GRASS-SDTS User Guide (preliminary version)

David Stigberg

USACERL
2902 Newmark Drive
Champaign IL 61826
217-352-6511 x7631
217-373-7222 (fax)

ABSTRACT. This document is a guide to GRASS software for the transfer of spatial data in conformance with the Federal Spatial Data Transfer Standard and the SDTS Topological Vector Profile. It describes the installation and use of the software, and also provides a introduction to SDTS, the TVP, and to ISO8211--the mandatory physical implementation for SDTS. It gives particular attention to the use of imported SDTS data with GRASS in conjunction with a relational database management system. Appendices give comprehensive information about the mapping of spatial data between GRASS and the SDTS Topological Vector Profile.

1. Introduction

The purpose of this document is to help the GRASS user get the most out of the software that has been developed for the transfer of vector data to and from GRASS, in conformance with the SDTS Topological Vector Profile. It aims to make clear, on the one hand, what is involved in the creation of an SDTS and TVP-conformant dataset from GRASS vector data, and on the other, how the variety and richness of SDTS data that is becoming available can best be taken advantage of by the GRASS user.

1.1 GRASS Programs for SDTS

GRASS-SDTS software that has been completed thus far is for the transfer of *vector* spatial data, attributes, and associated metadata; software for the transfer of raster data between GRASS and SDTS is still under development. Following is a brief listing of the programs now available:

> *v.in.sdts:* imports SDTS data conforming to the TVP into GRASS, creating one or more GRASS vector maps, along with associated attribute files ready for installation in a relational database.
>
> *v.out.sdts:* exports a GRASS vector map layer, creating an SDTS dataset conforming to the TVP.
>
> *v.sdts.meta*: interactive menu-driven utility to create and install supplementary metadata and data quality reports for a vector map layer, preparatory to their incorporation in an SDTS export dataset.
>
> *v.sdts.meta.cp*: simple utility for installing supplementary metadata from a user-specified file, created by *v.sdts.meta* or other means.
>
> *v.sdts.dq.cp*: simple utility for installing SDTS data quality reports from user-specified files, created by *v.sdts.meta* or other means.
>
> *m.sdts.read:* program for reading SDTS or other data from files in FIPS 123 (ISO 8211) format.

1.2 The FIPS 123 Library and Utility Programs:

The required physical implementation for the SDTS-TVP is in accordance with that described in Federal Information Processing Standard (FIPS) 123: Specification for a Data Descriptive File for Information Interchange, also known as ISO 8211. In order to facilitate use of the FIPS 123 format, the U.S. Geological Survey has made available a FIPS 123 Function Library that was developed for them by Computer Sciences Corporation. The various GRASS/SDTS programs make use of this library to read and write SDTS files in FIPS 123 format.

Extensive documentation is included with the distribution of the FIPS 123 library. Also included are several sample application and utility programs that make use of the library. *m.sdts.dump* was derived from one of these programs. Finally, sample DLG3/SDTS and TIGER/SDTS datasets are included with the FIPS 123 Function Library. All of these materials are included in the GRASS-

SDTS installation package.

1.3 Installing and Compiling the GRASS-SDTS software

At the present time, you will need the *gcc* compiler to compile the FIPS 123 Library and the various GRASS-SDTS programs.

The software is currently available via *anonymous ftp* from the site at USACERL: moon.cecer.army.mil (129.229.20.254). There are three relevant files:

/grass/incoming/sdts.README
/grass/incoming/sdts_uguide.ps (this document, in Postscript format)
/grass/incoming/sdts.tar.Z

To install and compile the programs, follow these steps:

1. download all three files.
2. read sdts.README, for possible instructions that may supercede those given here.
3. uncompress sdts.tar.Z: zcat sdts.tar.Z
4. cd ...src.contrib/CERL in your GRASS source tree
5. tar -xf [path]/sdts.tar
6. cd SDTS
7. compile and install the FIPS 123 library and the various GRASS-SDTS programs by running gmake4.1 (or the equivalent) from the current directory (...src.contrib/CERL/SDTS).
8. when compilation is completed, cd ../../../src/CMD
9. run MAKELINKS (if more than one MAKELINKS exists in the src/CMD directory, run the one that is appropriate for your machine.

Once the compilation and installation process is completed, you may run *v.in.sdts*, *v.out.sdts*, etc. as you would any other GRASS program.

1.4 SDTS materials and the SDTS Task Force FTP Site

The SDTS Task Force at the U.S. Geological Survey is maintaining an ftp site for public domain SDTS-related materials. Copies of the Standard document itself (FIPS 173), including both the Topological Vector Profile and Raster Profile (Parts 4 and 5), may be found there. Also available from the ftp site are several articles about SDTS, source and executable versions of the FIPS 123 Library, sample SDTS vector datasets exported from DLG-3, DLG-E, TIGER, and GRASS, and sample digital orthophoto quad data compliant with the SDTS Raster Profile.

All the above materials may be downloaded via anonymous ftp from:

sdts.er.usgs.gov (130.11.52.170)

In addition, printed copies of the Standard and Profiles may be obtained from:

National Technical Information Service (NTIS)
Computer Products Office
5285 Port Royal Road

Springfield, VA 22161
(703) 487-4600

Finally, for more information about the SDTS, you may contact:

SDTS Task Force
U.S. Geological Survey
526 National Center
Reston, VA 22092
FAX (703) 648-5542
email: sdts@usgs.gov

2. SDTS and the Topological Vector Profile

The following section describes SDTS and the Topological Vector Profile. It gives enough details to enable the user to make good use of the GRASS/SDTS software, but it is by no means comprehensive. The reader is advised to obtain the Standard document itself, either from the USGS ftp site or from NTIS (see the preceding section).

The Topological Vector Profile can be seen, in most respects, as a limited subset of the Standard itself. The Standard contains a great deal of optionality, as to elements that may or may not be included in a transfer dataset and as to how various elements are encoded. The Topological vector Profile reduces this optionality significantly. It is designed for use with a specific kind of data, namely, for "geographic vector data with planar graph topology" It excludes "raster data, geometry-only vector data, planar graphs which do not have GT-polygons, and non-planar graph-based network data [which] may be accommodated by other profiles, or future extensions to this profile" (SDTS, Part 4, 1.1 and 1.1.2).

2.1 SDTS Modules

SDTS and the TVP organize data in terms of "modules", containing records and fields. Among other things, there are different modules for different kinds of spatial objects, for attributes, and for metadata.

The rules for module names are as follows:

1. All module names are standardized, and consist of four characters. All letters in module names must be upper case.

2. For most module names and in most circumstances, the 4-character names are prescribed, and shall be identical to the 4-character "primary field module field mnemonic (ISO 8211 Tag) specified for each module by SDTS: e.g., IDEN for the Identification module, CATD for the Catalog/Directory module, and so on.

3. Object module names begin with the standard 2-character "object representation code" for the objects shown in Table 1 below. The last two characters of the module name are then used to distinguish multiple modules of the same type.

4. Attribute Primary and Secondary modules may be multiple, and are named "Axxx" and "Bxxx" respectively (where x is any number 0-9 or any upper case letter).

5. Multiple modules of other types are indicated by changing the 4th character in the module's name. (In Table 1, a lower-case letter, as in "SCUr" indicates that it

can be changed to distinguish multiple modules.)

Following is a brief description of each of the modules that may, and in many cases must, be found in a TVP dataset. For more detailed information, see the appropriate passages in the SDTS document itself.

2.1.1 Global Information Modules (SDTS Part 1, 5.2)

IDEN (Identification): contains general identifying information about the contents of the transfer.

CATD (Catalog/Directory): contains listing of the modules and files that make up the transfer.

CATX (Catalog/Cross Reference): indicates pairwise relationships between modules

CATS (Catalog/Spatial Domain): identifies modules with specific domains, themes, map layers or aggregate objects (manifolds) within the transfer.

SCUR (Security): contains information on the security of the data in the transfer.

IREF (Internal Spatial Reference): transformation parameters for translating and scaling between internal and external spatial reference systems.

XREF (External Spatial Reference): contains specification of the external spatial reference system.

SPDM (Spatial Domain): specifies a geographic area domain within which the spatial data of other modules are contained.

DDDF (Data Dictionary/Definition): definitions of attributes and entities.

DDOM (Data Dictionary/Domain): specifies associated value domains of attributes.

DDSH (Data Dictionary/Schema): describes relationships among attributes and entities, and composition of, and key relationships among, attribute modules.

STAT (Transfer Statistics): contains table of module names, record counts, and spatial address counts.

2.1.2 Data Quality Modules (SDTS Part 1, 3, 5.3)

The five data quality reports contain narrative text reports, organized into one or more records each, optionally containing specific references to data records in other modules:

DQHL (Data Quality/Lineage): contains a “description of the source material from which the data were derived and the methods of derivation, including all transformations involved in producing the final digital files. . . .” (SDTS Part 1, 3.1).

DQPA (Data Quality/Positional Accuracy): includes an account of “the degree of compliance to the spatial registration standard”, etc. (SDTS Part 1, 3.2)

DQAA (Data Quality/Attribute Accuracy): includes accounts of tests and assessments of attribute accuracy (SDTS part 1, 3.3).

DQLC (Data Quality/Logical Consistency): describes “the fidelity of relationships encoded in the data structure of the digital spatial data”, including an account of “the tests performed and the results of such tests” (SDTS Part 1, 3.4)

DQCG (Data Quality/Completeness): includes “information about selection criteria, definitions used and other relevant mapping rules” (SDTS Part 1, 3.5).

2.1.3 Spatial Object Modules (SDTS Part 1, 2.3, 5.6)

NP (Point): a zero-dimensional object specifying location. TVP only allows usage in data quality reports.

NL (Label point): reference point for displaying map and chart text.

NE (Entity point): point for identification of location of point features (buildings, places, etc.)

NA (Area point): representative point within an area usually carrying attribute information about the area.

NO (Planar node): zero-dimensional object joining two or more chains, or an endpoint of a chain.

LE (Complete chain): a directed nonbranching sequence of line segments with left and right polygon references and start and end nodes.

PC[1] (GT-polygon): an area that is an atomic two-dimensional component of one and only one two-dimensional manifold, bounded by a set of complete chains.

PW (Universe polygon): the part of the universe outside the perimeter of the area covered by other GT-polygons.

PX (Void polygon): defines a part of a two-dimensional manifold bounded by other GT-polygons but otherwise with the same characteristics as the universe polygon.

FF (Composite object): constructed by aggregation, consisting of one or more other objects, either simple or composite.

LS, AC, AE, AU, AB, RU, PR, PU, and PX: object types only permitted as extensions (Annexes) to the TVP. These object types do not contain information not already found in other objects, and are not processed by the GRASS/SDTS software.

2.1.4 Attribute Modules (SDTS Part1, 4.1.3.6, 5.4)

A (Attribute Primary): Attribute Primary modules contain attribute values that are associated directly with spatial objects.

B (Attribute Secondary): Attribute Secondary modules contain attribute values associated with values of primary as well as other secondary attribute modules.

2.2 SDTS modules and GRASS

Table 1 lists the modules which may be found in a TVP dataset. The table indicates, for each module, the minimum and maximum number permitted in a transfer, whether a module of this type is created during the GRASS-to-SDTS export process, and whether or not the module, if found in an SDTS dataset, is processed by the current version of the GRASS/SDTS import software.

1. The object module code “PC” is used for module names for all three polygon types: PC, PW, and PX.

Table 1: SDTS -TVP Modules

Module Type	Name	Min-Max	GRASS export	GRASS import
Identification	IDEN	1	Y	Y
Catalog/Directory	CATD	1	Y	Y
Catalog/Cross Reference	CATX	0-1	Y	N
Catalog/Spatial Domain	CATS	1	Y	Y
Security	SCUr	0-n	N	N
Internal Spatial Reference	IREf	1-n	Y	Y
External Spatial Reference	XREF	1	Y	Y
Spatial Domain	SPDm	0-n	Y	Y
Data Dictionary/Definition	DDDf	0-n	Y	N
Data Dictionary/Domain	DDOm	1-n	Y	N
Data Dictionary/Schema	DDSh	1-n	Y	N
Transfer Statistics	STAT	1	Y	N
Lineage	DQHl	1-n	Y	N
Positional Accuracy	DQPa	1-n	Y	N
Attribute Accuracy	DQAa	1-n	Y	N
Logical Consistency	DQLc	1-n	Y	N
Completeness	DQCg	1-n	Y	N
Attribute Primary	Axxx	1-n	Y	Y
Attribute Secondary	Bxxx	0-n	N	Y
Point-Node	NOxx NPxx, NLxx NExx, NAxx	1-n 0-n 0-n	Y N Y	Y N Y
Line (Complete Chain)	LExx	1-n	Y	Y
Polygon	PCxx	1-n	Y	Y
Composite	FFxx	0-n	N	Y

2.3 Modules, Files and Filenames

Each file that makes up an SDTS-TVP dataset will contain one and only one module. In most cases, a module will only occupy a single file. A module will span files only when the size of a single file exceeds volume capacity due to media limitations, e.g., a module might be split into separate files on separate floppy discs.

Filenames consist of an 8-character basename followed by, in most cases, the extension ".DDF". The same first four characters of the basename, which can be any combination of numbers and letters, are used for all files. The next four characters will be the name of the module contained in the file. Names can be modified in two ways to handle modules that span files: by adding a 9th character to the basename of the file, or by modifying the last character of the extension (".DDF", "DDG", etc.).[2] All letters in filenames must be upper-case.

2.4 Module Contents and Interrelationships

Figure 1 illustrates typical SDTS module usage. It displays a single record in a "Complete Chain" Line module, which is required by the TVP for line segments. It also displays a corresponding record in an associated Attribute Primary module.

LE01: Line (Complete Chain) Module

LINE		Attributes	Left Polygon		Right Polygon		Start Node		End Node		Spat. Addrs
MODN	RCID	MODN RCID,...	MODN	MODN	MODN	RCID	MODN	RCID	MODN	RCID	SADR,...
LE01	1	A001	PC01	72	PC01	85	NO01	18	BO01	19	x1, y1; x2, y2...

A001: Bridge Construction Attr. Module

Attr. Primary		Attributes		
MODN	RCID	Serial_No	Constr_Date	Material_code
A001	1	132466	1946	33

Figure 1. Typical SDTS Module Relationships

The following points about Figure 1 should be noted:

1. Each field in the Line Module, with one exception, is made up of a pair of subfields, with Module Name (MODN) as the first subfield, and Record ID (RCID) as the second. The only field not of this type is the "Spatial Address" (SADR) field. The first MODN-RCID subfield-pair in the record identifies the Line Module record itself. The contents of this first MODN-RCID subfield pair form a unique key for the record, and *this is true for all records in all SDTS modules*. The other MODN-RCID subfield-pairs are *foreign identifiers,* which are in effect pointers to records in other modules. As is shown, one foreign identifier in the Line Module points to records in an Attribute Primary Module; others point to other modules (not shown), which contain records for the line segment's start and end nodes, and for polygons on the left and right of the particular line

2. Current GRASS-SDTS software cannot process multi-volume transfers.

segment.

2. Some module fields may repeat, i.e., they may have multiple entries for a single record. In the Line Module, this is true for the series of coordinate pairs in the Spatial Address field. It may also be, and often is, true for the Attributes field, which could contain multiple foreign identifiers for a single record, thus indicating one (at least)-to-many relationships between objects and attributes.

3. The Attribute Module in the figure contains two fields, an "Attribute Primary" field, which in turn consists of two subfields, and an "Attributes" field made up of three subfields. It is important to note that while the fields and subfields in the Line Module and other spatial object modules are defined by SDTS, (as is the "Attribute Primary" field in the Attribute Primary Module), the subfields in the "Attributes" field are defined by the user. In fact, the number and content of these attribute subfields, and the number of attribute modules themselves, are unlimited.

3. Creating an SDTS-TVP dataset from GRASS Vector Data: v.out.sdts

With *v.out.sdts*, an SDTS dataset is created from data associated with a GRASS vector map layer. This section presents a general discussion of the transfer process, highlighting some issues the exporter of GRASS data via SDTS should be aware of. More detailed treatment of the mapping of individual data elements between GRASS and SDTS-TVP modules is found in two Appendices. Appendix B gives a general view of the mapping between GRASS spatial objects and SDTS spatial objects. Appendix C is organized in terms of SDTS-TVP modules, listing the contents and sources for each field and subfield represented in a GRASS-to-SDTS transfer set.

For the actual mechanics of running *v.out.sdts,* see the GRASS *man* pages.

3.1 GRASS spatial data

Coordinate conversion: The TVP requires that coordinate data be transferred as 32-bit signed integers. This means that data must be converted on transfer between GRASS (and other GIS's) and SDTS. The GRASS-SDTS export software employs scaling factors, for geographic coordinates, of 1,000,000 for longitude and -1,000,000 for latitude, and 100 for UTM coordinates.

Apart from coordinate conversion, for the most part, vector objects in GRASS--lines, nodes, areas, etc.--can be transferred to and from SDTS without modification. There are, however, two exceptions to this statement that the user should note:

1. GRASS distinguishes between line segments that represent linear features, such as roads and streams, and line segments that represent area edges, such as soil polygons or field boundaries. Topology is different for the two types in GRASS: area edges carry information about the polygons on their left and right, but linear features do not. The TVP, on the other hand, makes no such distinction: *all* lines (Complete Chains) must carry left and right polygon markers.

v.out.sdts generates polygon markers for the lines that lack them. This is not problematic, except in one respect. If a map layer contains both area edges and linear features, the creation of new polygons can cause existing, attributed polygons to be altered (imagine, for example, a grid of linear features representing roads inside the bounds of a polygon representing a county), and hence their attribution to become uncertain. To work around this problem, an "area-only"option in *v.out.sdts* allows for the screening out of linear features on export.

2. Conversion is also necessary in the case of the export of certain kinds of point, or site, vector data from GRASS. GRASS allows such data, i.e., "Lines of type DOT", to be co-located with

nodes. The TVP does not. To handle this discrepancy, nodes co-located with such data are exported as attributed nodes in place of the "Line of type DOT". On import, the process is reversed (GRASS does not permit nodes to have attributes).

3.2 Metadata

GRASS in its present form is limited in the amount of metadata it stores in association with particular map layers, mapsets and databases. For the most part, SDTS requirements can be satisfied, at least in rudimentary fashion, by the combination of the metadata stored with GRASS vector maps and that generated or supplied by the export software. However, SDTS and the TVP do require five Data Quality Reports with every transfer, and these will have to be supplied by the user. In addition, much more metadata, if somehow available, would be desirable.

v.sdts.meta, *v.sdts.meta.cp*, and *v.sdts.dq.cp are* utility programs to aid in the preparation of supplemental metadata, and data quality reports intended to accompany a transfer dataset. *v.sdts.meta* is a menu-driven interface program for preparing and installing metadata and data quality files in the user's mapset in association with a particular GRASS vector mapset. *v.out.sdts* incorporates this installed metadata in the transfer dataset. *v.sdts.meta.cp* and *v.sdts.dq.cp* are simple utilities to install existing metadata and data quality files, created by *v.sdts.meta* or by some other means. See the man pages for these programs; also see Appendix C for details about metadata items.

3.3 Attributes

v.out.sdts exports two Attribute Primary modules with a GRASS vector map. Attribute Primary module AP00 consists of a single record with several fields, and contains global attributes, i.e., metadata items applicable to the entire transfer set (most are derived from the *dig* file header. Attribute Primary module AP01 contains the attribute data for the GRASS spatial objects themselves. There are two attribute fields: ATTR_NUM contains the integer value attributes from the map layer's *dig_att* file; ATTR_LABEL contains the corresponding labels or descriptions from the *dig_cats* file.

Default specifications for these attributes are supplied to the various Data Dictionary modules (Definition, Domain, and Schema) by *v.out.sdts*, unless superceded by user-supplied metadata via *v.sdts.meta.*

4. SDTS to GRASS: Spatial Objects and Metadata

Thus far, SDTS representative datasets from four different sources have been processed with the SDTS-to-GRASS transfer program, *v.in.sdts*: DLG-3/SDTS, DLG-E/SDTS, TIGER/SDTS, and, of course, GRASS/SDTS. It is worth noting that there are quite significant differences among these types, particularly but not exclusively with regard to the constellation of attributes and attribute-object relationships each contain. *v.in.sdts* can handle these different kinds of data, and, it is hoped, other kinds of data of similar complexity yet unseen. The TVP does allow the transfer of certain kinds of data and data organization that the current version of *v.in.sdts* does not accommodate; some of these are identified in the section on *Enhancements to GRASS/SDTS TVP Software* below.

4.1 Spatial objects

There is very little problematic as regards the import of *simple* spatial objects from SDTS TVP datasets into GRASS. There are two areas in which some special procession is involved, but neither of these is of real concern to the user. First, since the TVP does not require Area Points (for attaching attributes to polygons) and GRASS does, these points, in some cases, have to be newly created on import by *v.in.sdts*.Second, attributed nodes, permitted by the TVP but not allowed within GRASS, have to be converted to GRASS “lines of type DOT.”

The TVP allows for the presence of another kind of object which is a problem for GRASS. Composite objects, which are formed of one or more other objects, simple or composite, are sanctioned by the TVP but not known to GRASS. Composite objects, therefore, are treated as special kinds of attributes. Composite objects are discussed in the section on attributes below.

4.2 Metadata

GRASS is only able to make direct use of a small amount of the metadata that will be available in SDTS datasets. In fact, several SDTS modules dedicated to metadata are not processed by the GRASS-SDTS software. These include CATX (Catalog/Cross Reference), SCUR (Security), DDDF and DDOM (Data Dictionary/Definition and Data Dictionary/Domain), and the five DQ modules (Lineage, Positional Accuracy, Attribute Accuracy, and Completeness).

The data in these modules, which is potentially quite valuable, can be read directly by the user with *m.sdts.read.* This program, a modification of a utility supplied by the U.S. Geological Survey, prints the contents of SDTS modules in readable form.

5. SDTS to GRASS: Simple Attributes and “Basic” GRASS

Figure 1, referred to in Section 2.4 above, illustrates the kind of attribute data that can be accommodated relatively comfortably within the capabilities of “basic” GRASS, i.e., making use of no resources other than GRASS’s *dig_att* and *dig_cats* to store the imported attributes. Such usage is adequate if two conditions prevail: if, one, there is a one-to-one relationship between spatial objects and attribute records (which may consist of several different attribute fields); and, two, all object attributes in the SDTS dataset are in a single Primary Attribute module (ensuring that the attribute schema is uniform for all objects).

For data of this type, the contents of the *dig_att* and *dig_cats* files generated by *v.in.sdts* are as follows:

> *dig_att*: contains an entry for each attributed non-composite object (line segment, polygon, etc.). The category number in each entry is a unique integer value, functioning as a feature ID (fid).
>
> *dig_cats*: there will be a corresponding *dig_cats* record for each *dig_att* record. The *dig_cats* record will begin with the FID from the *dig_att* file, followed by the familiar “:” field separator. This is followed by an object code (obj_code) derived from the SDTS object module, an attribute code (attr_code) derived from the SDTS Primary Attribute module, and finally the attribute or attribute set itself. These latter fields are separated by “|”.

A *dig_cats* record derived from the data given in Figure 1 would look like this:

```
1:LE01#1|A001#1|132466|1946|33
```

Vector *dig_att* and *dig_cats* files are also generated by *v.in.sdts* when more complex TVP datasets are imported. The *dig_att* file is formed identically to that just described, but the *dig_cats* file is somewhat different, and in fact varies with different kinds of data. These matters are covered in the following section.

6. SDTS to GRASS: GRASS with a Relational Database

As is no doubt apparent, SDTS and the TVP are capable of handling data of great attribute complexity. As noted, SDTS distinguishes between two basic kinds of attributes: primary attributes that are related directly to spatial objects (simple or composite), and secondary attributes related to primary or other secondary attributes. Attribute modules are equivalent to relational tables, with relations among primary and secondary attribute tables through common fields. Spatial objects may be linked, via foreign identifiers, to multiple attributes in one or more different attribute tables. The schema of an SDTS TVP dataset--the number and kind of attribute fields and attribute tables, and the relationships among attributes and objects--is not predefined by SDTS or the TVP, but is determined by the producer of the dataset.

For most kinds of data likely to be available through SDTS, optimal access requires use of GRASS in conjunction with a relational database management system. In support of this, on request, *v.in.sdts* imports SDTS attribute tables in a form ready for installation and use with your RDBMS.

6.1 Setting up the GRASS-SDTS database: creat_db.sql and load_db.sql

On request, *v.in.sdts c*reates “relational database-ready” files. In addition, it generates two scripts, *creat_db.sql* and *load_db.sql*, along with the attribute files themselves. *Creat_db.sql* contains the SQL commands necessary to create a database and database tables conforming to the schema of the imported SDTS data. *Load_db.sql* contains commands to actually load the imported files into the newly-created database.

Before running *creat_db.sq*l, you must first specify a name for the new database. This is done by modifying the first line of both scripts. Thus, for example, if the new database were to be named “SDTSBASE1”, the files would be modified as follows:

creat_db.sql:	change	“create database <database_name>;”
	to	“create database SDTSBASE1;”
load_db.sql:	change	“database <database_name>;”
	to>	“database SDTSBASE1;”

These scripts have been tested with the Informix (tm) RDBMS, but should work with little or no modification in other SQL RDBMS’s. In Informix, the commands are “*isql database [path]/creat_db.sql*” and “*isql database [path]/load_db.sql*”, where *database* is the name of an already existing Informix database.

6.2 The GRASS-SDTS database: General design

Most typically, a relational database created as described above will contain relational tables of two kinds: attribute tables and object tables.

Attribute tables correspond almost exactly in content and order to the attribute primary and attribute secondary tables in the SDTS dataset. They differ in that, (1) the module name and record id fields in the original file are replaced by a new *attr_code* field (see below), and (2) in certain schemas, an additional *obj_code* field is added. All actual data fields, and their names, are carried over intact.

Object tables contain records for each simple object in the original SDTS dataset. *These tables function as links between the RDBMS attribute tables, and the object data now stored in GRASS dig and dig_att files*. The object tables are simple, consisting only of a feature ID (*FID*), an *obj_-code*, and in some schemas, an *attr_code*. There will be a single object table for each separate map layer in the data set. It is likely that in the most typical case, SDTS databases will consist of one object table linking a GRASS vector map layer with several attribute tables.

Some SDTS datasets will be simpler, and will make use of a single object/attribute table instead of the multiple tables just described. On the other hand, some datasets will be much more complex, and will have to make use of various "intersect" tables to achieve full functionality. These different possibilities will be illustrated in the section on "Import schemas" below.

dig_att and *dig_cats* files: *dig_att* file will be created under all possible scenarios, and will be identical to that described above (in section 5). *Dig_cats* files will be somewhat different, however. The *dig_cats* file will be identical in content to that of the object table, or, in the case of simple datasets, the object-attribute table. It should be noted that, in the case of complex attribute data, *v.in.sdt*s makes no attempt to select a single attribute or attribute set from the data and populate the dig_cats file accordingly. The user may, upon inspection of the output attribute tables and relevant metadata, may make such a selection and modify the *dig_cats* file accordingly.

6.3 Key fields

For the most part, the contents and order of the data, and the names of fields, in the SDTS attribute tables are imported intact. However, key fields joining object tables, both simple and composite, with attribute tables are newly generated by *v.in.sdts* on import.[3] These fields are standardized, and include the following:

> *fid*: feature ID, as already noted, is a unique integer identifier assigned to each attributed (simple) object. It is also used for the integer identifiers in the *dig_att* file.
>
> *obj_code:* object code, one for each fid. obj_code is generated on import by combining the SDTS object module name + '#' + record ID, e.g., "PC01#123" = polygon 123 from the PC01 module.
>
> *attr_code*: attribute code, one for each row in each attribute table. created similarly to *obj_code*, e.g., "AP09#452" = attribute record 452 from the AP09 attribute primary module.
>
> *comp_code*: composite object code, one for each composite object (see Section 6.4.5). constructed similarly to the other codes, e.g., "FF01#22" = composite object 22 from the "FF01" composite object module.
>
> *elem_code*: element code, i.e., code designating a constituent part of a composite object. Since composite objects can be made up of simple objects and/or other composite objects, *elem_codes* will match *obj_codes* in some cases and *comp_-codes* in others.

3. Key fields relating attribute primary and attribute secondary tables are defined in the SDTS dataset. They can be identified by consulting the SDTS files containing Catalog/Cross Reference (CATX) and Data Dictionary/Schema (DSH) modules.

6.4 Import schema

When *v.in.sdts* is run, the program analyzes the SDTS dataset being import, and on the basis of this analysis, selects the most appropriate from a set of generic attribute schemas to use for program output. Each of these schemas are explained below.

6.4.1 One-to-one object-attribute relationships; all object attributes in a single file

This is the simplest in structure of the SDTS datesets likely to be confronted by the GRASS user. In fact, a GRASS/SDTS dataset exported from GRASS to SDTS with *v.out.sdts* would have such a structure.

Here, only one "database-ready" file, a single object/attribute table, is created. It simply copies over the single attribute table's fields, and prepends three fields to these: two of these, *fid* and *obj_code*, derive from SDTS's object modules, while the third, *attr_code*, comes from the attribute table. The *fid* entries, which match those in the *dig_att* file, provide the link between GRASS and the relational database. It should be noted that for this simple date, *obj_code* and *attr_code* have no function other than providing documentation of the transfer dataset (Figure 2).

PC01: SDTS Polygon Module

MODN	RCID	Attribute Pointer
PC01	9	AP01 458
PC01	10	AP01 459

SDTS

AP01: SDTS Attribute Table

MODN	RCID	Field_1	Field_2	Field_3	...
AP01	458	aaa	bbb	ccc	...
AP01	459	xxx	yyy	zzz	...

GRASS/RDBMS

GRASS/RDBMS Object-Attribute Table

FID	Obj_code	Attr_code	Field_1	Field_2	Field_3	...
9	PC01#9	AP01#458	aaa	bbb	ccc	...
10	PC01#10	AP01#459	xxx	yyy	zzz	...

Figure 2. one-to-one relationships

6.4.2 One-to-one object-attribute relationships with multiple tables; many-to-one relation-

ships

A simple example of this type might have two attribute tables, one with line attributes and the other with polygon attributes. *v.in.sdts* would generate three database-ready files for this dataset, two for the attribute tables and a third for an object table connecting these attribute tables to the lines and polygons imported to GRASS. In the example in Figure 3, note that, (1), there is a foreign key- primary key relationship between *attr_code* in the object table and the *attr_code* entries in each of the two attribute tables; (2) the *obj_code* entries for lines (LE01#1, LE01#2, etc.) and for polygons (PC01#1, PC01#2, PC01#3, etc.) are, in this case, merely documentary in function; (3) as usual, the *fid* entries in the object table equal entries in the GRASS *dig_att* file.[4]

GRASS/RDBMS AP01 Attribute Table

Attr_code	Attr_1	Attr_2	...
AP01#12	aaaa	123	---
AP01#13	bbbb	456	---
AP01#14	cccc	789	---

GRASS/RDBMS Object Table

FID	Obj_code	Attr_code
53	PC01#1	AP01#12
54	PC01#2	AP01#13
55	PC01#3	AP01#14
56	PC01#4	AP02#47
57	PC01#5	AP02#48

GRASS/RDBMS AP02 Attribute Table

Attr_code	Attr_x	Attr_y
AP02#47	xx	88
AP02#48	yy	99

Figure 3. one-to-one relationships with multiple tables

6.4.3 One-to-many object-attribute relationships

DLG-3/SDTS datasets exhibit this schema, with objects having a "feature code" along with other attributes in one table, and with some of these objects having an additional, "coincident code" attribute in a second table. This kind of data is imported in a manner different from that illustrated in Figure 3, as follows: *attr_code* is omitted from the object table, and *obj_code* is added to the appropriate attribute tables. In this schema, reversing that just shown, there is a primary key-foreign key relationship between *obj_codes* in the object table and the attribute table, respectively (Figure 4).

4. TIGER/SDTS datasets we have encountered employ this kind of schema, although it is complicated greatly by the presence of composite objects.

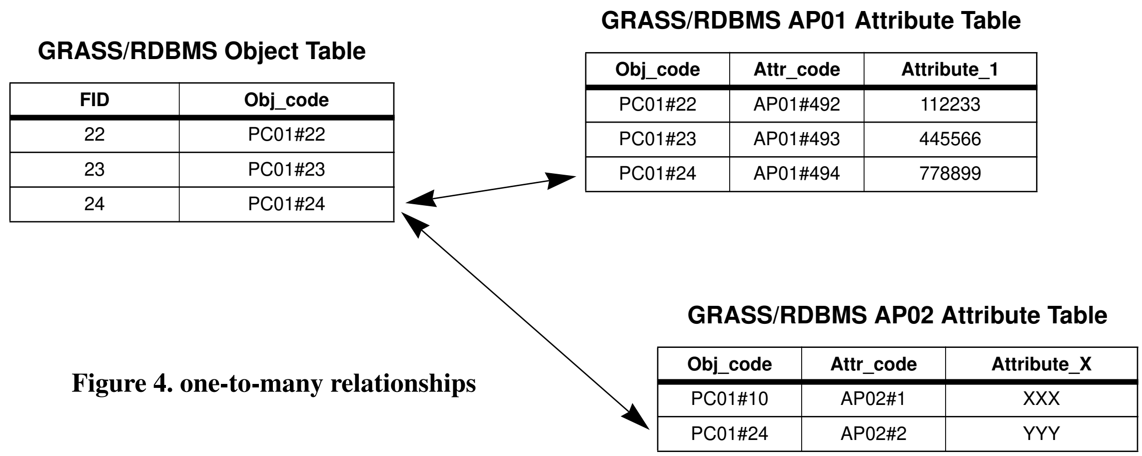

Figure 4. one-to-many relationships

6.4.4 Many-to-many object-attribute relationships

For this kind of data, an additional table, an *object/attribute intersect* table, which serves to link the object table with the various attribute tables, is needed. The table has two fields, for *obj_code* and for *attr_code*. In the object/attribute intersect table, *obj_code* has a foreign key relationship with the object table and *attr_code* has a foreign key relationship with one or another of the attribute tables. Figure 5 illustrates this solution to the many-to-many problem. Note that each Line in the example has attributes in two different tables (e.g., LE01#1 has attributes AP01#1 and AP12#1), and attribute AP12#1 applies to both LE01#1 and LE01#2.

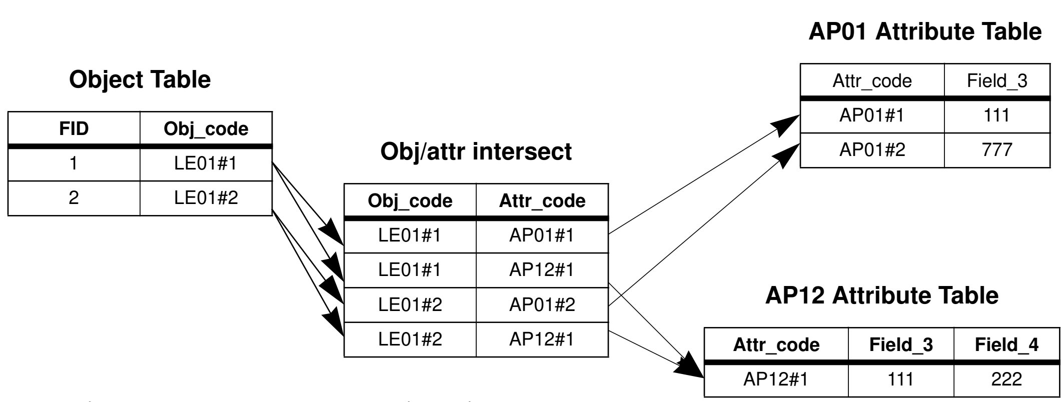

Figure 5. many-to-many relationships

6.4.5 composite objects

Composite objects in SDTS are made up of one or more other objects, primitive or composite. Composite objects can themselves have attributes. In fact, in SDTS composite object records consist of nothing more than pointers to their attributes and to their constituent objects. Composite object tables can be identified by their FF prefix (FF01, FF02, etc.).

Adding to the complexity of composite objects is the fact that their relationships with both their constituent parts and their attributes are very likely to be multiple. Many-to-many relations will be quite common between simple and composite objects, and also in at least some cases, between composite objects and attributes (TIGER/SDTS is such a case, with multiple composite object and attribute tables with many-to-many relationships.)

To handle this complexity, *v.in.sdts* generates two special intersect tables for datasets with composite objects: a composite/element (comp/elem) intersect table and a composite/attribute (comp/attr) intersect table.

The comp/elem table contains two fields: a *comp_code* field, combining composite object module name and record number; and an *elem_code* field, entries in which would be either simple *obj_-code* or *comp_code*.

The comp/attr intersect table contains *comp_codes* and *attr_codes* and functions like the object/attr intersect table. The comp/elem table is similar, but with an additional complication: it is inherently recursive, since an element may be part of a composite which in turn is part of another composite. Entries in the elem_code field may be composite codes that will also be entered in a different row in the comp_code column.

Figure 6. composite/element intersect table

Comp/elem intersect

Elem_code	Comp_code
FF07#1	FF02#1
PC01#1	FF07#1
FF07#3	FF02#2
FF07#2	FF02#2
PC01#434	FF07#3
PC01#353	FF07#3
PC01#184	FF07#2

As illustrated in Figure 7, traversing through a database with both attributed simple objects and attributed composite objects is somewhat complicated. In the example, the polygons listed in the object table represent city blocks, each of which has a “city-block-code” attribute, in attribute table AP22. Each city block, among other things, is identified with a particular elementary school district, and voting districts are modeled as composite objects with their attributes, “e_school_-code”, etc. in a separate attribute table (AP47). To discover the “city-block-code” for a particular polygon is straightforward, but finding a city-block’s elementary school district code means joining four different tables!

Object Table

FID	Obj_code
1	PC00#23
2	PC00#24
18	PC01#88

Comp/elem intersect

Elem_code	Comp_code
PC01#23	FF06#1
PC01#24	FF06#1
PC01#88	FF06#2

Comp/attr intersect

Comp_code	Attr_code
FF06#1	AP47#34
FF06#2	AP47#35

AP47 Attribute Table

Attr_code	e_school_code
AP47#34	472
AP47#35	593

AP22 Attribute Table

Attr_code	Obj_code	city_block_code
AP12#10	PC01#23	111
AP12#11	PC01#24	777
AP12#54	PC01#88	473

Figure 7. Composite and Simple Object-Attribute Relationships

6.4.6 DLG-E/SDTS: a special case?

While a typical use of composite objects is to group smaller objects (e.g., individual line segments bundled into composites for named roads) there are other uses, reflecting different ways of modelling spatial data itself. In the prototype data model for SDTS/DLG-E, primitive spatial objects (lines, polygons, nodes) do not carry attributes directly. Instead, “feature objects” both carry attributes and references to their corresponding spatial objects. And these feature objects are represented by SDTS composite objects. Thus, for this kind of data, all object-attribute relationships must be discovered by tracing through comp/elem and comp/attr intersect tables. These data would look similar to those shown in Figure 7, except there would be no attribute tables, such as that represented by AP22, related directly to the Object Table.)

6.5 SQL SELECT statements

There is no question that using an “RDBMS-enhanced” GRASS with the kinds of data described here, especially with complex schemas like that exhibited by TIGER/SDTS, presents something of a challenge. In the relatively near future, the task will be easier. The Generic Database Management Interface (GDMI) for GRASS, whose development is well underway, will greatly facilitate access to data such as that described here. Database utilities and application programs tailored to handle the kinds of relationships we have seen will also be of immense value

In the meantime, SDTS data imported with *v.in.sdts* can be used with available GRASS tools. Following are examples of two SQL SELECT statements of the kind that could be used with the GRASS/Informix utility, *d.what.v.inf*. This program processes the object selections the user makes from a displayed vector map, retrieving associated attributes from a database according to selection criteria specified by the user.

Usage of this program, for our purposes, is

d.what.v.inf -s sql=*filename* **map=***name*

where **map** is a vector map imported from an SDTS dataset and **filename** names a file containing an SQL SELECT statement. The first example given below contains a typical query, accessing all the attributes from a particular table (AP01) for the selected object (i.e. selected via mouse from the screen display):

```
SELECT ap01.*
    FROM ap01, object
    WHERE ap01.obj_code = object.obj_code
    AND object.fid = ?
```

Handling composite objects demands a more complicated SELECT statement. The following example illustrates this. Assuming the dataset illustrated in Figure 7 above, it identifies the elementary school district code (e_school_code) for any polygon, i.e., city block, selected:

```
SELECT ap47.e_school_code
    FROM ap47, comp_attr, comp_elem, object
    WHERE ap47.attr_code = comp_attr.attr_code
    AND comp_attr.comp_code = comp_elem.comp_code
    AND comp_elem.elem_code = object.obj_code
    AND object.fid = ?
```

The tediousness of this SELECT statement is perhaps mitigated somewhat by the understanding that it is generic in foundation, and needs to me changed only minimally for retrieval of composite object attributes in other attribute tables. For instance, merely substituting “ap34” for “ap47” (in three places), and substituting “voting_district” for “e_school_code” retrieves the voting districts for any city block selected.

7. Enhancements to GRASS/SDTS TVP Software

Several enhancements for the GRASS-SDTS software are being planned. Short-term development plans include the following:

State Plane transfer
Z-value coordinate import
multi-volume import
improved metadata preparation and export functionality
tools to support improved SDTS-imported RDBMS access

Further development will depend on the results of other major GRASS efforts, including modifications of the GRASS vector library and data structures now underway, and the completion of the Generic Database Management Interface for GRASS.

8. Sources of SDTS data

Sample datasets, for TIGER-SDTS and DLG-3-SDTS, accompany the installation package. After installation of the package available from moon.cecer.army.mil (see Section 1.3) sample data sets may be found under *...src.contrib/CERL/SDTS/mapdev/v.in.sdts/sample_data*.

As noted, a far larger sample of SDTS datasets, for GRASS, DLG-3, DLG-E, and TIGER data in

SDTS format, can be obtained via anonymous ftp, from the U.S. Geological Survey's SDTS Task Force site, *sdts.er.usgs.gov* (130.11.52.170)

Information about production SDTS datasets for DLG-3 and DLG-E data may be obtained from:

U.S. Geological Survey
Earth Sciences Information Center
507 National Center
Reston VA 22092
phone: 1-800-USAMAPS

For TIGER/SDTS data:

Customer Services
U.S. Bureau of the Census
Washington DC 20233-8300

general product information:
phone: 1-301-763-4100
fax: 1-301-763-4794

TIGER-specific information:
phone: 1-301-763-1384

APPENDICES

A. Import file names

The following conventions are observed in the naming of the various files that are created as a result of running *v.in.sdts*. Note that all "database-ready" files have a ".db" extension. This extension will not appear in the files and table names created after running creat_db.sql and load_db.-sql.

1. *vector map name*: if the SDTS dataset contains but a single map layer, or if only a single map layer from a multiple-map dataset is imported, the name specified as **output** is used as is. Otherwise, the name specified as **output** is extended with integers to specify the individual layers.

2. *database-ready attribute files*: will be named with the following format:

4-character SDTS prefix + 4-character module name + ".db" extension, e.g. "PRNMAP01.db", "HY01LE01.db", etc.

3. *database-ready object tables*:

if a single map, name = vector map name + ".db"; e.g., stuff.d

if multiple maps, each name = first letter of layer name + number of layer + 4-character SDTS prefix +.db; e.g., s_1HY01.db.

4. *intersect tables*: these tables each have the form: SDTS-prefix + intersect type code+"db", for example:

object/attribute intersect = PRNMobat.db
composite/element intersect = PRNMffel.db
composite/attribute intersect = PRNMffat.db

5. *object/attribute table*: for simple imports that require only this table, the specified vector map

name is used, with a ".db" extension.

6. *special case*: if either the SDTS prefix, or the output map layer name chosen by the user lacks a leading alpha character (e.g., as in TIGER/SDTS's 2500 prefix), the letter "T" is arbitrarily prepended to the db-ready filenames in the creat_db and load_db scripts to create the actual database table names. This is done since Informix and other SQL RDBMS's require that both table names and databases begin with a letter. Thus, e.g., the database tablename for "2500AP57.db" would be "T2500AP57".

B. GRASS-to-SDTS mapping: Spatial Objects

The following table illustrates the mapping GRASS spatial objects to SDTS spatial objects allowed under the Topological Vector Profile. Note that GRASS "Lines of type DOT" become "Entity Points" in SDTS; a similar case is found with DLG-3, where objects of the "Degenerate Line" type become Entity Points.]The Table also illustrates the fact that GRASS's distinction between AREA and LINE types are not preserved directly in the SDTS object modules (the distinction could be recorded in an attribute field).

Finally, it should be noted that the reverse mapping, from SDTS to GRASS is identical to that shown here, except that attributed nodes in SDTS map to Lines of Type DOT in the current version of GRASS. Planar Nodes without attributes are not imported into GRASS, since they are generated within GRASS from the end-points of line segments.

GRASS Spatial Objects Mapped to SDTS Spatial Objects

GRASS Spatial Objects	SDTS Spatial Objects
Node	Planar Node "NO"
Line of type DOT	Entity Point "NE"
Line of type LINE	Complete Chain "LE"
Line of type AREA	Complete Chain "NE"
Area	GT Polygon "PC"
Area attr. coords (in dig_att file)	Area Point "NA"

C. GRASS to SDTS mapping: detailed listing of SDTS modules

Following is a listing, in tabular form, of each of modules included in a SDTS dataset produced by *v.out.sdts.* Each table lists all of the fields and subfields that would be present, their contents, and the source of their contents.[5]

Terms used in the "Source" column are defined as follows:

Profile	content defined by SDTS and the Topological Vector Profile
generated	defined within v.out.sdts, or the product of simple generation or counting procedure by v.out.sdts
GRASS	extracted from GRASS data or metadata

5. The organization and many of the details of this section are based on the document, "DLG-3 to SDTS Vector Profile Mapping (DRAFT May 5, 1993)", produced by the SDTS Task Force, U.S. Geological Survey.

Profile/GRASS	selection from Profile-defined choices based on GRASS data
user-supplied	external to GRASS, supplied by user through v.sdts.meta or other external means

Identification Module

Field	Subfield	Contents	Source
Identification	Module Name	“IDEN”	Profile
	Record ID	“1”	generated
	Standard ID	“Spatial Data Transfer Standard”	Profile
	Profile ID	“SDTS Topological Vector Profile”	Profile
	Profile Version	“Version 1.0 June 10, 1994”	Profile
	Profile Document Reference	“FIPS 173-1 Part 4”	Profile
	Title	default = MAP NAME from dig file header	GRASS, unless user-supplied (v.sdts.meta)
	Data Structure	“GRASS vector [version number from dig file header] format--transfer includes entity points and area points with pointers to surrounding GT-polygons”	Profile and GRASS
	Map Date	Original source date, YYYYMMDD or YYYY format	user-supplied (v.sdts.meta)
	Data Set Creation Date	system date function call	generated
	Scale	dig file header MAP SCALE	GRASS
	Comment	default = “see AP00 module for other ID info”	generated unless user-supplied (v.sdts.meta)
Conformance	Composites	“N”	generated
	Vector Geometry Only	“N”	Profile
	Vector Topology	“Y”	Profile
	Raster	“N”	Profile
	External Spatial Reference	“1”	Profile
	Feature Level	“4” (means “non-SDTS”)	generated
Attribute ID	Module Name	global attribute module name “AP00”	generated
	Record ID	“1”	generated

Catalog/Directory Module

Field	Subfield	Contents	Sources
Catalog/Directory	Module Name	"CATD"	Profile
	Record ID	determined by order in this module	generated
	Name	name of module referenced	Profile
	Type	primary field name from module referenced	Profile
	File	file name = user-supplied prefix + referenced module name + ".DDF"	user-supplied (v.out.sdts parameter)

Catalog/Cross-Reference Module

Field	Subfield	Contents	Sources
Catalog/Cross-Reference	Module Name	"CATX"	Profile
	Record ID	determined by order in this module	generated
	Name 1	name of module referenced	Profile
	Type 1	primary field name from module referenced by Name 1	Profile
	Name 2	name of module cross-referenced	Profile
	Type 2	primary field name from module referenced by Name 2	Profile

Catalog/Spatial Domain Module

Field	Subfield	Contents	Sources
Catalog/Spatial Domain	Module Name	"CATS"	Profile
	Record ID	determined by order in this module	generated
	Name	name of module referenced	Profile
	Type	primary field name from module referenced by Name	Profile
	Map	title from "MYNAME" file in PERMANENT mapset	generated
	Theme	user-supplied vector map layer name	user-supplied (v.out.sdts parameter)
	Aggregate Object	NULL	
	Aggregate Object Type	"GT"	Profile

Internal Spatial Reference Module

Field	Subfield	Contents	Sources
Internal Spatial Reference	Module Name	"IREF"	Profile
	Record ID	"1"	generated
	Spatial Address Type	"2-Tuple"	Profile/GRASS
	Spatial Address X Component Label	"Longitude" or "Easting"	Profile/GRASS
	Spatial Address Y Component Label	"Latitude" or "Northing"	Profile/GRASS
	Horizontal Component Format	"BI32"	Profile
	Scale factor X	"-0.000001" (lat-lon) or "0.01" (UTM)	generated
	Scale Factor Y	"0.000001" (lat-lon) or "0.01" (UTM)	generated
	X Origin	"0.0"	generated
	Y Origin	"0.0"	generated

External Spatial Reference Module

Field	Subfield	Contents	Sources
External Spatial Reference	Module Name	"XREF"	Profile
	Record ID	"1"	generated
	Reference System Name	"GEO" or "SPCS" or "UTM" or "UPS"	Profile/GRASS
	Horizontal Datum	default = NULL, unless user-supplied	user-supplied (v.sdts.meta)
	Zone Number	if UTM or State Plane, value returned from GRASS G_zone() function	GRASS

Spatial Domain Module

Field	Subfield	Contents	Sources
Spatial Domain	Module Name	"SPDM"	Profile
	Record ID	"1"	generated
	Spatial Domain Type	"MINMAX"	Profile
	Domain Spatial Address Type	"INTERNAL"	Profile
	Domain Spatial Address	W, E, S, N Edge values from GRASS dig file header	GRASS

Data Dictionary Modules

For the most part, the entries generated by *v.out.sdts* for the three SDTS Data Dictionary modules are fixed. The exceptions are as follows:

Data Dictionary/Definition module: contents of the Definition field for the records for GRASS_ENT, ATTR_NUM, and ATTR_LABEL may be user supplied, via *v.sdts.meta* or other means. Otherwise, default definitions are supplied by *v.out.s-dts.*

Data Dictionary/Domain module: minimum and maximum Domain Values for the records for ATTR_NUM are derived from the GRASS data.

Data Dictionary/Schema module: (1) Maximum Subfield Length values for the records for OTHER_INFO, ATTR_NUM, and ATTR_LABEL are derived from the GRASS data. (2) The Entity Label field for the ATTR_NUM and ATTR_LA-BEL records may be user-supplied via *v.sdts.meta*; otherwise the vector map layer name specified as a parameter to *v.out.sdts* is used for Entity Label.

Data Dictionary/Definition Module

Field	Subfield	Contents	Sources
Data Dictio-nary/Definition	Module Name	“DDDF”	Profile
	Record ID	defined by order in this module	generated
	Entity or Attribute	“ENT” or “ATT”	Profile
	Entity/Attribute Label	name of entity or field name of attribute	generated
	Definition	free text definition of entity or attribute	generated or user-supplied
	Attribute Authority	“USACERL”	generated
	Attribute Author-ity Description	“U. S. Army Construction Engineering Labora-tories”	generated

Data Dictionary/Domain Module

Field	Subfield	Contents	Sources
Data Dictio-nary/Domain	Module Name	“DDOM”	Profile
	Record ID	defined by order in this module	generated
	Attribute Label	field name of attribute	generated
	Attribute Authority		generated
	Attribute Domain Type	“ALPHANUM”, “INTEGER”, “REAL”, etc.	Profile/generated
	Attribute Domain Value Format	“A”, “I”, “R”, etc.	Profile/generated
	Attribute Domain Measurement Unit	dependent upon attribute label	generated
	Range or Value	“MIN”, “MAX”, or “VALUE”	Profile/generated
	Domain Value	valid values for referenced attribute	generated or user-supplied (v.sdts.-meta)
	Domain Value Def-inition	text describing domain value	generated

Data Dictionary/Schema Module

Field	Subfield	Contents	Sources
Data Dictionary/Schema	Module Name	“DDSH”	Profile
	Record ID	defined by order in this module	generated
	Name	name of module referenced (“AP00” or “AP01”)	generated
	Type	primary field name of module referenced (“ATPR”)	generated
	Entity Label	name of entity to which attribute(s) apply	generated or user-supplied (v.sdts.-meta)
	Entity Authority	“USACERL”	generated
	Attribute Label	((sub)field name of attribute	generated
	Attribute Authority	“USACERL”	generated
	Format	format of the attribute (sub)field referenced	generated
	Unit	measurement unit for the attribute (sub)field referenced	generated
	Maximum Subfield Length	maximum length of contents of (sub)field referenced	generated or user-supplied (v.sdts.-meta)
	Key	“NOKEY”, “PKEY”, “FKEY”, “PFKEY”	generated

Transfer Statistics Module

Field	Subfield	Contents	Sources
Transfer Statistics	Module Name	“STAT”	Profile
	Record ID	defined by order in this module	generated
	Module Type Referred	primary field name of referenced module	Profile
	Module Name Referred	name of referenced module	Profile
	Module Record Count	number of records in the referenced module	generated
	Spatial Address Count	number of spatial addresses in the referenced module	generated

Data Quality Modules (5)

Field	Subfield	Contents	Sources
Lineage or Positional Accuracy or Attribute Accuracy or Logical Consistency or Completeness	Module Name	"DQHL" or "DQPA" or "DQAA" or "DQLC" or "DQCG"	Profile
	Record ID	defined by order in this module	generated
	Comment	narrative about aspect of data quality covered by this module	user-supplied (v.sdts.meta)

Planar Node Module

Field	Subfield	Contents	Sources
Primary Field	Module Name	"N001"	Profile
	Record ID	Node internal ID number (GRASS dig_plus file)	GRASS
	Object Representation Code	"NO"	Profile
Spatial Address Field	X and Y Components	X and Y coordinates for node (GRASS dig_plus file)	GRASS
Attribute ID	Module Name	"AP01"	generated
	Record ID	Attr. internal ID number for "Line of type DOT" collocated with this node, if any (GRASS dig_plus file)	GRASS

Entity Point Module

Field	Subfield	Contents	Sources
Primary Field	Module Name	"NE01"	Profile
	Record ID	"Line of type DOT" internal ID number (GRASS dig_plus file	GRASS
	Object Representation Code	"NE"	Profile
Spatial Address Field	X and Y Components	X and Y coordinates for "DOT" (GRASS dig_plus file)	GRASS
Attribute ID	Module Name	"AP01"	generated
	Record ID	Attr. internal ID number for "DOT" (GRASS dig_plus file)	GRASS
Area ID	Module Name	"PC01"	generated
	Record ID	internal ID number for polygon on left or right of "DOT" (GRASS dig_plus file)	GRASS

Area Point Module

Field	Subfield	Contents	Sources
Primary Field	Module Name	“NA01”	Profile
	Record ID	internal ID number for associated area (GRASS dig_plus file)	GRASS
	Object Representation Code	“NA”	Profile
Spatial Address Field	X and Y Components	X and Y coordinates for area point (= dig_att coordinates) (GRASS dig_plus file)	GRASS
Area ID	Module Name	“PC01”	generated
	Record ID	Area internal ID number (GRASS dig_plus file)	GRASS

Line Module

Field	Subfield	Contents	Sources
Primary Field	Module Name	“LE01	Profile
	Record ID	internal ID number for line (GRASS dig_plus file)	GRASS
	Object Representation Code	“LE”	Profile
Attribute ID	Module Name	“AP01”	generated
	Record ID	internal ID number for line attr (GRASS dig_plus file)	GRASS
Polygon ID Left	Module Name	“PC01”	generated
	Record ID	internal ID number for left polygon (GRASS dig_plus file)	GRASS
Polygon ID Right	Module Name	“PC01”	generated
	Record ID	internal ID number for right polygon (GRASS dig_plus file)	GRASS
Start Node ID	Module Name	“NO01”	generated
	Record ID	internal ID number for start node (GRASS dig_plus file)	GRASS
End Node	Module Name	“NO01”	generated
	Record ID	internal ID number for end node (GRASS dig_plus file)	GRASS
Spatial Address Field	X and Y Components	X and Y coordinates for line (GRASS dig file)	GRASS

Polygon Module

Field	Subfield	Contents	Sources
Primary Field	Module Name	"PC01"	Profile
	Record ID	internal ID number for area (GRASS dig_plus file)	GRASS
	Object Representation Code	"PW" for universe polygon, "PC" for all others"	Profile
Attribute ID	Module Name	"AP01"	generated
	Record ID	Internal ID number for area attribute (GRASS dig_plus file)	GRASS

Attribute Module AP00

Field	Subfield	Contents	Sources
Attribute Primary Field	Module Name	"AP00"	Profile
	Record ID	"1"	GRASS
Primary Attributes Field	ORGANIZATION	ORGANIZATION from dig file header	GRASS
	DIGIT_DATE	DIGIT DATE from dig file header	GRASS
	DIGIT_NAME	DIGIT NAME from dig file header	GRASS
	MAP_NAME	MAP NAME from dig file header	GRASS
	MAP_DATE	MAP DATE from dig file header	GRASS
	OTHER_INFO	OTHER INFO from dig file header	GRASS
	ZONE	ZONE from dig file header	GRASS
	WEST_EDGE	WEST EDGE from dig file header	GRASS
	EAST_EDGE	EAST EDGE from dig file header	GRASS
	SOUTH_EDGE	SOUTH EDGE from dig file header	GRASS
	NORTH_EDGE	NORTTH EDGE from dig file header	GRASS
	DIGIT_THRESH	Digitizing Threshold from dig file header	GRASS
	MAP_THRESH	MAP THRESH	GRASS
	VERSION	Version Major + Version Minor from dig file header	GRASS
	BACK_VERSION	Back_Major + Back_Minor from dig file header	GRASS
	ELLIPSOID	Ellipsoid, if available, from "PROJ_INFO" file in PERMANENT mapset	GRASS

,

Attribute Module AP01

Field	Subfield	Contents	Sources
Attribute Primary Field	Module Name	"AP01"	Profile
	Record ID	Internal ID for attributes from GRASS dig_plus file	GRASS
Primary Attributes Field	ATTR_NUM	value of dig_att referenced by internal ID number	GRASS
	ATTR_LABEL	corresponding dig_cats value	GRASS