ESDT, and only the highest version numbers from the precursor products will be included. The current product will always be at the beginning of the list and the remaining products will not be in any specific order.

The list will be generated by summarizing the product versions of all of the input data sets read to produce the current product, excluding lookup tables and ancillary data. The summarization will be done by first concatenating the process history list from all the input data sets to generate a single list. Then the current product ESDT and version will be added to the beginning of the list and duplicates eliminated. Next, if there are two products in the list with the same ESDT name, all will be deleted except for the one with the highest version.

Implementation

Several aspects of this versioning framework call for all MODIS science software to read specific versioning information in a common way and write it into each product following identical formats. The required functionality could be implemented as a small library of functions. SCFs could then use these functions instead of developing their own implementations. This common approach would promote consistency throughout the MODIS science software system.

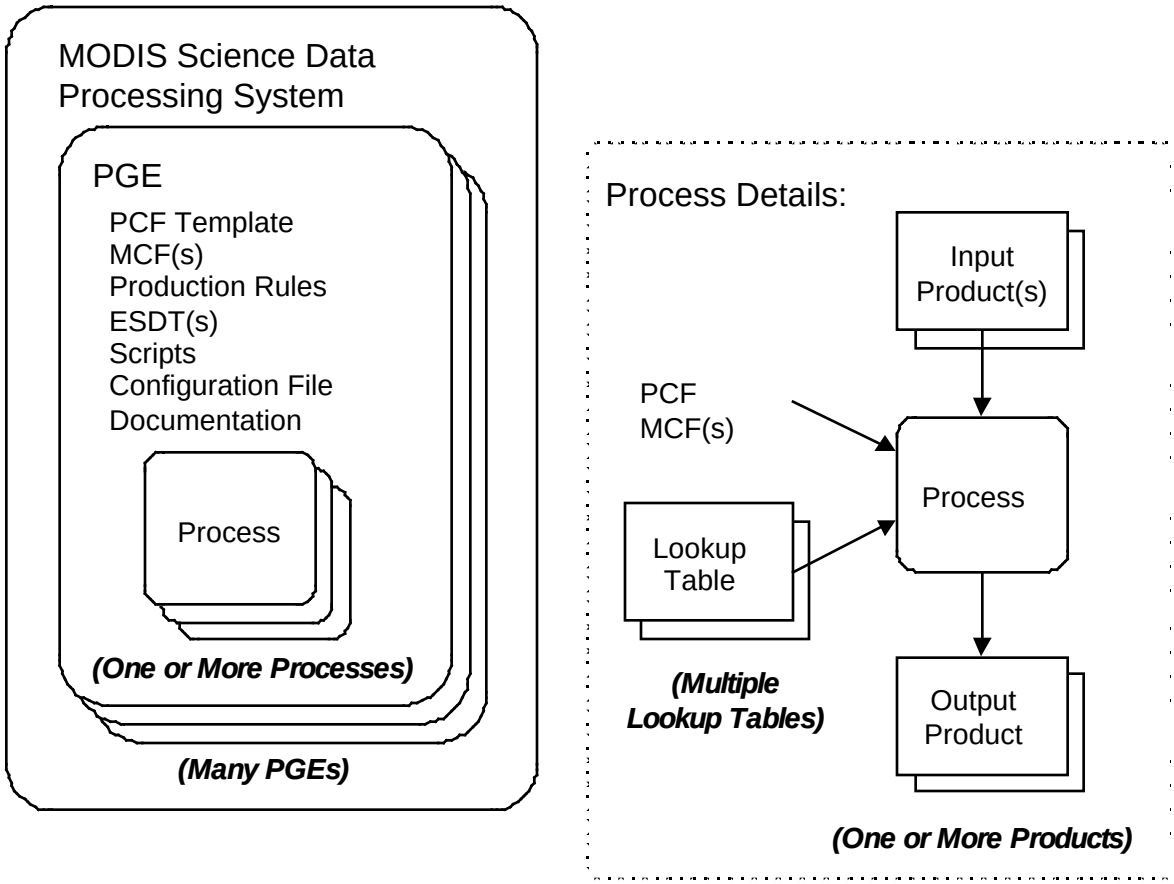

Figure 1. The relationships between elements of the MODIS Science Data Processing System where versioning is needed.

Area	**Format**	**Example**	**Location in Output Product**	**Where Stored**
System	*major.minor*	2.1	external to product; table of individual elements' versions	SDST CM
Process	*major.minor.update*	2.1.8	*ProcessVersion* - Inventory Metadata Product Specific Attribute (PSA)	Process Include File
PGE	*major.minor.update*	2.3.1	*PGEVersion* - Inventory Metadata Attribute	PCF Template
ESDT	integer from 0 to 255	2	*VersionID* - Inventory Metadata Attribute	MCF
Lookup Table	free form	2.3	none specified, possibly as part of file name in *InputPointer* Inventory Metadata Attribute	File Name or Internal to the Table
Product	*major.minor.update*	2.1.3	*LocalVersionID* - Inventory Metadata Attribute and as Part of Local Granule ID	PCF Template

Table 1. The format of the version numbers and the location of the versioning information within a MODIS product.