Standards Change Request

File Checksums **SCR3-1034.v3**
Elizabeth D. Rye **March 6, 2006**

Provenance:

Date: 2005-11-14, revision 2.0
Working Group: T. King (lead), M. McAuley
Title: MD5 Checksums (SCR3-1034.v2)

Date: 2004-11-22, revision 1.0
Working Group: J. Wilf (lead), T. King, M. McAuley
Title: MD5 Checksums for Files (SCR3-1034.v1)

Problem:

Provide a means for testing the integrity of PDS data.

As an entity responsible for maintaining data, it is critical that the PDS be able to ascertain the integrity of its archive. This includes verifying the integrity of data stored on various types of external physical media (all of which have finite life spans), detecting errors introduced during transfer of data to newer media, and detecting errors that occur during the transmission of data from data providers to the PDS, between PDS nodes, from the PDS to the NSSDC, and from the PDS to end users.

Proposed Solution:

A simple method for detecting these types of errors is to create and maintain a catalog of checksum values for every data product and file contained in PDS archives.

(Alternatively, checksums may be contained within data product labels, although this is only practical for storing the checksums of products with detached labels and provides no test for the integrity of the label files themselves. When used this way, the checksum keyword should be positioned within the object description of the object it is calculated for. Thus, in the case of a data product with a single detached label describing two data objects in two separate files, the checksum for each file would be stored within each of the explicit FILE objects.)

There are various types of checksums available that can be used for this purpose. The PDS has traditionally supported a simple checksum consisting of a 32-bit integer sum of all bytes in a data file. The one currently best suited to this purpose utilizes the 5th generation Manifest Digest (MD5) algorithm, briefly described in an attachment to this SCR. However, the proposals outlined in this SCR are not limited to any particular type of checksum and should allow for the easy implementation of new and better checksum algorithms as they become available.

The proposed changes outlined in this SCR are:

1. Update the element definition for the MD5_CHECKSUM keyword, correcting the errors in the original entry.
2. Establish a reserved file, "CHECKSUM.TAB", which contains checksum values for all files in an archive, to be optionally included in the INDEX directory of the archive.
3. Create a new keyword, CHECKSUM_TYPE, for use in the CHECKSUM.LBL file, to provide flexibility in permitting various types of checksums to be used.

Requested Changes:

Changes to the Standards Reference

The following changes to the PDS Standards Reference are required to support this SCR:

Add to section 10.2.2 Reserved File Names, "CHECKSUM.TAB".

Add to Chapter 19.3.2.3 INDEX Subdirectory, after INDEX.TAB:

CHECKSUM.TAB **Optional**

This file contains a checksum for every file on the volume. The format is to be standard PDS ASCII tabular format, with one column providing the file specification name for each file in the archive, and another column providing the checksum.

CHECKSUM.LBL **Optional**

This is the PDS label for the CHECKSUM.TAB file. The column object for the CHECKSUM column should contain the CHECKSUM_TYPE keyword, specifying the type of checksum recorded in the CHECKSUM.TAB file.

For an example of the CHECKSUM.TAB and CHECKSUM.LBL files, see Appendix D, section D.2. *(Is this the correct place to put these?)*

Each figure in Chapter 19. Volume Organization and Naming, will need to be updated to include a “CHECKSUM.TAB” and a "CHECKSUM.LBL" file in the INDEX directory.

Appendix D, section D.2 *(again, is this the correct place?)* add the sample CUMINDEX.TAB and CUMIDNEX.LBL files as shown in the attachment. (The following sections of Appendix D will need to be re-numbered.)

<u>Changes to the Data Dictionary</u>

Modify the description of the MD5_CHECKSUM keyword as shown in the attached element definition template.

Add the new keyword, CHECKSUM_TYPE, as shown in the attached element definition template.

<u>Changes to the PDS Tool Suite</u>

There are no immediate changes *necessary* in any PDS tool. However, as soon as practical, a simple PDS tool should be generated which incorporates existing published software and algorithms to create the CHECKSUM.TAB and CHECKSUM.LBL files for an archive (or archive volume). The same or an additional tool should be capable of then validating the archive against the checksum files and producing an error report.

Impact Assessment:

In addition to the above described changes,

1. It remains to be determined if PDS computers and software are capable of handling a 32 character hexadecimal number. If they are not, the approach of dealing with MD5 checksums as a hexadecimal number rather than as a character will need to be re-thought.
2. In the event that we utilize the appropriate hexadecimal number format for MD5 checksums, missions currently supplying a character value for the MD5_CHECKSUM keyword will need to receive a waiver.

Additional Information:

An MD5 checksum [MD5] is calculated for a bit stream of any length. An MD5 checksum is a 128-bit number, represented as a 32-character string of hexadecimal digits, e.g., 754b9db19f79dbc4992f7166eb0f37ce. The MD5 checksum specification states:

> It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given pre-specified target message digest.

In other words, no two files will have the same MD5 checksum unless they are identical. This has been demonstrated to be true in all but the most extreme circumstances.

While the MD5 algorithm is not recommended by the National Institute of Standards and Technology (NIST) for secure transmissions (SHA-1 is preferred) the MD5 checksum is faster to compute than the SHA-1 checksum and is well suited for integrity checking.

MD5 Support

The generation of MD5 checksums is widely supported. The algorithm with source code written in C is available at [MD5]. MD5 checksum generation is also supported in Java. The tool MD5deep [MD5-Deep] supports the recursive generation of MD5 checksums and the validation of files. The output of the MD5deep tool is one line per file consisting of the MD5 checksum, white space and the file name (with path). An example is:

```
$ md5deep -lr .
f8dd7758cb5231c9e7817c4710d00b6e ./aareadme.htm
d8b83365f5e117b9665181944889da3d ./aareadme.lbl
1e8d45f622e09b9e2998af1a6d67a296 ./aareadme.txt
7dcfa51691ddd149a5a091ebe87b9bb1 ./errata.txt
7f310bf58a37af7f9b16c4fe68a131fb ./voldesc.cat
```

[MD5] The document describing the MD5 algorithm can be found at: http://www.faqs.org/rfcs/rfc1321.html

[MD5-DEEP] MD5deep resources: http://md5deep.sourceforge.net/

```
PDS_VERSION_ID                    = PDS3
LABEL_REVISION_NOTE               = "2004-04-06, CN: BAM;
2004-10-14, PPI: S. Joy; 2006-03-06, EN: EDR"

OBJECT                            = ELEMENT_DEFINITION
  ELEMENT_NAME                    = "md5_checksum"
  BL_NAME                         = "md5checksum"
  DESCRIPTION                     = "
```

The MD5 algorithm takes as input a file (message) of arbitrary length and produces as output a 128-bit 'fingerprint' or 'message digest' of the input. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest. The MD5 algorithm is intended for digital signature applications.

The MD5 algorithm is designed to be quite fast on 32-bit machines. In addition, the MD5 algorithm does not require any large substitution tables; the algorithm can be coded quite compactly.

Most standard MD5 checksum calculators represent the checksum as a 32 character hexadecimal and it will be thus represented in PDS applications. Be aware that when feeding the checksum value to existing checksum calculators, it may be necessary to strip off the ODL hexadecimal mask from the value.

```
Example:  MD5_CHECKSUM = 16#0ff0a5dd0f3ea4e104b0eae98c87f36c#
```

The MD5 algorithm is an extension of the MD4 message-digest algorithm 1,2]. MD5 is slightly slower than MD4, but is more 'conservative' in design. MD5 was designed because it was felt that MD4 was perhaps being adopted for use more quickly than justified by the existing critical review; because MD4 was designed to be exceptionally fast, it is 'at the edge' in terms of risking successful cryptanalytic attack. MD5 backs off a bit, giving up a little in speed for a much greater likelihood of ultimate security. It incorporates some suggestions made by various reviewers, and contains additional optimizations. The MD5 algorithm has been placed in the public domain for review and possible adoption as a standard.

For OSI-based applications, MD5's object identifier is

```
md5 OBJECT IDENTIFIER ::=
  iso(1) member-body(2) US(840) rsadsi(113549) digestAlgorithm(2) 5}
```

In the X.509 type AlgorithmIdentifier [3], the parameters for MD5 should have type NULL.

The MD5 algorithm was described by its inventor, Ron Rivest of RSA Data Security, Inc., in an Internet Request For Comments document, RFC1321 (document available from the PDS).

References
==========

[1] Rivest, R., The MD4 Message Digest Algorithm, RFC 1320, MIT and
    RSA Data Security, Inc., April 1992.

```
   [2] Rivest, R., The MD4 message digest algorithm, in A.J.  Menezes
       and S.A. Vanstone, editors, Advances in Cryptology - CRYPTO '90
       Proceedings, pages 303-311, Springer-Verlag, 1991.

   [3] CCITT Recommendation X.509 (1988), The Directory -
       Authentication Framework."

  GENERAL_DATA_TYPE                = "NON_DECIMAL"
  MAXIMUM                          = "16#FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF#"
  MINIMUM                          = "16#00000000000000000000000000000000#"
  MAXIMUM_LENGTH                   = "36"
  MINIMUM_LENGTH                   = "36"
  STANDARD_VALUE_TYPE              = "RANGE"
  STANDARD_VALUE_SET_DESC          = "N/A"
  KEYWORD_DEFAULT_VALUE            = "N/A"
  UNIT_ID                          = "none"
  SOURCE_NAME                      = "PDS CN/R. SWORD"
  FORMATION_RULE_DESC              = "N/A"
  SYSTEM_CLASSIFICATION_ID         = "COMMON"
  GENERAL_CLASSIFICATION_TYPE      = "N/A"
  CHANGE_DATE                      = "2006-03-06"
  STATUS_TYPE                      = "APPROVED"
  STANDARD_VALUE_OUTPUT_FLAG       = "N"
  TEXT_FLAG                        = "N"
  TERSE_NAME                       = "md5checksum"
  SQL_FORMAT                       = "CHAR(36)"
  BL_SQL_FORMAT                    = "char(36)"
  DISPLAY_FORMAT                   = "JUSTLEFT"
  AVAILABLE_VALUE_TYPE             = "N/A"
END_OBJECT                         = ELEMENT_DEFINITION
END
```

```
PDS_VERSION_ID                    = PDS3
LABEL_REVISION_NOTE               = "2006-03-06, EN: EDR"

OBJECT                            = ELEMENT_DEFINITION
  ELEMENT_NAME                    = "checksum_type"
  BL_NAME                         = "checksumtype"
  DESCRIPTION                     = "

   The CHECKSUM_TYPE keyword is used to specify the type of checksum
   calculated for a file or data object.  There are currently only two
   checksums approved for use in the PDS: "SIMPLE" and "MD5".  For details on
   how each of these checksums is calculated, please see the element
   definitions for CHECKSUM and MD5_CHECKSUM, respectively."


  GENERAL_DATA_TYPE               = "IDENTIFIER"
  MAXIMUM                         = "N/A"
  MINIMUM                         = "N/A"
  MAXIMUM_LENGTH                  = "12"
  MINIMUM_LENGTH                  = "1"
  STANDARD_VALUE_TYPE             = "DYNAMIC"
  STANDARD_VALUE_SET              = {"SIMPLE", "MD5"}
  STANDARD_VALUE_SET_DESC         = "N/A"
  KEYWORD_DEFAULT_VALUE           = "N/A"
  UNIT_ID                         = "N/A"
  SOURCE_NAME                     = "PDS EN/E. Rye"
  FORMATION_RULE_DESC             = "N/A"
  SYSTEM_CLASSIFICATION_ID        = "COMMON"
  GENERAL_CLASSIFICATION_TYPE     = "N/A"
  CHANGE_DATE                     = "2006-03-06"
  STATUS_TYPE                     = "APPROVED"
  STANDARD_VALUE_OUTPUT_FLAG      = "Y"
  TEXT_FLAG                       = "N"
  TERSE_NAME                      = "checksumtype"
  SQL_FORMAT                      = "CHAR(12)"
  BL_SQL_FORMAT                   = "char(12)"
  DISPLAY_FORMAT                  = "JUSTLEFT"
  AVAILABLE_VALUE_TYPE            = "N/A"
END_OBJECT                        = ELEMENT_DEFINITION
END
```

D.2 CHECKSUM.TAB and CHECKSUM.LBL

Each PDS archive volume may optionally include a "CHECKSUM.TAB" file in the INDEX subdirectory. (If included, this file must be accompanied by a "CHECKSUM.LBL" file.) This file contains a checksum for every file contained on the archive volume (or in the entire archive, if stored as a virtual volume online).

D.2.1 Example of CHECKSUM.TAB

```
"AAREADME.TXT                        ",16#F8DD775BCB5231C9E7B17C4710D00B6E#
"ERRATA.TXT                          ",16#7DCFA51691DDD149A5A091EBEB7B9BB1#
"BROWSE/MARS/C1246XXX/I862934L.IMG   ",16#7F310BF58A37AF7F9B16C4FE68A131FB#
"BROWSE/MARS/C1246XXX/I862934R.IMG   ",16#1E8D45F622E09B9E299BAF1A6D67A296#
.
.
.
```

D.2.1 Example of CHECKSUM.TAB

```
PDS_VERSION_ID              = PDS3

RECORD_TYPE                 = FIXED_LENGTH
RECORD_BYTES                = 77
FILE_RECORDS                = 3623

DESCRIPTION                 = "CHECKSUM.TAB provides a checksum for all
                               files included on this archive volume, with
                               the exception of the checksum file itself."

^CHECKSUM_TABLE             = "CHECKSUM.TAB"

OBJECT                      = CHECKSUM_TABLE
  INTERCHANGE_FORMAT        = ASCII
  ROW_BYTES                 = 77
  ROWS                      = 3623
  COLUMNS                   = 2

  OBJECT                    = COLUMN
    NAME                    = FILE_SPECIFICATION_NAME
    DESCRIPTION             = "Identifies the file for which the checksum
                               was calculated."
    DATA_TYPE               = CHARACTER
    START_BYTE              = 2
    BYTES                   = 36
  END_OBJECT                = COLUMN

  OBJECT                    = COLUMN
    NAME                    = CHECKSUM
    DESCRIPTION             = "The checksum of the indicated file."
    CHECKSUM_TYPE           = MD5
    DATA_TYPE                    = ???
    START_BYTE                   = 40
    BYTES                        = 36
  END_OBJECT                     = COLUMN

END_OBJECT                       = CHECKSUM_TABLE
END
```