

**DECODES**  
**Device Conversion and Delivery System**  
**Version 7.3**  
**User's Guide**

by



**ilex engineering, inc**

Ilex Engineering, Inc  
9250 Bendix Road, North  
Columbia, MD 21044  
Tel: 410.715.1117  
Email: [info@ilexeng.com](mailto:info@ilexeng.com)

Revision 3  
April, 2007



# Table of Contents

<b>1.</b>	<b>INTRODUCTION TO DECODES.....</b>	<b>1</b>
1.1	WHAT'S NEW? .....	3
1.1.1	<i>New Features for DECODES 7.1</i> .....	3
1.1.2	<i>New Features for DECODES 6.5</i> .....	3
1.1.3	<i>New Features for DECODES 6.4</i> .....	3
1.1.4	<i>New Features for DECODES 6.3</i> .....	3
1.1.5	<i>New Features for DECODES 6.2</i> .....	3
1.1.6	<i>New Features for DECODES 6.1</i> .....	3
1.1.7	<i>New Features for DECODES 6.0</i> .....	4
<b>2.</b>	<b>DECODES THEORY OF OPERATIONS .....</b>	<b>6</b>
2.1	DECODES SOFTWARE COMPONENTS .....	7
2.2	DECODES DATABASE ARCHITECTURE.....	8
2.2.1	<i>Setup Information</i> .....	8
2.2.1.1	Enumerations .....	8
2.2.1.2	Time Zones .....	10
2.2.1.3	Sensor Data Types .....	10
2.2.1.4	Engineering Units .....	11
2.2.2	<i>Properties</i> .....	11
2.2.3	<i>Decoding, Formatting, and Converting Specifications</i> .....	12
2.2.3.1	Equipment Models .....	12
2.2.4	<i>Sites &amp; Site Names</i> .....	12
2.2.5	<i>Platform Information</i> .....	13
2.2.6	<i>Presentation and Unit Conversion</i> .....	15
2.2.7	<i>Routing Specification Information</i> .....	17
<b>3.</b>	<b>CONFIGURING DECODES .....</b>	<b>18</b>
3.1	THE "DECODES.PROPERTIES" FILE.....	18
3.1.1	<i>Surrogate Keys Generated by SQL Database</i> .....	20
3.1.2	<i>Date Formats in SQL Database</i> .....	20
3.2	SQL DATABASE FORMAT CHANGES .....	21
<b>4.</b>	<b>MAINTAINING THE DECODES DATABASE.....</b>	<b>22</b>
4.1	INITIALIZING YOUR EDITABLE DECODES DATABASE.....	22
4.1.1	<i>Standard XML Setup Files</i> .....	22
4.1.2	<i>Importing Data from EMIT or Pre-Release-5 DECODES</i> .....	23
4.1.3	<i>Importing XML Data from Other DECODES Sites</i> .....	24
4.2	INTERACTIVELY EDITING THE DATABASE .....	24
4.3	UPDATING THE INSTALLED DATABASE .....	25
4.4	EXPORTING PLATFORMS TO XML FILES .....	26
4.5	EXPORTING THE ENTIRE DATABASE .....	27
4.6	OTHER DATABASE UTILITIES .....	27
4.6.1	<i>Creating the Platform Cross-Reference File</i> .....	27
4.6.2	<i>Creating LRGS-Style Network List Files</i> .....	28
<b>5.</b>	<b>THE DECODES DATABASE EDITOR .....</b>	<b>29</b>
5.1	GUI ORGANIZATION .....	30
5.1.1	<i>List Panels in General</i> .....	30
5.1.2	<i>Edit Panels in General</i> .....	30
5.1.3	<i>Exiting the Editor</i> .....	30
5.2	THE PLATFORM EDIT PANEL .....	32
5.2.1	<i>Platform Sensors</i> .....	32
5.2.2	<i>Transport Media</i> .....	32
5.3	THE SITE EDIT PANEL .....	34

5.4	THE PLATFORM-CONFIG EDIT PANEL .....	36
5.4.1	<i>The Decoding Script Edit Dialog</i> .....	38
5.5	THE EQUIPMENT-MODEL EDIT PANEL .....	41
5.6	THE PRESENTATION GROUP EDIT PANEL .....	42
5.6.1	<i>Using a Presentation Group as a Sensor Filter</i> .....	43
5.7	THE DATA SOURCE EDIT PANEL .....	44
5.8	THE NETWORK LIST EDIT PANEL .....	46
5.9	THE ROUTING SPECIFICATION EDIT PANEL .....	47
5.10	IMPORT XML FILES .....	48
5.11	EXPORT RECORDS TO AN XML FILE .....	49
5.12	INSTALL RECORDS TO YOUR PRODUCTION DATABASE .....	50
5.12.1	<i>Automatic Installation when You Change Records</i> .....	50
5.12.2	<i>Manual Installation to the Production Database</i> .....	51
<b>6.</b>	<b>THE DECODES FORMAT LANGUAGE .....</b>	<b>52</b>
6.1	EXECUTION OF FORMAT STATEMENTS BY A ROUTING SPEC .....	52
6.2	STEPPING THROUGH THE SCRIPT AND THE DATA .....	52
6.3	FORMAT OPERATION OVERVIEW .....	53
6.4	SKIPPING AND POSITIONING OPERATIONS .....	55
6.5	THE CHECK OPERATION.....	55
6.6	THE SCAN OPERATION .....	56
6.7	THE JUMP OPERATION.....	56
6.7.1	<i>The Repeat Operation</i> .....	56
6.8	FIELD OPERATIONS .....	57
6.8.1	<i>Field Length and Delimiters:</i> .....	58
6.8.2	<i>Date Fields</i> .....	59
6.8.3	<i>Time Fields</i> .....	60
6.8.4	<i>Time Interval Fields</i> .....	61
6.8.5	<i>Minute Interval and Offset Fields</i> .....	61
6.8.6	<i>Format Label Fields</i> .....	62
6.8.7	<i>Sensor Value Fields</i> .....	63
6.9	SPECIAL DECODING FEATURES .....	63
6.9.1	<i>How to Omit Specific Sensor Values</i> .....	63
6.9.2	<i>Data Delimited by either a Plus or Minus Sign</i> .....	63
<b>7.</b>	<b>DECODES ROUTING SPECIFICATIONS .....</b>	<b>64</b>
7.1	HOW TO RUN A ROUTING SPECIFICATION .....	65
7.1.1	<i>Routing Spec Properties</i> .....	66
7.1.2	<i>Adding Network List Names from the Command Line</i> .....	66
7.1.3	<i>Overriding Time Range from the Command Line</i> .....	66
7.1.4	<i>Status Output File</i> .....	67
7.1.5	<i>Optional Lock File</i> .....	67
7.2	TIME TAGGING DATA SAMPLES .....	67
7.3	EXPANDING ENVIRONMENT VARIABLES .....	69
<b>8.</b>	<b>DATA SOURCES .....</b>	<b>70</b>
8.1	LRGS DATA SOURCE.....	70
8.1.1	<i>Timeouts in LRGS Data Sources</i> .....	72
8.2	FILE DATA SOURCE.....	73
8.2.1	<i>Delimiting Messages Within the File</i> .....	73
8.3	DIRECTORY DATA SOURCE .....	74
8.4	HOT BACKUP GROUP DATA SOURCE .....	76
8.5	ROUND ROBIN GROUP DATA SOURCE.....	77
8.6	SOCKET STREAM DATA SOURCE .....	78
8.6.1	<i>Using SocketStreamDataSource for NOAAPORT</i> .....	80

<b>9.</b>	<b>OUTPUT FORMATTERS.....</b>	<b>81</b>
9.1	SHEF OUTPUT FORMAT.....	82
9.2	SHEFIT OUTPUT FORMAT.....	84
9.3	HUMAN READABLE OUTPUT FORMAT.....	85
9.4	EMIT-ASCII FORMAT.....	86
9.5	EMIT-ORACLE FORMAT.....	87
9.6	DUMP FORMATTER.....	88
9.7	USGS STDFMT OUTPUT FORMATTER.....	89
9.8	TRANSMIT MONITOR FORMATTER.....	90
<b>10.</b>	<b>CONSUMERS.....</b>	<b>92</b>
10.1	PIPE CONSUMER.....	92
10.2	FILE CONSUMER.....	93
10.3	DIRECTORY CONSUMER.....	94
<b>11.</b>	<b>SPECIAL CONSIDERATIONS FOR EDL FILES.....</b>	<b>95</b>
11.1	HOW DOES DECODES FIND THE PLATFORM RECORD?.....	95
11.2	TIME ZONES FOR DATES & TIMES IN EDL FILES.....	95
<b>12.</b>	<b>SPECIFIC SCENARIOS.....</b>	<b>97</b>
12.1	HOW TO CREATE A NEW PLATFORM SPECIFICATION.....	97
	<i>Create a Site for the New Platform.....</i>	<i>97</i>
	<i>Create an Equipment Model Record for the New Platform.....</i>	<i>97</i>
	<i>Create a Configuration for the New Platform.....</i>	<i>98</i>
	Rules and Conventions for Configuration Naming.....	99
	Enter the Sensor and Formatting Information.....	100
	<i>Create the Platform Record.....</i>	<i>101</i>
	<i>Add the new Platform to a Network List.....</i>	<i>102</i>
	<i>Testing the new Platform in a Routing Spec.....</i>	<i>102</i>
<b>13.</b>	<b>REFERENCE LIST EDITOR.....</b>	<b>103</b>
13.1	ENUMERATIONS.....	104
13.2	ENGINEERING UNITS.....	105
13.3	ENGINEERING UNIT CONVERSIONS.....	106
13.4	DATA TYPE EQUIVALENCIES.....	107
<b>14.</b>	<b>RATING COMPUTATION USING USGS RDB FILES.....</b>	<b>108</b>
14.1	STORE THE RDB FILES IN A KNOWN DIRECTORY.....	109
14.2	ASSOCIATE RDB FILES WITH PLATFORM SENSORS.....	109
14.3	CONFIGURE THE COMPUTATION PROCESSOR.....	110
14.4	SIMPLE ASCII TABLE FILES.....	111
14.5	RUN YOUR ROUTING SPEC WITH COMPUTATIONS ENABLED.....	111
<b>15.</b>	<b>THE DECODES PLATFORM WIZARD.....</b>	<b>112</b>
15.1	PLATFORM WIZARD START PANEL.....	113
15.2	PLATFORM WIZARD SITE PANEL.....	114
15.3	PLATFORM WIZARD SENSORS PANEL.....	115
15.4	PLATFORM WIZARD EQUIPMENT MODEL PANEL.....	116
15.5	PLATFORM WIZARD DECODING SCRIPT PANEL.....	116
15.6	PLATFORM SPECIFIC INFORMATION.....	118
15.7	SAVE YOUR WORK.....	119
<b>16.</b>	<b>THE INTERACTIVE DECODING WIZARD.....</b>	<b>120</b>
16.1	THE FILE SCANNING PANEL.....	121
16.2	THE DECODING AND TIME SHIFT PANEL.....	122

16.3	THE SAVE-RESULTS PANEL .....	123
<b>17.</b>	<b>USGS NATIONAL WATER INFORMATION SYSTEM (NWIS) INTEGRATION .....</b>	<b>124</b>
17.1	CONFIGURE DECODES FOR YOUR NWIS DATABASE .....	124
17.2	NWIS MAPPING FOR SITES AND SITE NAMES .....	125
17.3	ADDITIONAL NWIS SENSOR PARAMETERS .....	127
	<b>APPENDIX A: ENGINEERING UNIT LIST.....</b>	<b>141</b>

## Table of Figures

FIGURE 1-1: WHAT DECODES DOES.....	1
FIGURE 2-1: DECODES DETAILED DATA FLOW.....	6
FIGURE 2-2: DECODES COMPONENTS.....	7
FIGURE 2-3: ENUMERATIONS ERD.....	8
FIGURE 2-4: DATA TYPES ERD.....	10
FIGURE 2-5: ENGINEERING UNITS & CONVERTERS ERD.....	11
FIGURE 2-6: EQUIPMENT MODEL ERD.....	12
FIGURE 2-7: SITES AND SITE NAMES ERD.....	12
FIGURE 2-8: PLATFORM INFORMATION ERD.....	13
FIGURE 2-9: PRESENTATION AND UNIT CONVERSION ERD.....	15
FIGURE 2-10: ROUTING SPECIFICATION ERD.....	17
FIGURE 5-1: DATABASE EDITOR PLATFORM LIST SCREEN.....	29
FIGURE 5-2: PLATFORM CONFIG EDIT PANEL.....	31
FIGURE 5-3: PLATFORM EDIT PANEL.....	33
FIGURE 5-4: TRANSPORT MEDIUM EDIT DIALOG.....	34
FIGURE 5-5: THE SITE EDIT PANEL.....	35
FIGURE 5-6: PLATFORM CONFIG EDIT PANEL.....	36
FIGURE 5-7: EDIT CONFIG SENSOR DIALOG.....	37
FIGURE 5-8: DECODING SCRIPT EDIT DIALOG SHOWING INTERACTIVE DECODING.....	39
FIGURE 5-9: LOAD SAMPLE MESSAGE DIALOG.....	40
FIGURE 5-10: EQUIPMENT MODEL EDIT DIALOG.....	41
FIGURE 5-11: PRESENTATION GROUP EDIT PANEL.....	43
FIGURE 5-12: DATA SOURCE EDIT PANEL SHOWING LRGS DATA SOURCE.....	44
FIGURE 5-13: DATA SOURCE EDIT PANEL SHOWING HOT BACKUP GROUP.....	45
FIGURE 5-14: NETWORK LIST EDIT PANEL.....	46
FIGURE 5-15: ROUTING SPECIFICATION EDIT PANEL.....	47
FIGURE 5-16: IMPORT XML FILES DIALOG.....	48
FIGURE 5-17: IMPORT DIALOG SHOWING SUCCESSFUL SCAN.....	48
FIGURE 5-18: THE EXPORT DIALOG.....	49
FIGURE 5-19: THE AUTO-INSTALL DIALOG.....	50
FIGURE 5-20: MANUAL INSTALL DIALOG.....	51
FIGURE 7-1: DATA FLOW FOR ROUTING SPECIFICATIONS.....	64
FIGURE 9-1: EXAMPLE OF SHEF .A OUTPUT.....	83
FIGURE 9-2: EXAMPLE OF SHEF .E OUTPUT.....	83
FIGURE 9-3: EXAMPLE OF SHEFIT OUTPUT FORMAT.....	84
FIGURE 9-4: EXAMPLE OF HUMAN READABLE OUTPUT FORMAT.....	85
FIGURE 9-5: EXAMPLE OF EMIT-ASCII FORMAT.....	86
FIGURE 9-6: EXAMPLE OF EMIT-ORACLE OUTPUT FORMAT.....	87
FIGURE 9-7: EXAMPLE OF DUMP OUTPUT FORMAT.....	88
FIGURE 9-8: EXAMPLE OF USGS STDFMT OUTPUT.....	89
FIGURE 9-9: EXAMPLE OF TRANSMIT MONITOR FORMAT.....	90
FIGURE 13-1: REFERENCE LIST EDITOR ENUMERATIONS TAB.....	103
FIGURE 13-2: REFERENCE LIST EDITOR ENGINEERING UNITS TAB.....	105
FIGURE 13-3: REFERENCE LIST EDITOR EU CONVERSIONS TAB.....	106
FIGURE 13-4: REFERENCE LIST EDITOR - DATA TYPE EQUIVALENCIES TAB.....	107
FIGURE 14-1: USGS RDB RATING FILE EXAMPLE.....	108
FIGURE 14-2: SELECT PLATFORM SENSOR AND PRESS "SENSOR PROPERTIES".....	109
FIGURE 14-3: PLATFORM SENSOR PROPERTIES DIALOG.....	110
FIGURE 14-4: EXAMPLE "COMPUTATIONS.CONF" FILE.....	110
FIGURE 14-5: EXAMPLE OF SIMPLE ASCII TABLE FILE.....	111
FIGURE 15-1: PLATFORM WIZARD START PANEL.....	113
FIGURE 15-2: PLATFORM WIZARD SITE PANEL.....	114
FIGURE 15-3: PLATFORM WIZARD SENSORS PANEL.....	115
FIGURE 15-4: PLATFORM WIZARD EQUIPMENT MODEL PANEL.....	116

FIGURE 15-5: PLATFORM WIZARD DECODING SCRIPT PANEL.....	117
FIGURE 15-6: PLATFORM WIZARD "PLATFORM SPECIFIC INFO" PANEL.....	118
FIGURE 15-7: PLATFORM WIZARD "SAVE YOUR WORK" PANEL.....	119
FIGURE 16-1: DECODING WIZARD - FILE SCANNING PANEL.....	121
FIGURE 16-2: DECODING WIZARD - DECODING AND TIME SHIFT PANEL.....	122
FIGURE 16-3: DECODING WIZARD - SAVE RESULTS PANEL.....	123
FIGURE 17-1: SITE NAME EDIT DIALOG.....	125
FIGURE 17-2: EDIT CONFIG SENSOR DIALOG.....	127
FIGURE 17-3: EDIT PLATFORM SENSOR DIALOG.....	128

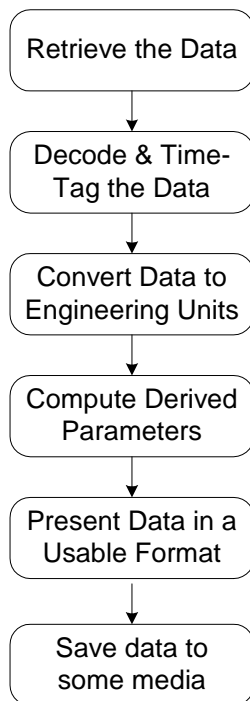


# 1. Introduction to DECODES

DECODES stands for DEvice CONversion and DELivery System. DECODES is a suite of software that takes data from a variety of recording devices and converts it into standard engineering units, suitable for entry into a database.

The types of recording devices include both Electronic Data Loggers (EDLs), which are electronic recorders whose data for the most part are manually retrieved, and Data Collection Platforms (DCPs), whose data are retrieved by satellite telemetry.

The operations performed by DECODES are depicted in Figure 1-1.



**Figure 1-1: What DECODES Does.**

The fourth step, “Compute Derived Parameters” is not implemented in the current release.

Currently DECODES can handle data from any GOES DCP received either from a GOES receiver or over DOMSAT. It has the basic capability to parse data from EDL files, but this capability has not been extensively tested in the current release.

Data can be retrieved from saved files or over the network from an LRGS.

DECODES can handle any ASCII format currently in use by the DCS. This includes true ASCII values or the pseudo-binary values common in compact random messages.

DECODES uses a database of platform specifications to tell it how to decode data from a given source. DECODES manages a fairly complex database that includes entities for:

<b>DataSource</b>	Where to retrieve raw data from: Directory, File, LRGS Network Connection, shared memory, etc. If an LRGS network connection is specified, you can specify network lists, time-ranges, etc.
<b>Platforms &amp; Sites</b>	Site-specific parameters such as transmission times, GOES channel numbers, DCP address, etc.
<b>DecodingScript</b>	A structured scripting language that tells DECODES how to extract time-tagged samples from the raw messages
<b>EU-Converters</b>	How to convert raw values into engineering units, and how to convert between various types of engineering units (e.g. feet to meters).
<b>PresentationInfo</b>	How to format each type of sample. For example, you might want all stage values to be presented in centimeters with 10.3 resolution.
<b>DataTypes</b>	DECODES knows how to convert between the USGS (EPA) numeric codes and SHEF physical element codes.
<b>Formatters</b>	How to format data on output. Formatters are implemented for SHEF, SHEFIT, Human-Readable, and “Dump”.
<b>Consumers</b>	Where to put the output data: files, directories, pipes, etc.
<b>RoutingSpec</b>	Puts all of the above together. A RoutingSpec says where to get the raw data, how to decode it, how to EU convert it, how to format it, and where to send it.

DECODES is written in 100% pure Java. Therefore there should be (almost) no porting issues in running it on any modern computing platform. Ilex Engineering has tested it under Windows 2000 and Linux.

DECODES merges the capabilities of the former DECODES software used by U.S. Geological Survey (USGS) with the EMIT (Environmental Message Interpreter Translator) software used by several U.S. Army Corps of Engineers (USACE) districts.

DECODES is open-source software. It was developed by Ilex Engineering, Inc., under a contract jointly funded by USGS and USACE. To obtain a copy of DECODES software, contact the U.S.G.S. Water Resources Division Headquarters.

For more information on Ilex Engineering, Inc., visit our web site at [www.illexeng.com](http://www.illexeng.com), call us at 410.465.6948, or email us at [info@illexeng.com](mailto:info@illexeng.com).

## **1.1 What's New?**

### **1.1.1 New Features for DECODES 7.1**

- New settings in the “decodes.properties” file: TransportMediumTypePreference and DataTypeStdPreference. See Table 3-1 for details.
- Import, Export, and Install functions have been integrated into the graphical database editor. See sections 5.10 through 5.12.
- There is a new “decoding wizard” application. See Chapter 16.
- Several changes have been made to the DECODES SQL Database Schema. See the DECODES Database Schema document for details.
- DECODES has been integrated with the USGS NWIS Database. See Chapter 17.

### **1.1.2 New Features for DECODES 6.5**

- The Installation section of the manual has been split into a separate document.
- Enhancements to the SHEF formatter allow you to output a full 7-character SHEF code. The SHEF formatter also will now always output the maximum number of decimal points as specified by the presentation group. This is a work-around for parsers which require all samples to have decimal points.
- New “decodes.properties” values to support timezones for compatibility with SQL database systems from Ingres, Oracle, and Postgres. The new values are called “sqlDateFormat” and “sqlTimeZone”. See Table 3-1 for details.
- DirectoryDataSource property called “OneMessageFile”. If set, it assumes that each file contains a separate raw message. See section 8.3 for details.

### **1.1.3 New Features for DECODES 6.4**

- The Routing Spec (rs) and Database Editor (dbedit) commands can now take a -E option to specify the Explicit location of an XML database. This allows you to run from a database other than your normal editable or installed one.
- Field Operations can now specify more than one delimiter character. See section 6.8.1.

### **1.1.4 New Features for DECODES 6.3**

- A new database editor has been added called the “Platform Wizard”. This is especially valuable for novice users as it guides you through the complex process of created or editing platform records. See Chapter 15.

### **1.1.5 New Features for DECODES 6.2**

- Support for RDB file rating conversions. See Chapter 14.

### **1.1.6 New Features for DECODES 6.1**

- Where is my routing.log file? Routing specs now write individual log files with the name of the spec and a “.log” extension. These log files are placed in a directory

specified by the 'RoutingStatusDir' value in the decodes.properties file. If undefined, this defaults to \$DECODES\_INSTALL\_DIR/routstat.

- New Load Message dialog in the database editor makes it easy to retrieve a sample message to test your scripts.
- New Reference List Editor Application – see chapter 13.
- Support for DCP Monitor Web Application – See separate DECODES Web Applications Manual.
- For EDL Files, you can now specify a time-zone in the transport medium. See section 11.2 and Figure 5-4 for details.
- The -R switch has been added to the 'rs' routing-spec command to remove redundant DCP data from the output. Do not use this switch for EDL files.
- You can now do platform specific offsets and scaling for each sensor. See section 5.2.1 for details.

### **1.1.7 New Features for DECODES 6.0**

- New dialog-based installation procedure replaces the manual unpacking of zip files and environment variable setting.
- The database has been enhanced to support many new features. Some of these features are implemented in 6.0. Some will be implemented in the near future.
- SQL 'Sequences' are now segregated from other portions of the database. This will facilitate support for Ingres and Oracle.
- Support for EDL (Electronic Data Logger) Files. USGS provided examples of the wide variety of format and time-tagging conventions.
- Complete support for all date & time operators.
- Data order can now be set explicitly within the decoding script. This replaces the non-intuitive 'TimeOrder' property values.
- The Site records now contain elevation and a free-form description field.
- Properties can now be associated with Platform Records.
- Transport Medium records contain additional GOES parameters like Preamble and Time Adjustment.
- Configuration Sensors may now have multiple data types associated with them.
- Presentation Group Rounding Rules have been redone to be more useful.
- The 'rs' (Routing Spec) command now has options for setting network lists and time ranges. Previously these needed to be set in the database records. This will result in fewer, more flexible, routing specs.
- The 'Directory Data Source' is now implemented. A routing-spec can now watch a specified directory (or group of directories) for new files to appear. We expect this to be used heavily for EDL files.
- A new format operator 'w' can be used to skip any amount of white space (spaces, tabs, carriage returns, linefeeds).
- A new field type, 'MINT', can be used for parsing time intervals in the message given as a number of minutes. Field type 'MINT-' used for specifying negative interval. For details, see section 6.8.5.

- A new field type, ‘MOFF’, can be used for parsing a minute-offset from the message. Many GOES DCPs contain a minute offset to find the time of the first sample. You can now use these values. For details, see section 6.8.5.
- The LRGS Message Browser now has a ‘Display All’ button.
- SHEF Formatter option for complete 7-char SHEF codes.
- Numerous GUI improvements and bug-fixes.
- New “sutron standard” example files in the to\_import directory. This uses a flexible format that can handle most sutron ascii message formats.

## 2. DECODES Theory of Operations

Figure 2-1 provides a more detailed data flow diagram of what happens when data is decoded. In each box in the figure, different algorithms and parameters are applied according to information in your database.

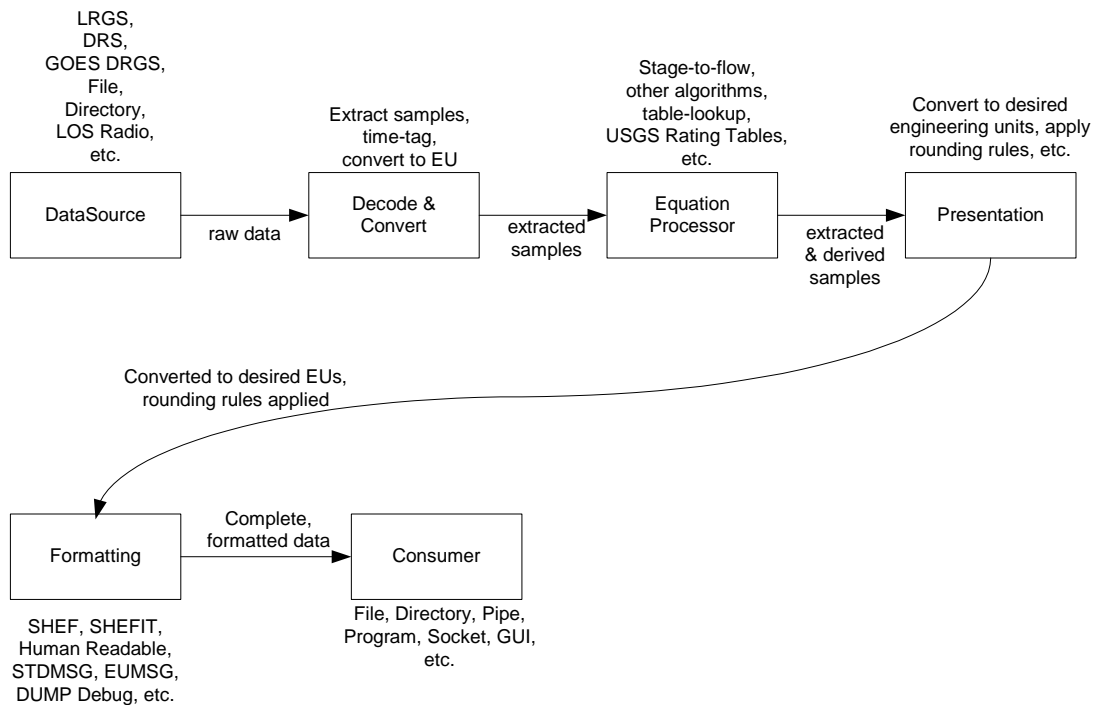


Figure 2-1: DECODES Detailed Data Flow

## 2.1 DECODES Software Components

Figure 2-2 shows the DECODES software components and how they relate.

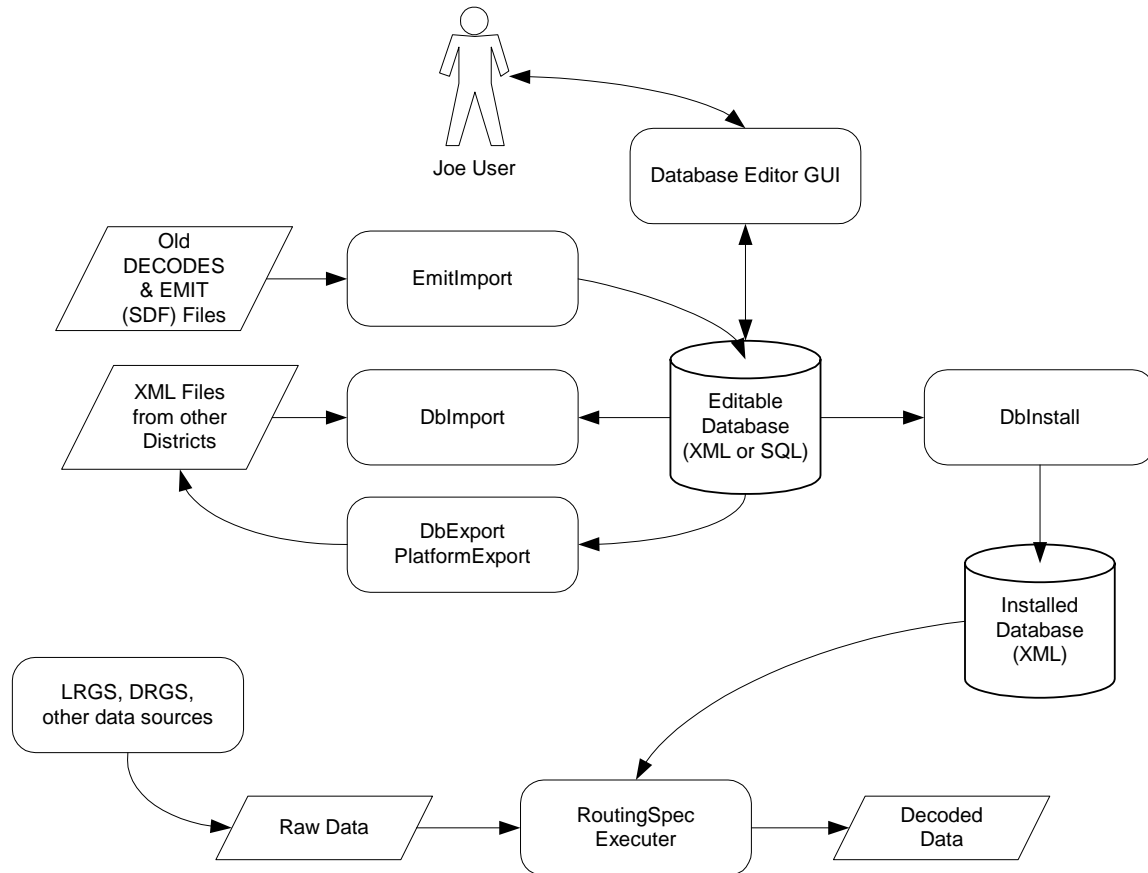


Figure 2-2: DECODES Components.

EmitImport can accept files from EMIT and legacy DECODES systems. These older software packages could export what was commonly called an SDF or Site Device File. If you currently use EMIT or an older version of DECODES, you can import your platform specifications directly into the new Java DECODES.

DbImport accepts XML files from other organizations using DECODES. A primary goal of DECODES is to encourage interagency cooperation.

Two utilities, DbExport and PlatformExport can be used to create XML files for exchange.

An extensive GUI database editor is provided for creating new DECODES specifications and modifying existing ones. The editor has features for interactively decoding raw data on-the-fly as you modify your specifications.

The DbInstall utility takes records from your editable database that you have blessed, and places them into the “installed database” for your production system.

The Routing Specs use information from the installed database to decode and convert raw data from a variety of sources.

## 2.2 DECODES Database Architecture

This chapter provides an overview of the DECODES database. For a more complete listing of elements, along with SQL and XML schema, see the DECODES 5 Database Schema Document.

We divide database records into two categories:

- **Setup Information** – These are records that should rarely change, such as standard unit conversions, data type records, etc.
- **Decoding & Converting Specifications** – You will modify these records as you add, delete, or modify platforms; pull data from different sources; integrate new back-end databases; etc. An extensive GUI editor (see Chapter 5) is provided for maintaining these records.

Database information is shown using Entity Relationship Diagrams (ERD). These diagrams show the information contained in each entity and how different entities relate (shared keys, etc.).

### 2.2.1 Setup Information

Setup information should rarely change. The DECODES distribution comes with fully-populated tables of setup information. This will be sufficient for most organizations.

Currently the only way to modify Setup Information is to modify XML files by hand. Future releases will include a setup GUI.

#### 2.2.1.1 Enumerations

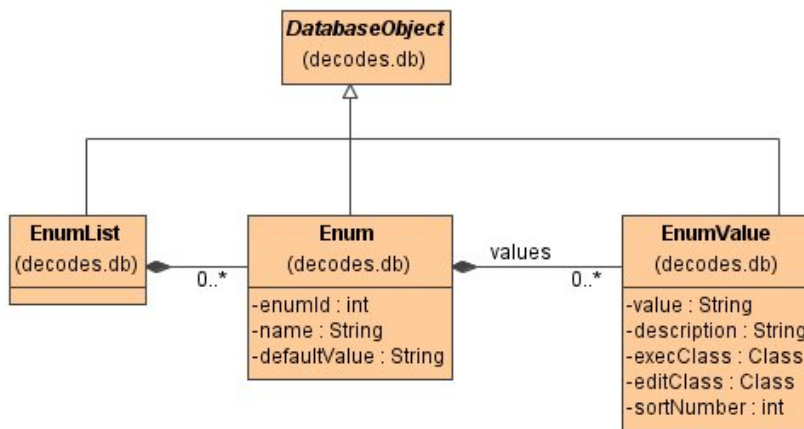


Figure 2-3: Enumerations ERD.

There are several places where an object must hold a string that must be constrained to one of several valid choices (i.e. an enumeration type). DECODES contains static tables in the database to store the valid choices for these enumerations. GUI programs can use these tables to offer pull-down menus for selection. They can also be used for validation when importing an XML file from another agency.



An enumeration is made up of a single ‘Enum’ entity and a series of associated ‘EnumValue’ entities. Think of ‘Enum’ as a type, and ‘EnumValue’ as the associated values that are valid for that type.

The Enum entity holds the name of the type. Table 2-1 lists the enumerations used in the DECODES database.

<i>Enum Name</i>	<i>Description</i>
SiteNameType	The ‘type’ values for site names. DECODES comes pre-loaded with four site name types: NWSHB5, USGS (site number), USGS-DRGS, Local.
DataTypeStandard	SHEF-PE, EPA-Code, etc.
RecordingMode	F=Fixed regular interval, V=Variable, triggered, or random.
ScriptType	Different ways to extract data from the raw platform messages. “Standard” scripts use the DECODES format statements to extract samples from your messages. This enumeration provides a hook for creating custom decoders.
TransportMediumType	GOES (either self-timed or random), File, Modem, NWSTG, etc.
DataOrder	A=ascending, D=descending. “Ascending” means that the earliest samples appear in the message first.
UnitFamily	“Metric” or “English”
UnitConversionAlgorithm	Four algorithms for converting from one EU to another: <ul style="list-style-type: none"> <li>• “none” means no-conversion-necessary. In other words, the input and output units are synonyms. Example mL and cc.</li> <li>• “linear”: <math>y = Ax + B</math></li> <li>• “USGS-Standard”: <math>y = A * (B + x)^C + D</math></li> <li>• “Poly-5”: <math>y = Ax^5 + Bx^4 + Cx^3 + Dx^2 + Ex + F</math></li> </ul> <p>Other algorithms can be easily added in the future.</p>
Measures	A list of physical quantities that are measured by sensors. Used to associate units in different unit families. For example Meters and yards are related because they both are ‘measures’ of length. This list includes “arc”, “area”, “flow”, “length”, “temperature”, “time”, “velocity”, “voltage”, and “volume”.
EquationScope	DCP, DCF, NL, SITE, ALL
LookupAlgorithm	Linear, Exponential, Logarithmic, Truncating, Rounding, or “Exact-Match”
OutputFormat	The following output formats are currently implemented <ul style="list-style-type: none"> <li>• SHEF – Standard Hydrometeorologic Exchange Format</li> <li>• SHEFIT – Intermediate format used by USACE</li> <li>• Human-Readable – compact row/column format</li> <li>• EMIT-ASCII – Compatible with the “ASCII” format produced by EMIT</li> <li>• EMIT-ORACLE – Compatible with the “ORACLE” format produced by EMIT</li> </ul>
DataConsumer	Data Consumers specify where to send data once it has been decoded, converted, and formatted: <ul style="list-style-type: none"> <li>• File – write to a specified output file</li> <li>• Directory – write each message to a separate file in a specified directory.</li> <li>• Pipe – send data to standard output, usually for piping into another program.</li> </ul>
DataSourceType	Data Sources provide raw messages to the decoder: <ul style="list-style-type: none"> <li>• File – read raw messages stored in a file</li> <li>• Directory – Each file in the specified directory should contain a single raw message</li> <li>• LRGS – Connect to an LRGS or DRS over the network and pull messages</li> <li>• HotBackupGroup – Used to specify a group of LRGS systems. If one</li> </ul>

	<p>connection fails, use another in the group.</p> <ul style="list-style-type: none"> <li>• RoundRobinGroup – Read data continually from a group of other data sources (directories, files, LRGS, etc.)</li> <li>• SocketStream - Reads a stream of messages from a TCP socket.</li> </ul>
--	--

**Table 2-1: Enumerations in the DECODES Database.**

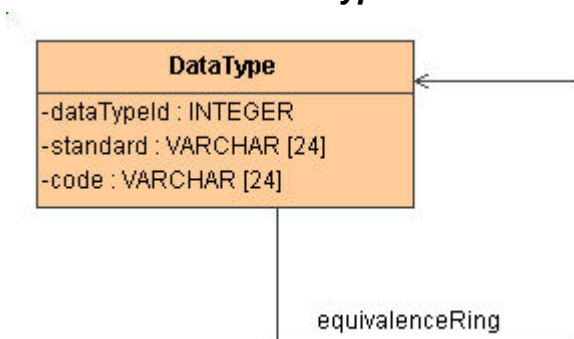
### 2.2.1.2 Time Zones

The initial release 5.0 of DECODES included database entities for time-zones. These have been removed in DECODES 5.1 (and later).

DECODES now uses the internal Java time zone definitions. Routing Spec and Site entities hold references to time zones by storing a “TimeZone ID” recognized by Java.

A list of the time zones supported by Sun Microsystem’s Java 1.4 is provided in Appendix B. You can also construct a custom time zone by specifying an offset to GMT. For example, “GMT-06:00” would mean 6 hours behind GMT, corresponding to Central Standard Time with no support for Daylight time.

### 2.2.1.3 Sensor Data Types



**Figure 2-4: Data Types ERD.**

DECODES can accommodate different systems for representing data types. Each data type is denoted by a ‘standard’ and a ‘code’. For example “SHEF:HG” could be used for stream stage values.

The database also contains records that assert an equivalence between two data types. For example, the SHEF code HG is equivalent to the EPA code 00065. These records allow the software output data in different coding standards, regardless of the agency maintaining the DCP.

For example, USGS may prepare platform records using EPA codes. USACE could use these records without modification, telling DECODES to convert all the data types to SHEF.

### 2.2.1.4 Engineering Units

The DECODES database contains a list of commonly-used of standard and English engineering units. It also contains records that specify conversions between them. Figure 2-5 shows these database entries.

Conversions are performed with one of the following algorithms:

- None (means that EUs are synonyms like cc and ml)
- Linear
- USGS Standard Equation
- 5<sup>th</sup> order polynomial.

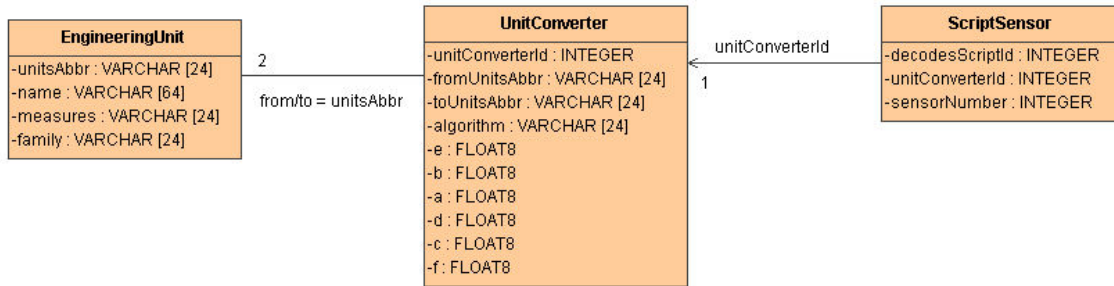


Figure 2-5: Engineering Units & Converters ERD.

### 2.2.2 Properties

There are several places in the model where we left “hooks” for arbitrary information that may be used by a particular agency. This type of information is stored in a set of properties associated with some other entity.

## 2.2.3 Decoding, Formatting, and Converting Specifications

### 2.2.3.1 Equipment Models

The EquipmentModel entity captures information about a piece of hardware, such as a platform, a transport medium, or a sensor.

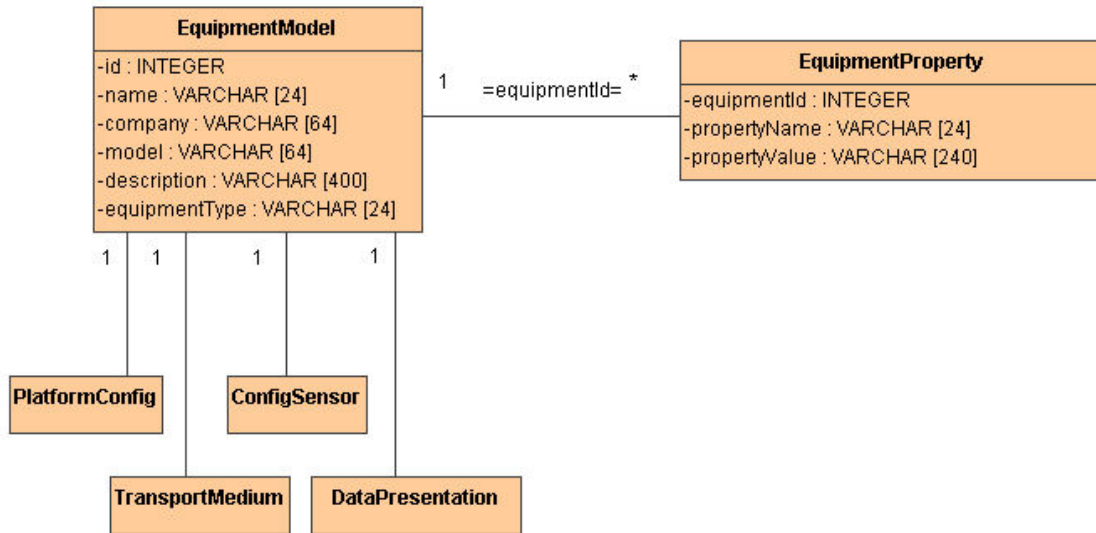


Figure 2-6: Equipment Model ERD.

The EquipmentModel entity stores information about the model and manufacturer of these pieces of equipment.

An equipment model may have an arbitrary set of properties. An example of a likely property might be time-ordering for platforms.

### 2.2.4 Sites & Site Names

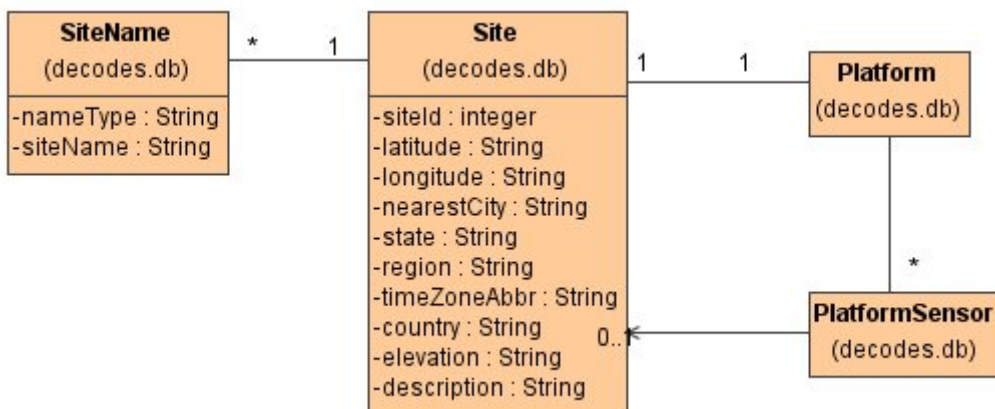


Figure 2-7: Sites and Site Names ERD.

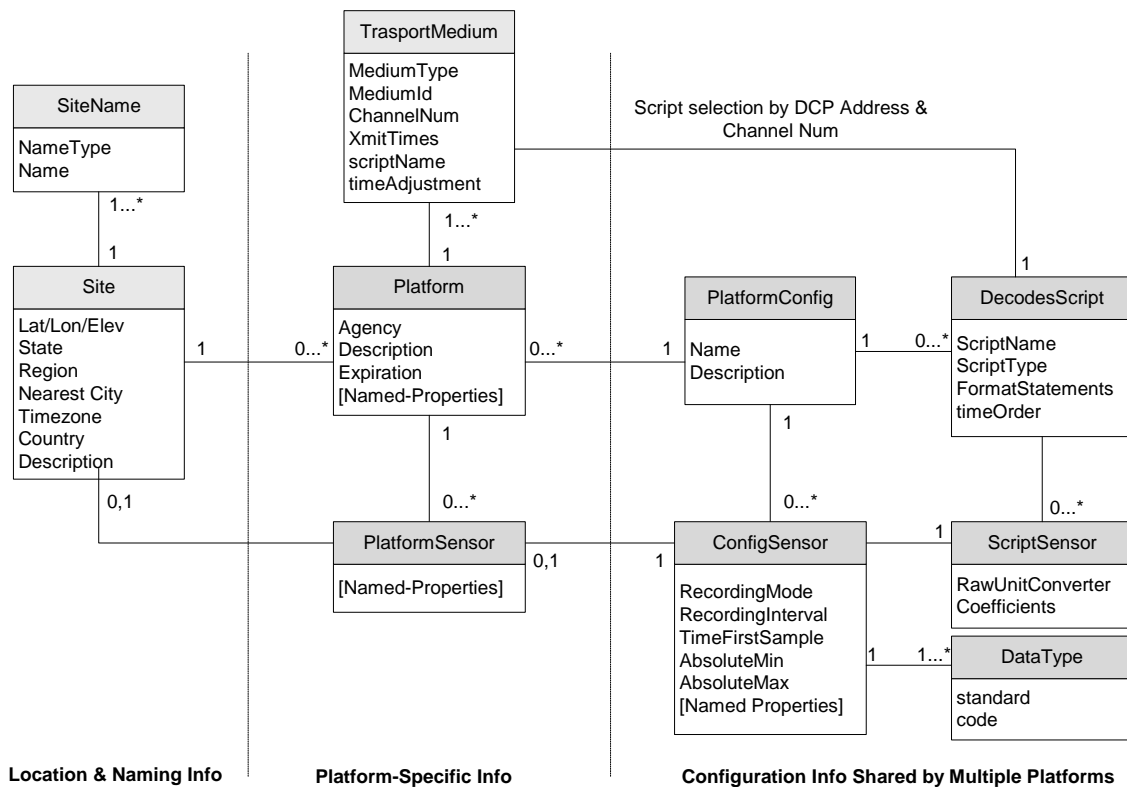
In DECODES, a “Site” is simply a location. This is a little bit different from the concept in EMIT where a site is synonymous with a platform. In DECODES, a site may contain more than one platform, and a platform may be reporting sensors at different sites.

Consequently, the Site entity contains attributes that describe the location only.

A site can have many names. For example, USACE typically uses the National Weather Service HB5 name. USGS uses a numeric site ID. Other agencies may define a “local” name specific to their organization.

One or more sensors can exist at a site. Normally these are associated directly with the platform at the same site. However, sometimes a sensor can be associated with a platform at a different site.

## 2.2.5 Platform Information



**Figure 2-8: Platform Information ERD.**

The “Platform” entity is the central entry point for decoding. The decoder recognizes an incoming message by its TransportMedium information. For example a GOES DCP has a unique DCP address. Each TransportMedium record points to a particular “Platform”.

The Platform has a PlatformConfig which determines the sensors installed in the platform, and the scripts used to decode information from that platform.

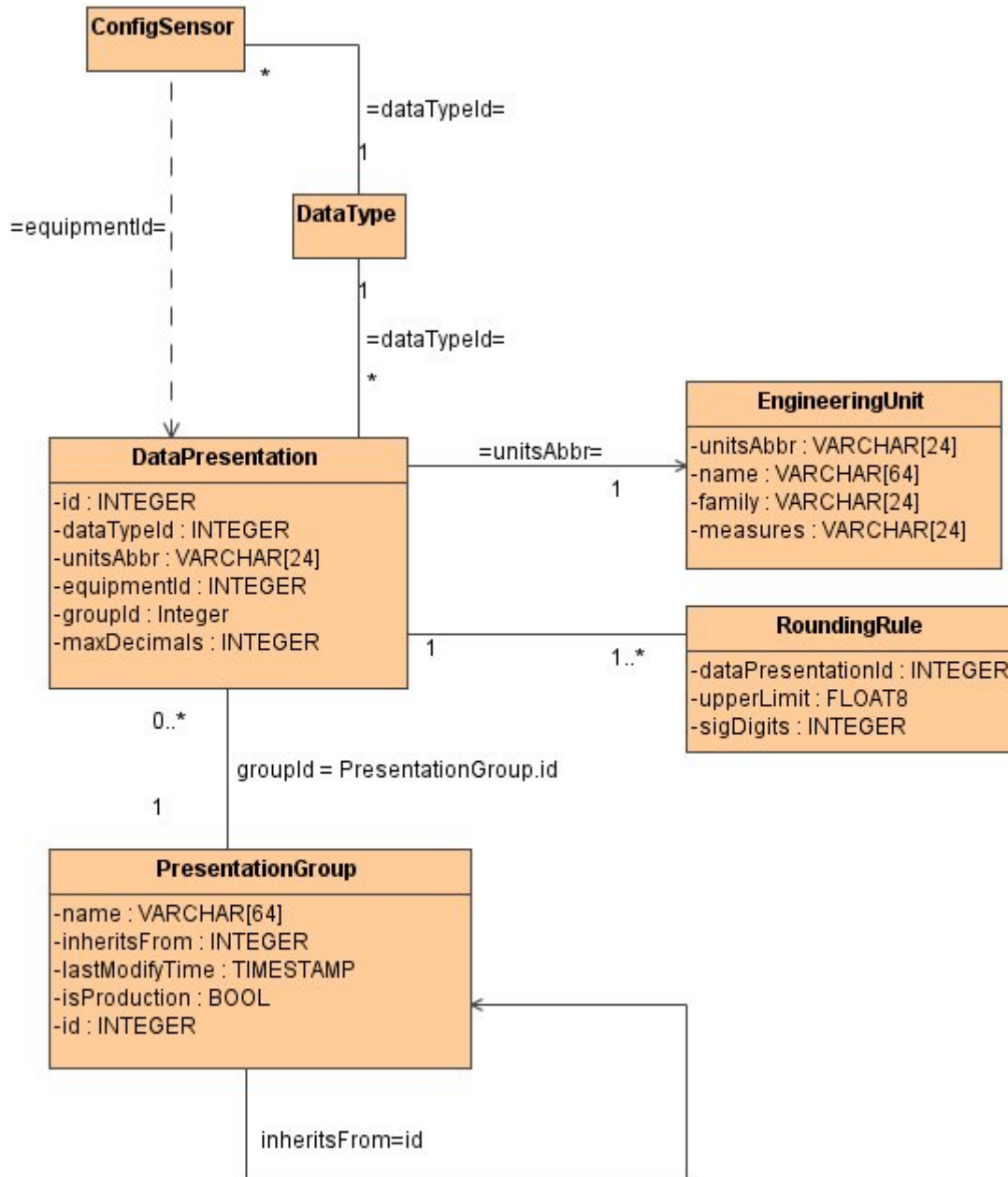
Each PlatformConfig has a series of ConfigSensors. Each ConfigSensor has a unique “sensorNumber”. Each ConfigSensor has a data type and may have a group of properties (ConfigSensorProperties).

The same PlatformConfig may be used by several platforms. For example, a group of Sutron 8200s which have identical sensors and message formats may share the same PlatformConfig.

Platform-specific information about sensors may be stored in the PlatformSensor and PlatformSensorProperty entities.

A DecodesScript contains the instructions for decoding messages from a platform that were received over a given transport medium. For example, a platform may have a data logger and a GOES transmitter. The GOES DCP messages would be decoded with one script, and the data logger files decoded with a different one. The choice of which script to use is based on which "TransportMedium" the data was received on.

## 2.2.6 Presentation and Unit Conversion



**Figure 2-9: Presentation and Unit Conversion ERD.**

Refer back to the diagram in the Platform Information section. Note that the “ScriptSensor” points to a “UnitConverter”. In that case, the UnitConverter translates the raw value contained in a message into its initial Engineering Units value. A typical case would be a stage sensor that reports in tenths of inches. The initial unit converter would convert the value to inches by dividing by 10.

In Figure 2-9, a UnitConverter converts from one EU into another. A library of standard unit conversions is built into DECODES. This enables you to specify which units you want to output, regardless of who created the decoding specification.

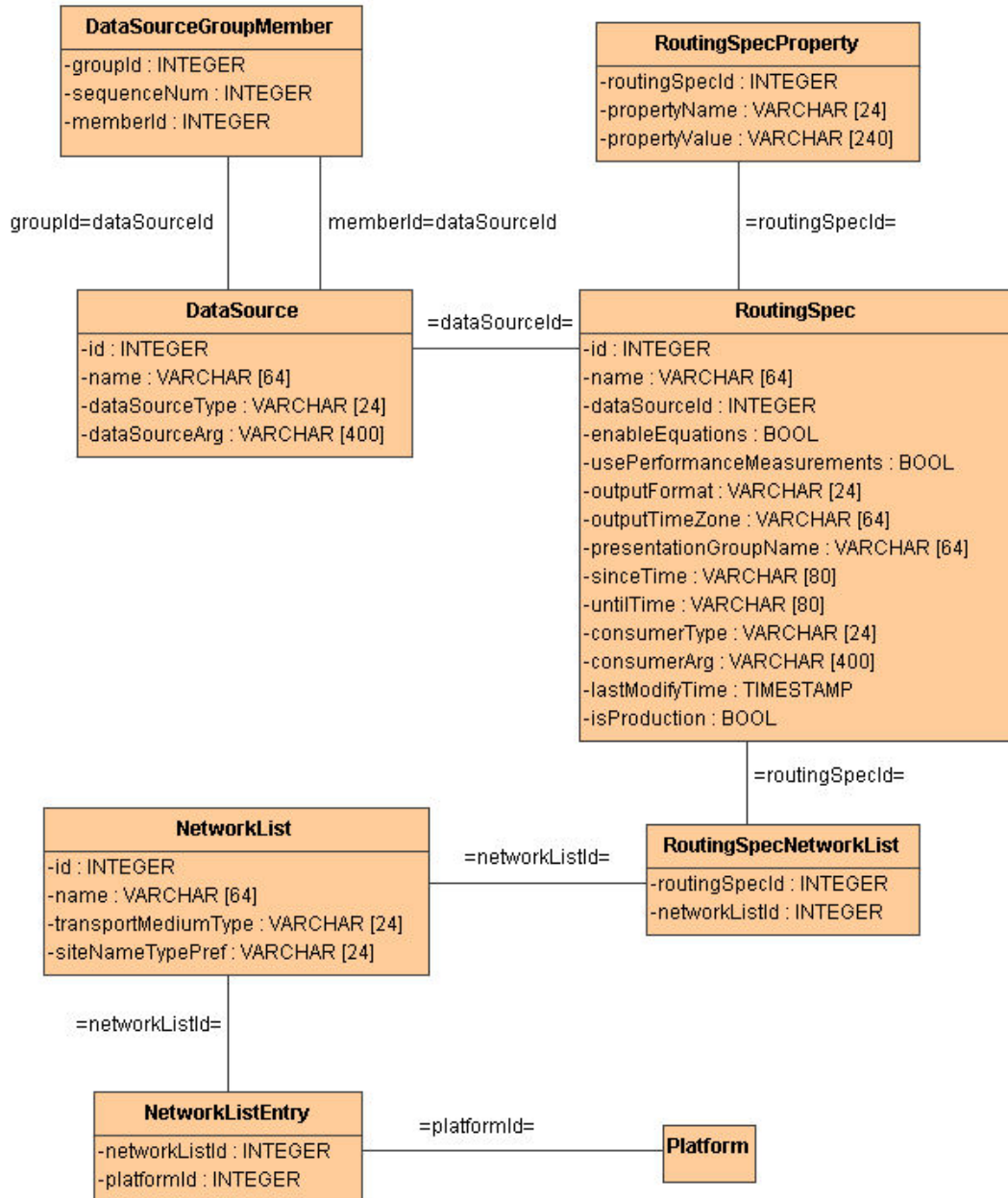
An “EngineeringUnit” belongs to a family (English or metric). It also has a unique abbreviation (e.g. “mm”) and a full name (“millimeters”). It uses one of the algorithms listed in Table 2-1.

A “DataPresentation” entity also uses a set of RoundingRules to determine the display resolution for a given data type. For example, a DataPresentation entity for stage values might assert that values should be output in inches. and...

- If the value is between 0...1, use 3 decimal places
- If the value is between 1 and 10, use 2 decimal places
- If the value is above 10, use 0 decimal places.



## 2.2.7 Routing Specification Information



**Figure 2-10: Routing Specification ERD.**

DECODES uses a “Routing Specification” to determine:

- Where to get data from (Data source entities)
- Which data to get (Network Lists)
- How to format data for output
- Where to send it once it is decoded and converted

A “NetworkList” in DECODES is analogous to a network list used by DOMSAT systems. It is simply a list of transport media (i.e. DCP addresses or NESS IDs).

## 3. Configuring DECODES

### 3.1 The “*decodes.properties*” File

The release directory contains a file called “decodes.properties”. This file contains “name=value” pairs, one per line. The options are shown in Table 3-1. “Default Value” is the value that will be used by the software if the property is missing from the file.

The installation procedure will create a “decodes.properties” file based on selections you made in the dialogs.

<i>Property Name</i>	<i>Default Value</i>	<i>Description</i>
DatabaseType	“none”	This is the type for the installed database. The value should be either “XML” or “SQL”. In the future, we plan to add “XMLURL”, for an XML database accessible via the Internet.
DatabaseLocation	N/A	This is the location for accessing the installed database. For “XML”-type databases, this is a directory name containing the DECODES directory tree. For “SQL”-type databases, this is a JDBC database-url. See section 5.2 for more information about the format of this value. For “XMLURL”-type databases, this is a URL.
SiteNameTypePreference	NWSHB5	Specifies the “preferred” type for DCP names. By default this is the Handbook-5 standard used by the National Weather Service.
EditDatabaseType	“xml”	The is the type for the Editable database. The same values allowed for the DatabaseType property are allowed here; viz “XML” or “SQL”.
EditDatabaseLocation	N/A	This is the location for accessing the editable database. The same values allowed for the DatabaseLocation property are allowed here.
EditOutputFormat	“Human-Readable”	Output format to test decoding scripts within dbedit.
EditPresentationGroup	N/A	Presentation group used to format samples when testing decoding scripts within the editor.
EditTimeZone	“UTC”	Time zone used when decoding sample data within the editor. Using UTC makes it easy to correlate sample times with the DCP message time stamp.
jdbcDriverClass	org.postgresql.Driver	Full Java class name of the JDBC driver here.
SqlKeyGenerator	Java Class Name	Class name of SQL key generator. See section 3.1.1.
SqlDateFormat	String Template	Format template for date/time stamps. See section 3.1.2 below.
sqlTimeZone	String	Time zone in which SQL will represent date/time stamps. If blank, it will be derived from system settings.
RoutingStatusDir	Directory Name	Default: \$DECODES_INSTALL_DIR/statmon By default, your routing specs will periodically place their status in this directory. The “Routing Status Monitor” Web Application can be used to make this information visible via a web page.
DefaultDataSource	Data Source Name	Default: “drot.wcda.noaa.gov”. This is used in the database editor (dbedit) to retrieve sample messages to test your decoding.
TransportMediumTypePreference	String	Used in the database editor list panels to choose which of the (possibly several) transport media to display.
DataTypesPreference	String	Used by editor and some output formatters to choose which of the (possibly several) data types to display.
decwizTimeZone	String	Time zone used in the decoding wizard displays.
decwizDebugLevel	0, 1, 2, 3	Debug level in trace log 0=no debug info, 3=most verbose.
decwizOutputFormat	String	Default = “stdmsg”.
decwizRawDataDir	Directory	Default location to move raw data files into.
decwizDecodedDataDir	Directory	Default location to save decoded data in.
decwizSummaryLog	File Name	Default file to append decoding summaries to.
hdbSiteDescriptions	True or False	Default = false. Set to true to automatically place the preferred name at the beginning of the description. HDB requires this.

**Table 3-1: DECODES Property Values.**

### **3.1.1 Surrogate Keys Generated by SQL Database**

If you are using a SQL database, DECODES uses a class that you specify to generate surrogate keys. The class must implement the `decodes.sql.KeyGenerator` interface.

In the `decodes.properties` file, set the `SqlKeyGenerator` property as follows:

- For PostgreSQL, use: `decodes.sql.SequenceKeyGenerator`
- For Oracle, use: `decodes.sql.OracleSequenceKeyGenerator`

Other key generators may be added in the future.

### **3.1.2 Date Formats in SQL Database**

SQL database products vary in the format used for storing and retrieving date/time values. The Java code uses a “`java.text.SimpleDateFormat`” object to format and parse dates. See the Java API manual page for a complete description on possible strings.

In the `decodes.properties` file, set the `SqlDateFormat` property as follows:

- For PostgreSQL, use: `yyyy-MM-dd HH:mm:ss`
- For Oracle, use: `dd-MMM-yyyy HH:mm:ss`

### **3.2 SQL Database Format Changes**

The database format has changed slightly for each major release (5, 6, and 7).

**If you use a DECODES XML Database on a Single Machine:** Don't Worry! The new DECODES programs will read the old XML files just fine. When you edit and commit an entity, the XML files will be saved with the new format.

**If you use a DECODES XML Database Shared between Multiple Machines:** You will want to upgrade all of the machines to the new release of DECODES. A problem can arise if you save a database element with version-7 format and then try to use it with an older release.

If you are nervous about upgrading everything at the same time, we recommend making a copy of your old database hierarchy. Run version 7 on a copy of the database until you are comfortable that it will work operationally.

**If you use a DECODES SQL Database:** Several of the table definitions have changed for versions 6 and 7. The new software will work fine on an old database but you may not be able to use some of the new features. We recommend that you create a new DECODES (version 7) database, as described in the DECODES Installation Guide. Then use 'dbexport' to dump the old database to an XML file. Finally use 'dbimport' to import everything to the new database.

## 4. Maintaining the DECODES Database

Take a moment to refer back to Figure 2-2. Note that the Editable Database is separate from the Installed database. This section describes tools that you will use to:

- Import EMIT and pre-release-5 DECODES files into the editable database.
- Import Platform and other database XML files from other organizations using DECODES 5+.
- Create a new Editable database from scratch.
- Install components from the Editable Database to the Installed Database

### 4.1 Initializing Your Editable DECODES Database

This section describes how to initialize and populate your first editable database.

#### 4.1.1 Standard XML Setup Files

If you are using an SQL database as your editable database, refer back to Chapter 5. This section is for the older XML editable database.

The setup files contain collections of information that probably does not need to change from one organization to another. Currently the only way to edit this information is to use a text editor to modify the XML files.

The files are all found under `$DECODES_INSTALL_DIR/edit-db`:

**datatype/DataTypeEquivalenceList.xml:** Described in section 2.2.1.3, this file contains definitions for common SHEF and USGS/EPA numeric type codes. You may want to edit this file if you use uncommon or custom type-codes.

**enum/EnumList.xml:** Described in section 2.2.1.1. You will probably not need to edit this file unless you are adding custom Java code to the DECODES system.

**eu/EngineeringUnitList.xml:** Described in section 2.2.1.4, this file contains records which describe most commonly-used English and Metric units. It also contains conversion algorithms & coefficients to convert between units that measure the same physical parameter.

## 4.1.2 Importing Data from EMIT or Pre-Release-5 DECODES

### Synopsis:

```
emitimport <options> file1 file2 . . .
```

### Options:

-t <i>name-type</i>	Sets the preferred site-name type to the specified value. The default is NWSHB5.
-d <i>debug-level</i>	Level should be 1, 2, or 3 from the least to most verbose.
-v	Validate only: Do not actually import any data. Just issue warnings about conflicts and parsing errors.
-x	Instead of importing into your editable-database, create an XML file containing the converted data. With this option, the program is a translator rather than an importer.
-o	Keep old records on conflict. Default is to overwrite old records with new ones.

### Description:

This program can accept SDF files (SDF stands for Site Device File) from EMIT or pre-release-5 DECODES. It can also accept network list files from LRGS or DRS systems.

It creates new DECODES database records and places them into the editable database.

This program writes log messages to a file called “util.log” in the current directory.

### Examples:

DECODES contains several sample files in the to\_import sub-directory. To import these files type:

```
cd $DECODES_INSTALL_DIR
emitimport to_import/*
```

### 4.1.3 Importing XML Data from Other DECODES Sites

#### Synopsis:

```
dbimport <options> file1 file2 . . .
```

#### Options:

-d <i>debug-level</i>	Level should be 1, 2, or 3 from the least to most verbose.
-v	Validate only: Do not actually import any data. Just issue warnings about conflicts and parsing errors.
-o	Keep old records on conflict. Default is to overwrite old records with new ones.

#### Description:

This program accepts XML files that were created by the export utilities described in section 4.4. Imported records are added to your editable database.

DbImport ignores ‘setup’ records that are part of a large database dump. This allows you to exchange a dump of your entire database with other users, without fear that special changes they make in the database structure will affect you. The following record types are only imported if the corresponding file is placed on the command line:

- Enumeration Records – found in the file enum/EnumList.xml
- Engineering Units and Conversion – found in the file eu/EngineeringUnitList.xml

This program writes log messages to a file called “util.log” in the current directory.

#### Examples:

```
pxport -a > platform-dump.xml  
...at a different organization  
dbimport platform-dump.xml
```

## 4.2 Interactively Editing the Database

The script ‘dedit’ starts the interactive Database Editor GUI. See section 5 for instructions on using this program.



## 4.3 Updating the Installed Database

### Synopsis:

```
dbinstall <options>
```

### Options:

`-d debug-level`      Level should be 1, 2, or 3 from the least to most verbose.

### Description:

The purpose of this program is to copy records that are ready for production from your Editable database into your Installed database.

This program writes log messages to a file called “util.log” in the current directory.

The dbinstall program copies all of the setup information from your editable database into your installed database.

If you examine the diagrams for the various database entities in you will see “isProduction” parameters in the following record types:

- EqTable
- EquationSpec
- Platform
- PresentationGroup
- RoutingSpec

For these record types, only the entities where the “isProduction” value is true will be copied. You can set this value for selected records in the Database Editor GUI (See section 5)

As a time-saving alternative, special script called ‘markproduction’ has been prepared. This script sets the isProduction flag to true for all records in the editable database.

### Examples:

```
...after importing records to the editable database  
dbinstall
```

## 4.4 Exporting Platforms to XML Files

### Synopsis:

```
pxport <options>
```

### Options:

<code>-d <i>debug-level</i></code>	Level should be 1, 2, or 3 from the least to most verbose.
<code>-n <i>network-list</i></code>	Export platforms referenced by the named network list.
<code>-s <i>site-name</i></code>	Export the platform record for a specific site.
<code>-a</code>	Export all platforms.
<code>-c <i>config-name</i></code>	Export platforms that use a given platform configuration.
<code>-i</code>	Export from the installed database. The default is to export from the editable database.

### Description:

This program writes XML records containing platforms (and all subordinate records such as site, config, script, and transport media). Records are written to standard output.

Multiple instances of the above options are acceptable. See examples below.

This program writes log messages to a file called “util.log” in the current directory.

### Examples:

Dump all platforms to a single XML file:

```
pxport -a > platform-dump.xml
```

Export three specific sites:

```
pxport -s TCLG1 -s HUDG1 -s LHMG1 > threesites.xml
```

Export platforms referenced by Atlanta’s network list:

```
pxport -n Atlanta > Atlanta-platforms.xml
```

## 4.5 Exporting the Entire Database

You can export the entire database to an XML file with the 'dbexport' command:

```
dbexport > file By default exports the editable database.
```

```
dbexport -i > file Option to export installed database
```

This command is very useful for taking periodic backups of the DECODES database.

## 4.6 Other Database Utilities

### 4.6.1 Creating the Platform Cross-Reference File

If you look in the edit-db/platform directory you will see that each platform is stored in a separate file named with a 'p' followed by a numeric ID assigned to the platform.

Each database assigns an arbitrary numeric key ID field to platforms as they are added to the database. For the most part this key is invisible to you and you shouldn't have to worry about it.

Also in this directory is a file called PlatformList.xml. This file is a cross reference that maps site names, configuration names, and DCP addresses to each platform.

If you suspect that your cross reference file has been corrupted, you can rebuild the PlatformList.xml file directly. This can happen if you add files to the platform directory using tar, zip, cp, copy, etc.

To run the program type:

```
java decodes.xml.CreatePlatformXref database-root
```

where 'database-root' is the path to the top of the XML database. For example, if you want to build the cross reference in your editable database, and you installed DECODES in /usr/local/decodes, type:

```
java decodes.xml.CreatePlatformXref /usr/local/decodes/edit-db
```

## 4.6.2 Creating LRGS-Style Network List Files

The LRGS and DRS support an alternative file format for network lists. LRGS Network Lists are ASCII files containing a list of DCP addresses, one per line:

```
DCP-Addr:DCP-Name Comment...
```

The line starts with a (8 hex-digit) DCP address, followed by a colon, followed by a blank-delimited DCP Name, followed by a free-form comment field.

When you run a routing spec that uses an `LrgsDataSource`, the software converts any network lists you specified into the above format, and sends them to the server. The LRGS DCP Data Server (DDS) then only sends the messages from those platforms.

Older SCO-DRS servers do not support network list transfers. If your data source is a SCO DRS, do the following:

1. Add a property called “sendnl” set to the value “false” to the data source record.
2. Generate the LRGS-style network list file using the “nl2lrgs” utility (see below).
3. Use FTP or some other file-transfer mechanism to copy the list onto the SCO DRS. Place it in the ‘drs’ home directory (/usr/drs).
4. Repeat steps 2 and 3 every time the list is modified.

### The “nl2lrgs” utility

```
nl2lrgs [-e] list1 list2 ...
```

The “nl2lrgs” will create a file in the current directory for each list specified. The file will have the same name as the network list, with a “.nl” extension.

The “-e” argument forces the utility to pull the network list from your editable database. The default is to use the installed database.

## 5. The DECODES Database Editor

### Synopsis:

```
dbedit <options>
```

### Options:

-d *debug-level*      Level should be 1, 2, or 3 from the least to most verbose.

-E *databaseLoc*      Edit the XML database at the specified location. This overrides the editable database location specified in your DECODES Properties file.

The editor starts as shown in Figure 5-1.

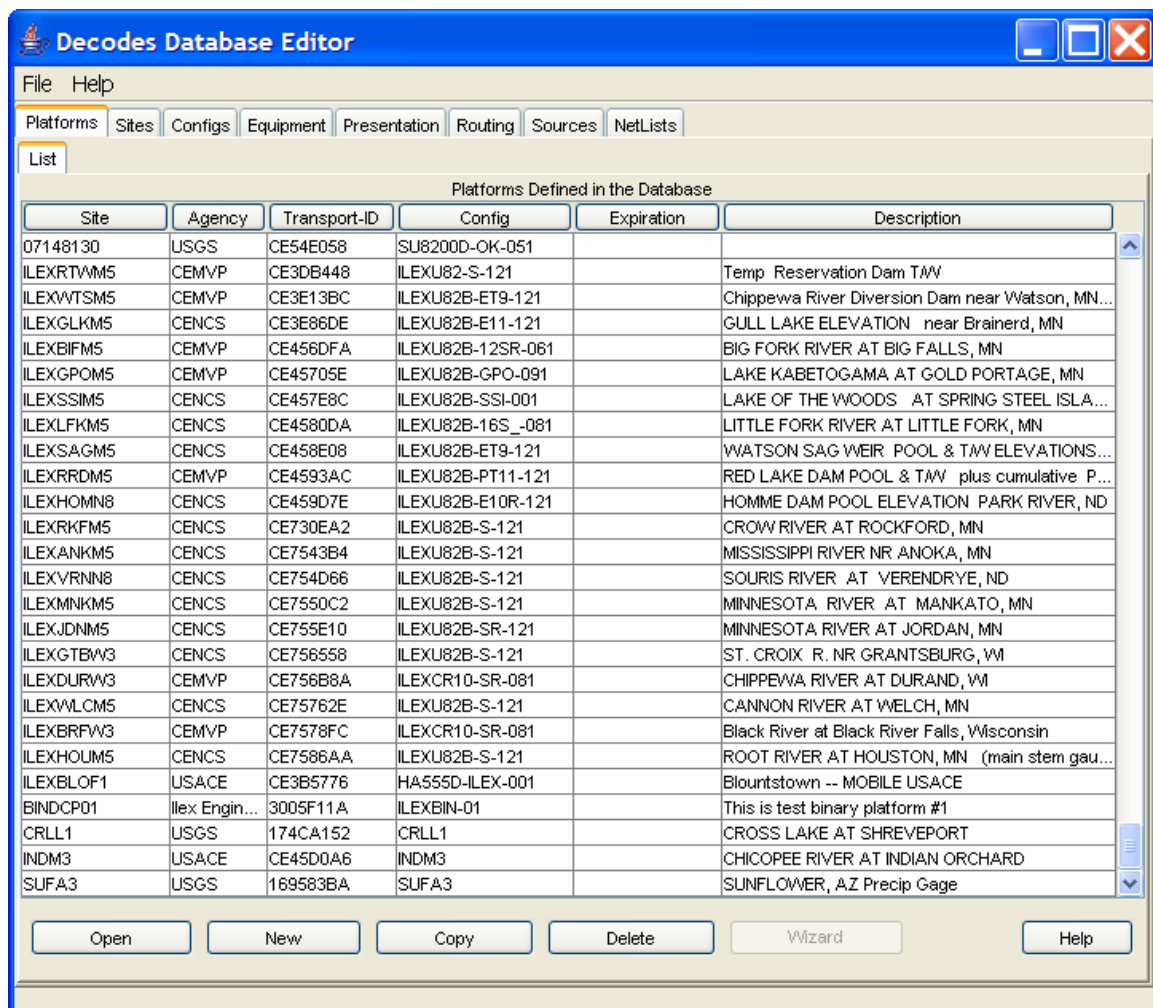


Figure 5-1: Database Editor Platform List Screen.

## 5.1 GUI Organization

A row of Tabs appears along the top corresponding to the different kinds of records in the database (Platforms, Sites, Configs, etc.)

### 5.1.1 List Panels in General

Underneath each of those tabs you will see a “List” tab. In Figure 5-1, the Platform tab is selected, so we see the List of platforms.

Click on the column header in the List tab to sort the elements by the columns value. In Figure 5-1 the ‘Site’ column header was clicked, so we see elements sorted by Site name.

Along the bottom of the List tab you see buttons with the following labels:

- Open** To edit a database record, click on it in the list and press Open
- New** Press new to create a new database record.
- Copy** To copy a database record, click on it in the list and press Copy. You will be prompted for a name for the copy.
- Delete** To delete a database record, click on it in the list and press Delete.
- Wizard** This is a placeholder for future features. We plan to implement wizard dialogs to guide you through creating various types of database entries. Currently the only ‘Wizard’ implemented is under the Configs tab. This Config wizard is simply a prototype that doesn’t do anything currently.
- Help** Coming soon: This button will bring up a context sensitive help screen.

### 5.1.2 Edit Panels in General

When you **Open** a record, a new tab appears to the right of the list tab. For example, Figure 5-2 shows the result after we do the following:

- Select the **Configs** top-level tab.
- Select the record HA555A-GA-008 from the list
- Press the **Open** button.

Separate edit screens have been implemented for each type of database record. In this edit screen you would change all of the parameters for HA555A-GA-008.

Notice the bottom of the Edit Panel. The **Commit** button writes the record back to the database. You can do this at any time. It does not close the panel.

The **Close** button closes the edit panel. If you have made changes to the record you will be asked if you want to save them.

### 5.1.3 Exiting the Editor

You can exit the editor by selecting File-Exit or by closing the window. If you have edit panels open in which changes have not been saved, you will be forced to close these panels before you can exit.

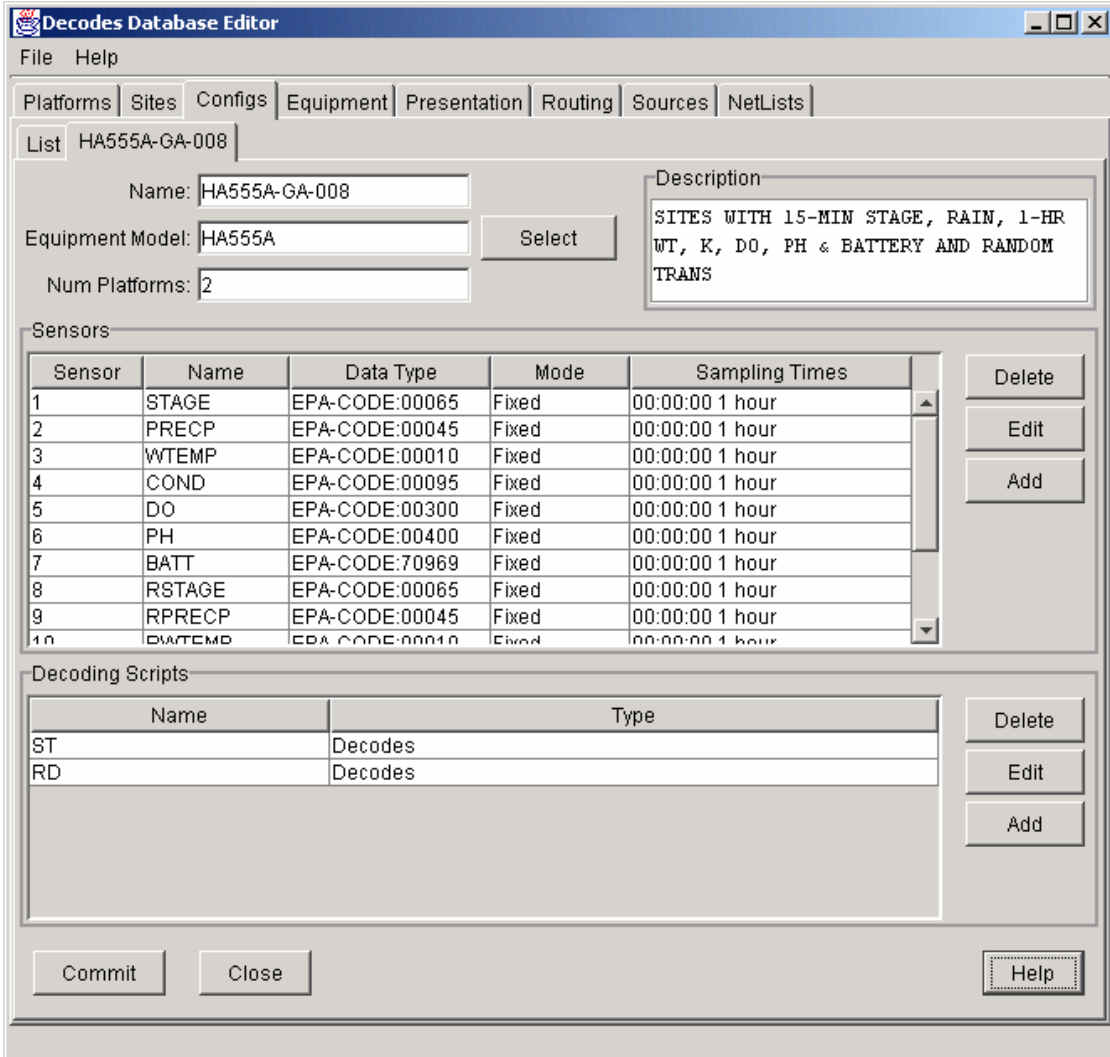


Figure 5-2: Platform Config Edit Panel.

## 5.2 The Platform Edit Panel

The platform edit panel is shown in Figure 5-3.

Every platform is associated with a site. Press the Site Choose button to bring up a dialog in which you can select a site.

Every platform is associated with a configuration. Press the Config Choose button to make this association. Be careful: Sensors are defined in the configuration.

The Owner Agency and Description are simple free-form type-in fields. They are informational and not used by other DECODES software.

If you want this platform to be placed into the 'install-database' by the dbinstall utility (see section 4.3), check the 'Production' checkbox.

Platforms change over time (sensors are added, removed, etc.). You can capture a historical version for a platform by pressing the 'Make Historical Version' button. Each historical version is a separate record with a specified Expiration time.

### 5.2.1 Platform Sensors

Information about sensors which is specific to a platform is stored here.

- If a sensor on this platform is actually located at a different site, you can associate the sensor with a site. In most cases, however, the "Actual Site" field is blank, meaning that this sensor is at the same site as the platform.
- You can associate arbitrary properties with a sensor to be used by downstream DECODES modules. In the example shown, the USGS DBNO and DDNO are associated with each sensor.

You can do simple scale and offset adjustments to sensor values on a platform specific basis. To do this add any of the following properties to the Platform Sensor Properties set:

- preoffset – A number added to each sensor value. This is done prior to scaling.
- scale – A number to be multiplied by each sensor value.
- offset – A number added to each sensor value (after scaling).

You can set a "minimum" and "maximum" property on each sensor. The value should be the absolute min/max values that are to be considered valid for this sensor on this platform. Values outside this range will be discarded and considered missing.

Note – minimum and maximum checks are done after offset and scaling.

### 5.2.2 Transport Media

Transport Media define how the data from this platform is retrieved. The data may need to be decoded differently depending on whether it was received over DOMSAT, DRGS, or EDL file, even though it came from the same platform.

The example shown shows two transport media for GOES-Self-Timed on channel 31.



You can add or delete transport media by clicking the buttons to the right of the list. Clicking Edit brings up the dialog shown in Figure 5-4.

Note that in this dialog, you associate each transport medium with the name of a “Script” which will be used to decode the data. Scripts are discussed more in section 5.4.

Decodes Database Editor

File Help

Platforms Sites Configs Equipment Presentation Routing Sources NetLists

List CE45705E

Site: local-ILEXGPOM5 Choose Last Modified: 11/15/2003 02:37:31

Config: ILEXU82B-GPO-091 Choose Expiration:

Owner Agency: CEMVP Production

Platform Properties Make Historical Version

Description

LAKE KABETOGAMA AT GOLD PORTAGE, MN

Platform Sensor Information

Sensor	Name	Actual Site	Properties
1	elev		minimum=1000., maximum=3333.
2	battery		

Clear Sensor Site

Select Sensor Site

Sensor Properties

Transport Media

Type	ID	Script Name	Channel
GOES-Self-Timed	CE45705E	ST	31

Add

Edit

Delete

Commit Close Help

Figure 5-3: Platform Edit Panel.

**Edit Transport Medium**

Transport Medium for Platform: nwshb5-CR114

Medium Type: goes-self-timed

Medium Identifier: CE26E000

Equipment Model: (none)

Decoding Script: ST

Time Zone: CST

GOES DCP Channel Parameters

Channel Number: 17 (1...266)

1st Transmission Time: 03:31:00 (HH:MM:SS)

Transmit Interval: 04:00:00 (HH:MM:SS)

Transmission Duration: 00:01:00 (HH:MM:SS)

Time Adjustment: 0 (# Seconds)

Preamble: Short

OK Cancel

**Figure 5-4: Transport Medium Edit Dialog.**

### **5.3 The Site Edit Panel**

Figure 5-5 shows an example of the Site Edit Panel. Recall that in DECODES, a site is simply a location with one or more names. The example shows a site near Athens, GA that has three names: A USGS station number of 02217500, a NWS Handbook 5 name of ATHG1, and a DRGS name of “ATHENS”. On the right are type-in fields for descriptive information about the site.

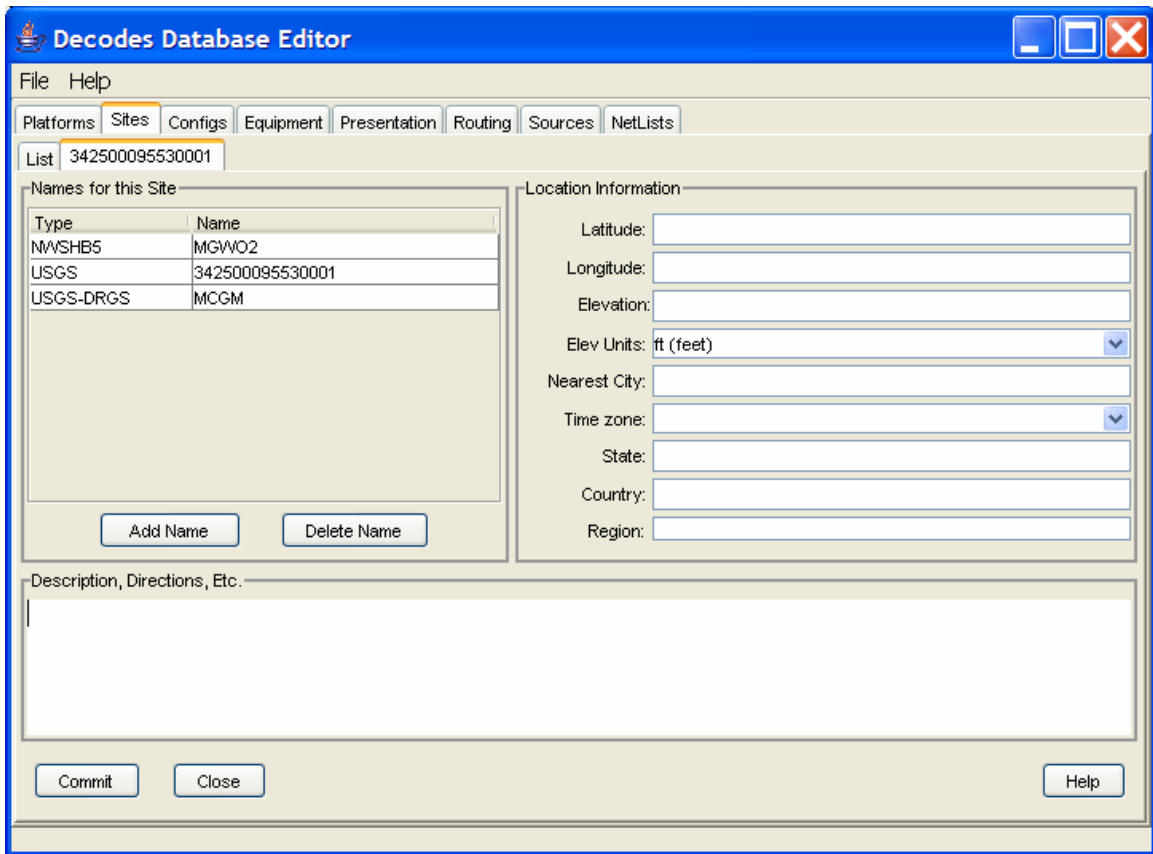


Figure 5-5: The Site Edit Panel.

## 5.4 The Platform-Config Edit Panel

Figure 5-6 shows an example platform configuration edit panel for a DCP maintained by the USGS. Configurations are associated with hardware. In this case it is a Sutron model SU8200D DCP. Press the Equipment Model Select button to change this association.

As a convenience, this panel shows you the current number of platforms that are using this configuration. This may be important if you plan to make modifications. Your modifications will effect all platforms using the config.

The center of the panel contains a list of Sensors defined in this configuration. Using the buttons to the right, you can Delete, Edit, or Add sensors in this list. If you edit or add a sensor, you will see the dialog shown in Figure 5-7.

At the bottom of the panel you see a list of decoding scripts. Decoding scripts do the work of extracting sensor samples from your raw message. Using the buttons to the right, you can Delete, Edit, or Add scripts in this list. If you edit or add a script, you will see the dialog shown in Figure 5-8

A new button has been added for DECODES Version 6.0, labeled “Add DCP PMs”. This button will add canned sensors used by USGS for “performance measurements”. These map to the values in the DOMSAT header so that they can be treated like sensors. These additional sensors will start with sensor number 100. They are only used by the USGS standard-message output formatter.

The screenshot shows the 'Decodes Database Editor' window with the 'Configs' tab selected. The configuration name is 'SU8200D-OK-027' and the equipment model is 'SU8200D'. The number of platforms is 1. The description field contains: 'ASCII FORMAT 16 values of 30 min stage= HG 16 values of 30 min prec = PC YK NOT DECODED 1 bv = VB . . 07158000 CIMARRON'. The Sensors table lists 5 sensors with their names, data types, modes, and sampling times. The Decoding Scripts table lists two scripts: 'RD' and 'ST', both of type 'Decodes'. Buttons for 'Delete Sensor', 'Edit Sensor', 'Add a sensor.', 'Add DCP PMs', 'Delete', 'Edit', 'Add', 'Commit', 'Close', and 'Help' are visible.

Sensor	Name	Data Type	Mode	Sampling Times
1	STAGE	EPA-CODE:00065	Fixed	00:00:00 1 hour
2	PREC	EPA-CODE:00045	Fixed	00:00:00 1 hour
3	BATVT	EPA-CODE:70969	Fixed	00:00:00 1 hour
4	STAGE	EPA-CODE:00065	Fixed	00:00:00 1 hour
5	PREC	EPA-CODE:00045	Fixed	00:00:00 1 hour

Name	Type
RD	Decodes
ST	Decodes

Figure 5-6: Platform Config Edit Panel.

Configuration:  Sensor:

Sensor Name:

Standard:  Code:

Data Types Standard:  Code:

Standard:  Code:

Valid Range - Min:  Max:

Recording Mode:

1st Sample Time:  (HH:MM:SS)

Sampling Interval:  (HH:MM:SS)

Equipment Model:

Properties

Name	Value
StatisticsCode	11

**Figure 5-7: Edit Config Sensor Dialog.**

### 5.4.1 The Decoding Script Edit Dialog

This dialog, shown in Figure 5-8 is one of the most important screens in the editor. Anyone who has worked with EMIT or older versions of DECODES will tell you that the hard part is getting the scripts right.

At the very top of the screen you see the name of the configuration that this script belongs to, the name of the script, and a selection for data order.

The name of the script must be unique within the configuration. You are strongly encouraged to follow these conventions:

- Script Name 'ST' for Self Timed GOES DCP Messages,
- Script Name 'RD' for Random GOES DCP Messages,
- Script Name 'EDL' for Electronic Data Logger Files.

Below the top line, there are four main areas of this dialog:

- Format Statements
- Sensor Units & Conversions
- Sample Message
- Decoded Data

**Format Statements:** Type the label in the left column and the statement in the right. Each statement must have a label. You can break up a long format statement on two lines with the same label.

The Order of Format Statements is Important! The script will always start with the first statement in the list. You can select a statement and press the Up or Down buttons to move statements around in the list. You can use the Add button to add a new statement at the end of the list. The Delete button will ask you for confirmation before deleting the selected statement.

**Sensor Units & Conversions:** In this list you assign units to each sensor and a raw conversion algorithm. In the example shown the user has selected the algorithm for Battery voltage. Linear conversion ( $y = Ax + B$ ) has been selected for both parameters. You then type the coefficients directly in the table.

**Sample Message Area:** You can load raw data and interactively try to decode it using your format statements and conversions. You can do this several ways.

To load a sample GOES DCP message from your LRGS server, do this:

- Press the 'Load' button to bring up the dialog shown in Figure 5-9.
- Select "Load from LRGS".
- Select an LRGS server from the list.
- Enter the DCP address, and optionally, the channel number.
- Click OK.

To load a sample message of any type from a file, do this:

- Press the 'Load' button to bring up the dialog shown in Figure 5-9.
- Select "Load from File".
- Enter the file name, or press the 'Select' button to navigate to it from a menu.
- Click OK.

You can also copy/paste from other applications directly into the Sample Message area. Use the standard copy/paste commands (CTRL-C=copy, CTRL-V=paste).

Press the 'Decode' button to apply the format statements to the raw data. The results are shown in the Decoded Data area.

The syntax of format statements is described in section 6, *The DECODES Format Language*.

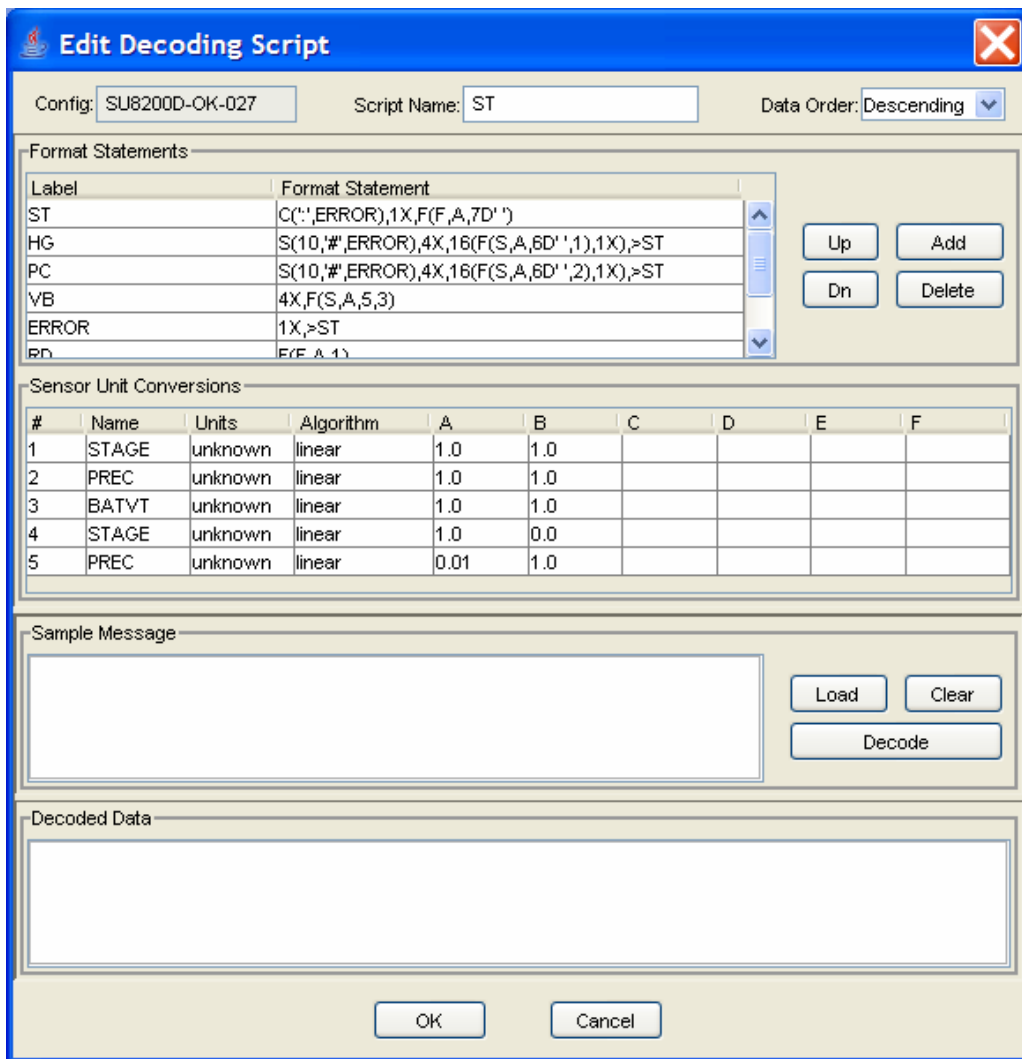
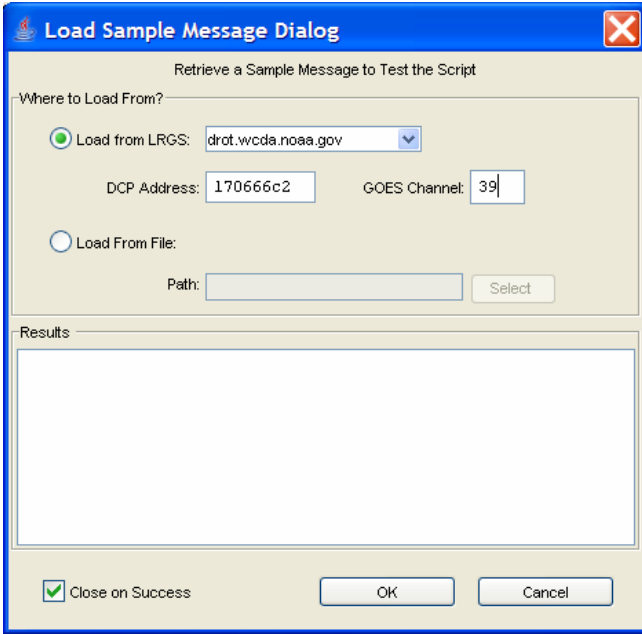


Figure 5-8: Decoding Script Edit Dialog Showing Interactive Decoding.



**Figure 5-9: Load Sample Message Dialog.**



## 5.5 The Equipment-Model Edit Panel

Figure 5-10 shows an example of this dialog. Most of the information here is descriptive in nature and not used by downstream DECODES modules.

An exception to this is the “DataOrder” property. If you place this property into an Equipment Model record with the value D (for Descending) or A (for Ascending), then the decoder will apply this to data from platforms using a configuration assigned to this equipment model.

Again the association goes like this: Platform → Config → Equipment Model

The screenshot shows the 'Decodes Database Editor' window with the 'Equipment' tab selected. The 'List' dropdown shows 'U82B'. The form fields are: Name: U82B, Type: DCP (dropdown), Company: Sutron Corporation, Model: 8200. A large empty text area is labeled 'Description'. Below is a 'Properties' section with a table:

Name	Value
PlatformType	DCP
RetrievalMethod	DCP
TimeOrder	D

Buttons for 'Add', 'Edit', and 'Delete' are to the right of the table. At the bottom are 'Commit', 'Close', and 'Help' buttons.

Figure 5-10: Equipment Model Edit Dialog.

## 5.6 The Presentation Group Edit Panel

An example of this panel is shown in Figure 5-11.

A Presentation Group determines how data will be formatted for output. This includes:

- What engineering units will be used on output.
- Numeric rounding rules to apply to each sample value

Look at the example. The first line in the Presentation Elements table says to display HG (stage) values in units of ‘ft’, or feet. The second line says that PC (precipitation) values are to be displayed in inches.

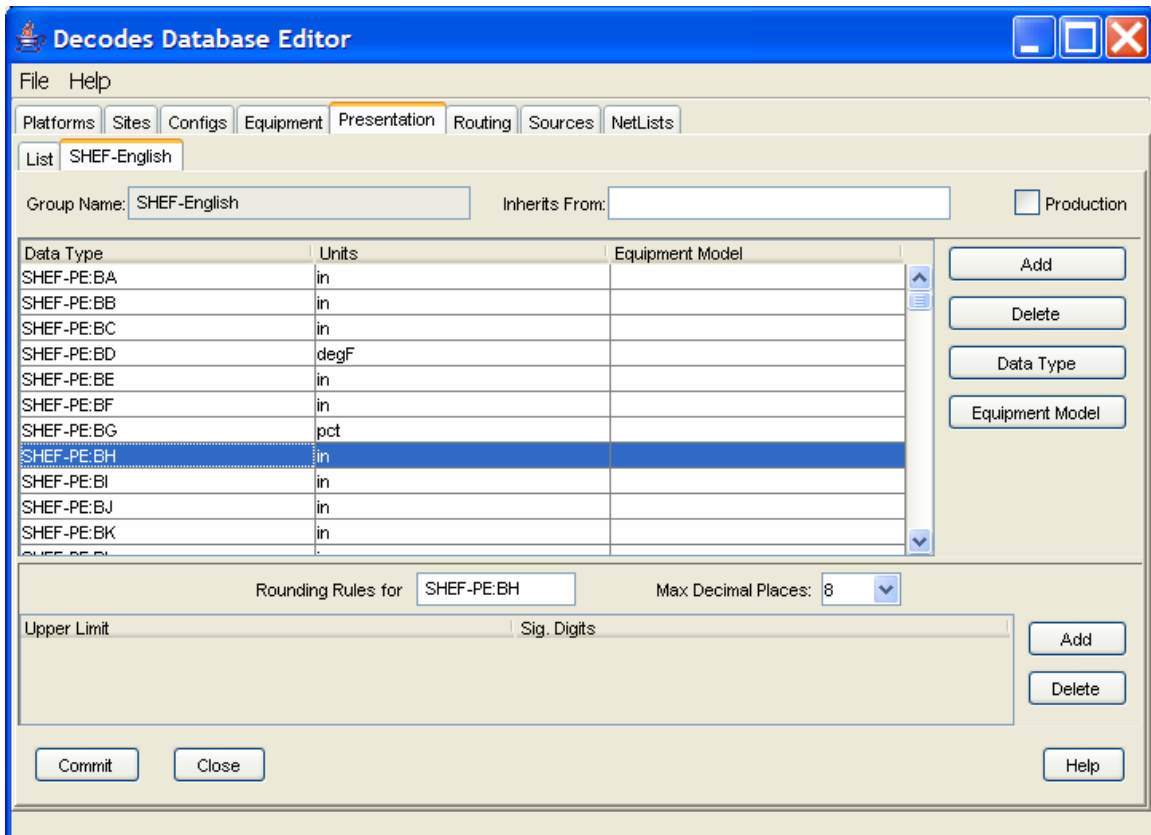
If you leave the Units field blank, then the value will be output in whatever values are decoded by the script. In other words, no conversion on output will be done.

The third line has a qualifier: It says that HR (reservoir height) values recorded on a Campbell Scientific CR-10B recorder should be displayed in inches. Hence you can get very fined-grained control over display settings.

Finally, notice the last line in this table is for data type “SHEF-PE:\*”. The ‘\*’ means any data type that is not explicitly listed elsewhere.

Notice that in the example the first line of the Presentation Elements table is selected (i.e. highlighted). When you select a presentation element, the Rounding Rules table at the bottom shows the rules to apply to those parameters.

For each rounding rule you specify a maximum value, significant digits, and maximum number of (fractional) decimal digits. Hence, the resolution can change over the possible ranges of values.



**Figure 5-11: Presentation Group Edit Panel.**

### 5.6.1 Using a Presentation Group as a Sensor Filter

A presentation group can be used to omit specified data types from your routing spec output. Suppose you want to run a routing spec with no battery voltage output. You can create a presentation group for this purpose as follows:

- Create a new Presentation Group called “SensorFilter”.
- In the “Inherits From” field, type in SHEF-English.
- Click the “Add” button. For data type, specify SHEF-PE with a value of “VB”.
- In the Units field, type “omit”.

Now, open your routing spec and select SensorFilter for presentation group.

## 5.7 The Data Source Edit Panel

Figure 5-12 shows a data source that pulls data from the DROT machine operated by NESDIS at Wallops, VA. Note the properties that are appropriate for LRGS data sources:

- host: host name or IP address of the LRGS or DRS
- username
- port

Note that in this figure the Group Members list is disabled. LRGS data sources are not groups.

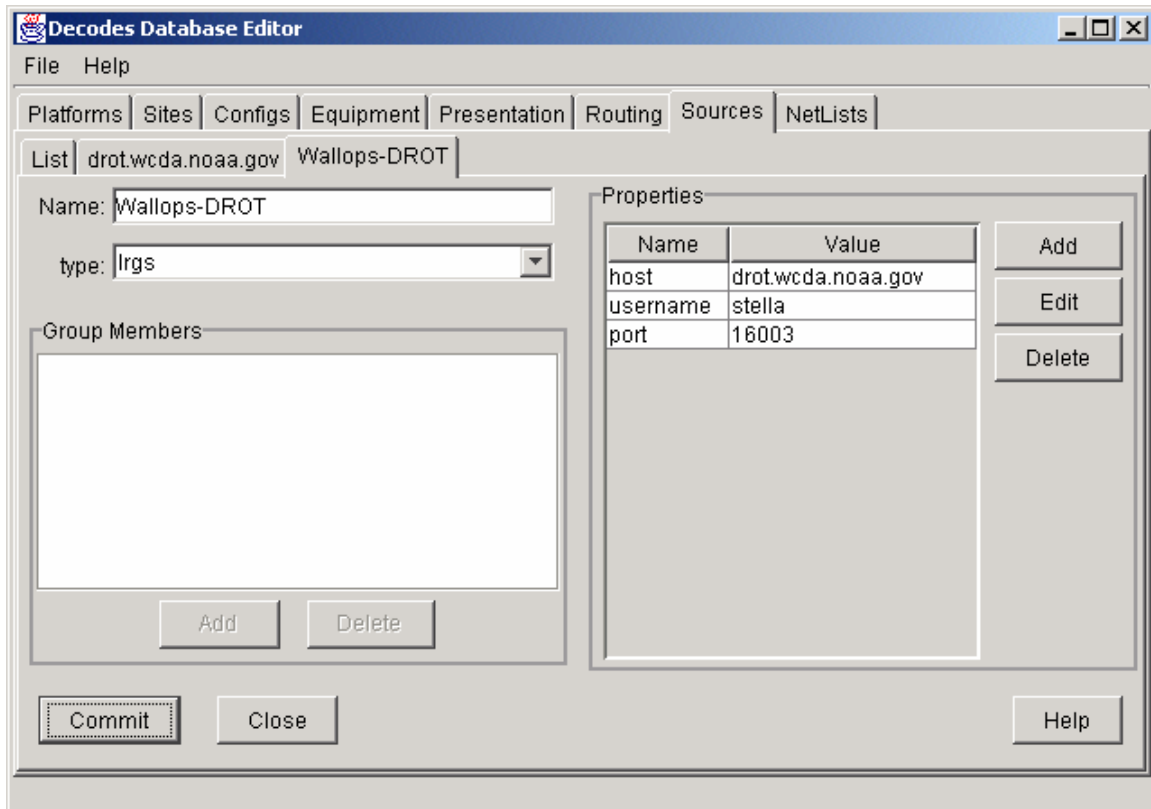
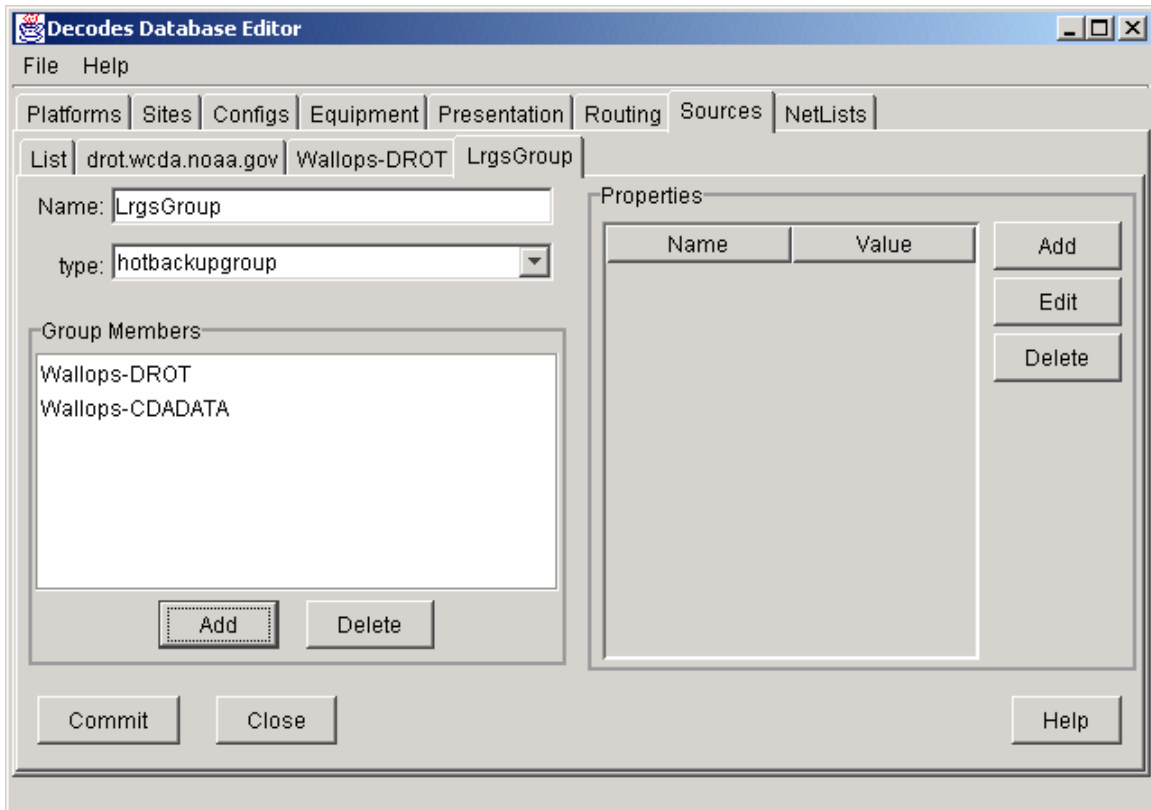


Figure 5-12: Data Source Edit Panel showing LRGS Data Source.

Figure 5-13 shows an edit panel for a hot-backup group. In the example shown, the source will try to pull data first from Wallops-DROT. If unsuccessful, or if it fails in mid-stream, it will automatically switch to another member of the group.

When making a connection, group members are always tried in the order they are specified in the list.



**Figure 5-13: Data Source Edit Panel showing Hot Backup Group.**

## 5.8 The Network List Edit Panel

Figure 5-14 shows the StPaul Network List being edited.

A network list is a collection of identifiers for a particular transport medium type. If the transport medium type is “GOES”, then the TransportID is a DCP address (as shown). Currently this is the only type of network list in use.

You can add or remove sites from the list using the buttons to the right of the list.

You can click in the headers of the list to cause the list to be sorted by Transport ID, Site Name, or Description.

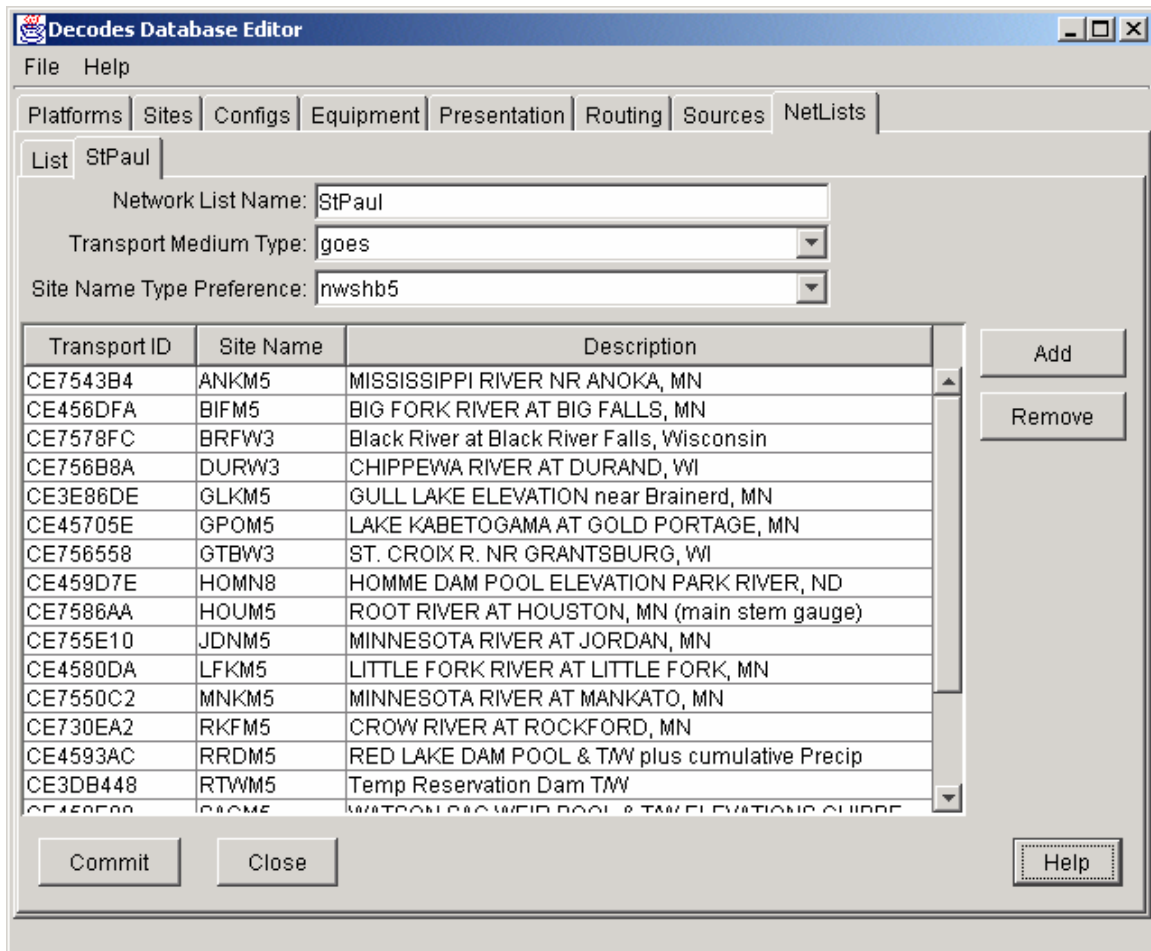


Figure 5-14: Network List Edit Panel.

## 5.9 The Routing Specification Edit Panel

We saved Routing Specification until last because they tie together all of the other entity types. Figure 5-15 shows a sample routing specification being edited.

The semantics of each field are covered at length in section 6. For now the example shows the following:

- A Routing Spec called “Atlanta-lrgs-input”
- It will read data from the data source called “LrgsGroup” that we saw in Figure 5-13.
- It will send data (consumer) to a pipe to the standard output (stdout).
- It will format data compatible with the EMIT ASCII output format.
- Sample times will be converted to EST
- Data will be presented according to the “local-presentation” group that we saw in Figure 5-11.
- Every time the routing-spec is run, data will be pulled from a time range of “now – 1 day” until “now”.
- There is a property defined called “OldChannelRanges” set to true. This causes the old rule to be in effect that GOES channels less than 100 are self-timed and over 100 are random.
- Only data from platforms referenced in the “Atlanta” Network List will be processed.

The screenshot shows the 'Decodes Database Editor' window with the 'Routing' tab selected. The 'Atlanta-lrgs-input' routing specification is being edited. The 'Properties' table is as follows:

Name	Value
OldChannelRanges	true

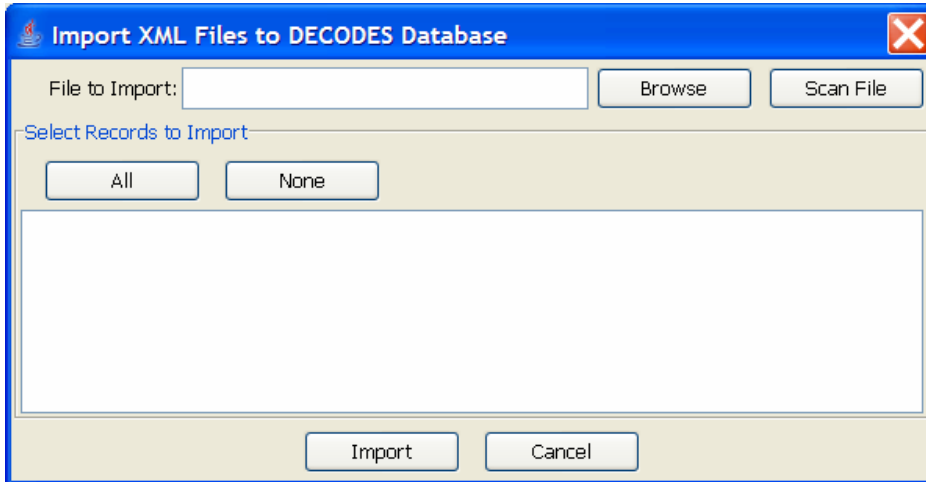
The 'Network Lists' table is as follows:

List Name	Medium Type	# Sites
Atlanta	GOES	20

Figure 5-15: Routing Specification Edit Panel.

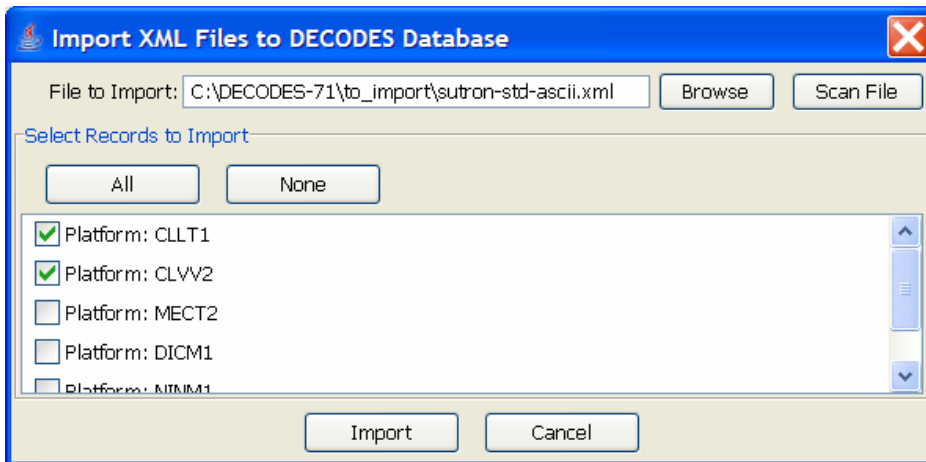
## 5.10 Import XML Files

Select File – Import. You will see the dialog shown in Figure 5-16. This dialog will allow you to open an XML file, scan its records, select which records to import, and finally, import them into your editable database.



**Figure 5-16: Import XML Files Dialog.**

Press the ‘Browse’ button or type in the file name in the area provided. After selecting a file, press the ‘Scan File’ button. Figure 5-17 shows the dialog after a file has been opened and scanned. The user has selected two platforms for import.



**Figure 5-17: Import Dialog showing successful scan.**

The ‘All’ button will select all records. The ‘None’ button will de-select all records. Once you have the desired records selected, press the ‘Import’ button at the bottom.

The imported records will be opened in editor tabs. You should then go to each tab and verify the information. Finally press the ‘Commit’ and ‘Close’ buttons on each tab.

If you are sure that all information is correct in all tabs, you can select ‘File – Commit All’ and ‘File – Close All’ as a short cut.



## 5.11 Export Records to an XML File

Press 'File – Export' to view the dialog shown in Figure 5-18. You have four choices as to what to export:

- Entire Database – Create an XML file containing all of your database records. This is the equivalent of the 'dbexport' command.
- All Platforms – Create an XML file containing all of your platform records. This is equivalent to the 'pxport -a' command.
- Platforms in network list - Create an XML file containing platform records for platforms in the specified network list. This is equivalent to the 'pxport -n' command.
- Platforms by Name – Create an XML file containing named-platform records only.

After selecting what to export, specify an output file by either typing the name in the area provided or by pressing the 'Choose' button. Finally press the 'Export' button. As data is exported, a message will be added to the 'Results' area at the bottom.

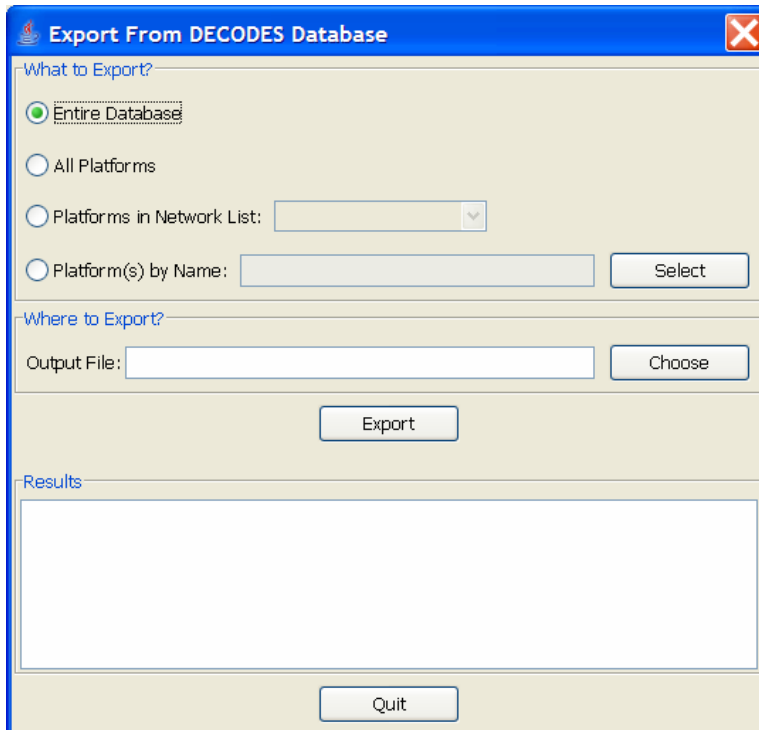


Figure 5-18: The Export Dialog.

## 5.12 Install Records to Your Production Database

The latest release of DECODES contains features for making the maintenance of a 'production' database easier.

If you *do not* want to use a production database, edit your "decodes.properties" file and comment-out the settings for "DatabaseType" and "DatabaseLocation" by placing a pound sign at the start of the line:

```
#DatabaseType=XML  
#DatabaseLocation=C:/DECODES/installed-db
```

This tells DECODES that you do not plan to use a production database and it will disable the features discussed in this chapter.

### 5.12.1 Automatic Installation when You Change Records

If you modify any records that are marked 'production', when you hit the 'Commit' button, you will be shown the auto-install dialog (See Figure 5-19). In this particular instance, we modified a configuration that was used by three different platforms, each of which was marked 'Production'. You may select which platform to install and press the 'Install' button to copy them to the production database.

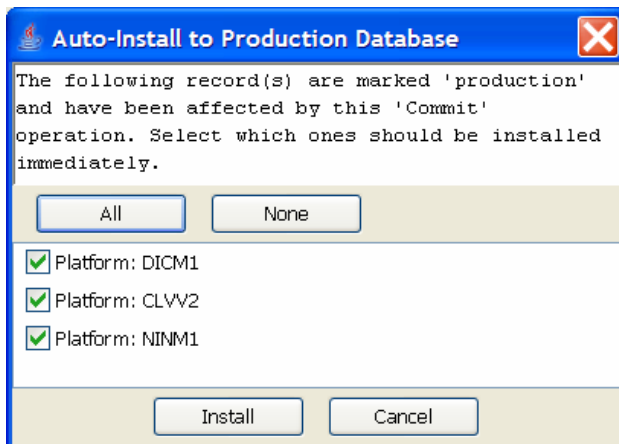


Figure 5-19: The Auto-Install Dialog.

### 5.12.2 Manual Installation to the Production Database.

Press 'File – Install' to display the dialog shown in Figure 5-20. Here you can select which records to install in a variety of ways and then press the 'Install' button.

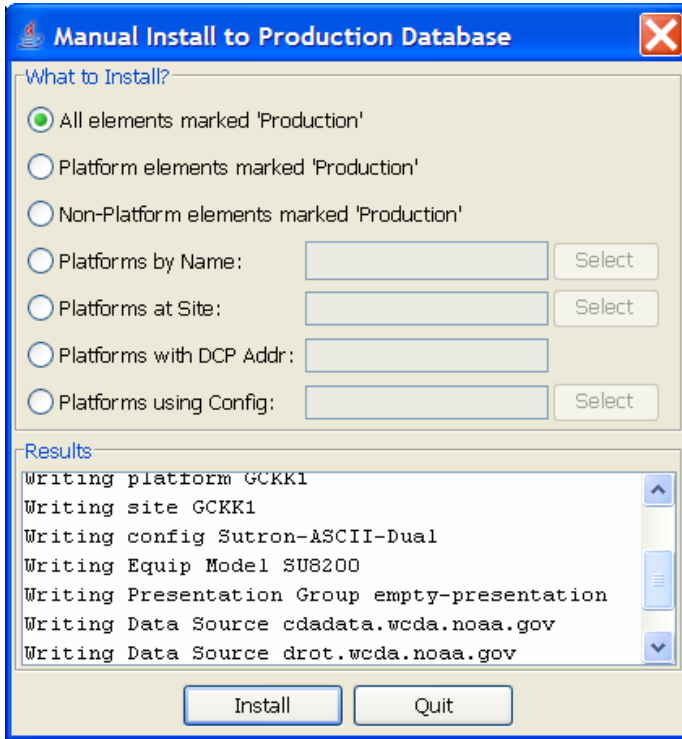


Figure 5-20: Manual Install Dialog.

## 6. The DECODES Format Language

DECODES uses Fortran-like format statements to interpret data received from a recording device. A *Decoding Script* is made up of one or more format statements. These format statements consist of two parts:

1. a *label* to identify the format, and
2. a *statement* containing a sequence of format operations.

Within a statement, the format operations are separated from each other by commas. You enter format statements within the Decoding Script Edit Dialog, described in section 5.4.1.

### 6.1 Execution of Format Statements by a Routing Spec

This is what happens when a routing spec decodes a message:

1. Use the DCP Address and channel number within the message to find a matching transport medium.
2. Get the platform record associated with that transport medium. The platform record is associated with a platform-config record, which in turn contains sensor records and one or more Decoding Scripts.
3. Retrieve the Decoding Script associated with this transport medium. For example, the message came in on channel 31, so use the 'ST' (self-timed) script.
4. Parse the script into a hierarchy of executable operations.
5. Starting with the first format statement in the script, execute the operations against the message data.

Step 4 (parsing the script) is only done once. If a second message is received for the same platform, the already-prepared scripts are reused.

Step 5 (executing the script against the message data) is the subject of this chapter.

### 6.2 Stepping through the Script and the Data

As it is executing, the script keeps track of three things:

- The currently executing format statement
- The current operation within the format statement
- The current position within the message data

The message header (e.g. 37-byte DOMSAT header) is not processed by the script. The data pointer is initialized to the first actual message byte.

The script starts with the first format statement, so position is important. This differs from previous versions of DECODES and EMIT.

Each format statement has a label. Several operations can cause decoding to jump to a new statement, identified by its label. Labels may only contain letters and digits.

Older versions of DECODES and EMIT had fixed rules about the labels for self-timed and random messages. Self-timed formats started with the label 'ST', and random message formats started with the label 'RD'. This is no longer required, but it is a useful convention to continue.

Adjacent format statements with the *exact* same label are joined into a single long statement before parsing and executing.

The various operations in the format statements step through the message data from beginning to end. There are operations for skipping characters and lines, and for positioning the data pointer within the message data.

### **6.3 Format Operation Overview**

A quick reference of DECODES format operations is presented in Table 6-1. The subsections that follow provide more detail on each one.

Several of the operators are identified by a letter. The parser is not case-sensitive, so 'X' and 'x' can both be used for skipping characters.

<b>Format Command</b>	<b>Description</b>	<b>Examples</b>
<i>nX</i>	Skip <i>n</i> data characters	2X - skip 2 characters (bytes).
<i>nP</i>	Position to the <i>n</i> th character in the current line.	2P - Position to 2 <sup>nd</sup> character in current line.
<i>n/</i>	Skip <i>n</i> data lines	3/ - skip 3 lines
<i>n\</i>	Skip backward <i>n</i> data lines.	
<i>&gt;label</i>	Jump to the format with the specified label	>ST3 -switch to format with label ST3
<i>n(operations...)</i>	Repeat operations enclosed in parenthesis <i>n</i> times	10(F(S,A,6,1)) – repeat “F(s,A,6,1)” 10 times.
<i>C(nN, label)</i>	Check the next ' <i>n</i> ' characters for number characters (digits, decimal point or sign). If all are number-characters, continue to the next format operation. If at least one is not, switch to format with specified label. Do not change the current data pointer.	C(3N, ERROR) - checks the next three characters for digits, decimal point, or sign. If at least one of the three is not, switch to format ERROR
<i>C(S, label)</i>	Check the next character for a sign ('+' or '-'). If it is a sign, continue to the next operation within this format statement; otherwise, switch to the format with specified label. Do not change the current data pointer.	C(S, ERROR) - checks the next character for a sign, switch to format ERROR
<i>C('str', label)</i>	Compare the string of characters 'str' with the next length-of-string characters in the device data. If there is a match, continue to the next operation in the current format. Otherwise, switch to the format with the specified label. Do not change the current data pointer.	C('001',NXT) - checks the next three characters for a match with '001'. If there is no match, change to format labeled NXT.
<i>S(n, N, label)</i> <i>S(n, S, label)</i> <i>S(n, A, label)</i> <i>S(n, 'str', label)</i>	<p>The second argument defines what to scan for:</p> <p>N = scan for any number character (digits or sign)  S = scan for any sign character ('+' or '-')  A = scan for any alphabetic character  'str' = Scan for specified string</p> <p>Starting at the current byte, scan at most <i>n</i> data bytes until either the target of the scan is found or an end-of-line (LF) is found.</p> <p>If the target of the scan is found, continue with the next operation in the current format. Otherwise switch to the format statement with the specified label. After the operation is completed the current data pointer points to where the scan halted, i.e. if target character(s) is found, it points to that character. Otherwise, it is moved '<i>n</i>' characters from the previous position.</p> <p>A special case of the S operation results when <i>n</i> is 0. In this case the current data pointer remains unchanged. If the target of the scan is found, continue with the next operation. Otherwise switch to specified format. This feature allows multiple tests on the same data character.</p>	<p>S(6,N,ERROR) - scan at most the next 6 characters searching for a number or a sign; and if found, set the data pointer to the matching character and continue to the next format operation; if not found, set the data pointer plus 6 and change to the format with the label ERROR</p> <p>S(0,'A',NXT) - check the current data character to see if it matches 'A'; if it does, continue to the next format operation; if not found, change to format with format label NXT; in either case the data pointer is not changed.</p> <p>S(10,'01+',ERROR)- scan the next 10 characters for the string '01+'. If not found, change to format with label ERROR.</p>
<i>nF(FT, DT, L, S, E)</i>	Field Descriptions.	Many varieties.
W	Skip any number of white space characters, including space, tab, carriage return, and linefeed.	

**Table 6-1: Format Operations at a Glance.**

## 6.4 *Skipping and Positioning Operations*

To skip a single character:

```
x
```

To skip a specified number of characters, place a number before the 'X':

```
5x
```

To skip to the end of the current line and continue processing data at the beginning of the next line, use a forward slash:

```
/
```

To skip to the end of more than one line, place a number before the slash:

```
2/
```

To position the data-pointer to a particular character position on the line, put a number followed by the letter 'p'. The following positions the pointer to the 5<sup>th</sup> character of the line. Note: byte position 1 is the start of the line.

```
5p
```

To skip backward a number of lines, use a backslash preceded by a number.

```
2\
```

## 6.5 *The Check Operation*

Check commands are used to check the current location in the data for a specified condition. If the condition is true, the data pointer is not altered. If the condition is false, you specify an alternate format statement to jump to.

To check to make sure the next *n* characters are numbers (digits), and jump to the statement labeled 'NAN' if any are not, do the following. Note that if the check is true, we proceed with the next operation, which assigns the numbers to a sensor value.

```
c(5N, NAN), f(s,a,5,1)
```

To check if the next character is a sign (either '+' or '-'), and jump to the statement NOSIGN if not:

```
c(S, NOSIGN), ...
```

To check to see if the data matches the string 'AA' and skip to the format labeled 'BB' if it does not:

```
c('AA', BB), ...
```

In this usage of the check command, the string must match exactly. The check is case sensitive and the entire string must match the current data position. Otherwise the check is false and control jumps to the named format statement.

## 6.6 The Scan Operation

Scan commands are used to scan forward from the current location in the data until a specified condition has occurred. These commands are used to position to a particular location based upon a specified condition.

Scan operations have the following syntax:

```
S(n, condition, label)
```

...where *n* is the number of characters to scan (or to the end of the current line), *condition* specifies what we are scanning for (see below), and *label* specifies the format that we jump to if the condition is not met.

The *condition* can be one of the following:

N	Scan for any digit
S	Scan for any letter, either upper or lower case
X <i>nn</i>	Scan for a character with the hex value <i>nn</i>
' <i>str</i> '	Scan for the exact string ' <i>str</i> '

If the condition is true (i.e. the requested pattern was found), processing continues to the next operation in the current format statement. The data pointer is left at the first character that matched the scan. For strings, the data pointer is left at the first character of the string.

## 6.7 The Jump Operation

The Jump operation causes an unconditional jump to a specified format statement. The data pointer remains unaffected. The jump operation has the following syntax:

```
>label
```

### 6.7.1 The Repeat Operation

Any group of operations can be performed repeatedly. Operations enclosed in parentheses and preceded by a number will be performed the specified number of times. For example,

```
8(x,F(S,B,3,1))
```

causes the operations within the parentheses (the **x** operation and the **F** operation) to be performed 8 times.



## 6.8 Field Operations

Field operations are used to extract time and sensor values from the message. The general form of a field description is:

*nF(ft, dt, length ,sensor # or fld-ID, E)*

where:

- *n* is a repetition factor
- *ft* defines the type of field
- *dt* defines the type of data
- *length* defines the field length with optional delimiters.
- *sensor #* the sensor number associated with this sensor-value field
- *fld-id* is used with DATE and TIME fields to specify different representations
- *E* is used with TIME fields to indicate that the recording of time should be viewed as an event

The field type can be one of the following:

D	Date Field (see 6.8.2 for Date and Date Components)
D+	Date of period just now ending
YR	Year
MN	Month
DY	Day of Month
JDY	Julian Day of Year
JDY+	Julian Day of Year just now ending
T	Time Field (see 6.8.3 for Time and Time Components)
HR	Hour
MIN	Minutes
SEC	Seconds
A	AM or PM
TI	Time Interval Field (see 6.8.4)
MINT	Minute Interval Field (see 6.8.4)
F	Format Label Field (see 0)
S	Sensor Value Field (see6.8.7)

The data type can be one of the following:

A	ASCII
B	Binary (unsigned)
I	Integer (signed binary)
L	Labarge pseudo-ASCII
X	Hexadecimal
S	String
BC	Campbell Scientific Binary Format
C	Campbell Scientific Binary Format (first byte defines sign and magnitude)
BD	Design Analysis binary Format (Integer value made negative by sign bit)
BT	Telonics Binary Format (Integer value made negative by sign bit)

### 6.8.1 Field Length and Delimiters:

*Length* can be optionally followed by the character D and a one or more delimiter characters. For example:

6D' , '

This indicates that the field has a length of 6 characters or can be delimited by a comma.

8D' , : '

This indicates that the field has a maximum length of 8 characters and can be delimited either by a space, comma, or colon.

The delimiter can be simply asserted, enclosed in single quotes, or represented as *xnn* where *nn* is the hexadecimal representation. If the delimiter is a comma, it must be enclosed in single quotes.

For example:

- |              |  |
|--------------|--|
| <b>6D','</b> | The field is delimited by a comma  |
| <b>6Dx1E</b> | The field is delimited by a period (the hexadecimal representation of a period is 1E). |
| <b>6D' '</b> | The field is delimited by a space  |
| <b>6DS</b>   | The field is delimited by a sign (+ or -) character.                                   |

If the character after the 'D' is 'S', it means that the data is delimited by a sign (+ or -).

Care must be taken in positioning your data pointer after a delimited field. The pointer will be left *at* the delimiter. Hence you will probably want to use a skip operation to skip the delimiter after parsing the field.

If the delimiter is not found, the pointer is advanced by *length* characters.

## 6.8.2 Date Fields

Date field descriptions have a field type of 'D'. Date fields are used in EDL files to extract time from the message data. The times are then subsequently used to time-tag data samples.

The form of a date field description is

**F(D, data type, length<Dc>, fld-id)**

The 'fld id' parameter is used to define four different date formats. Possible formats are as follows:

**F(D, type, length<Dc>,1)**

Fld-id 1 indicates the date is basically in the format year, month, day. The format differs slightly for different field lengths. For length 8, fields have the format YY/MM/DD, YY-MM-DD, and YY MM DD; for length 6, fields have the format YYMMDD.

**F(D, type, length<Dc>,2)**

Fld-id 2 indicates a Julian day is used. For length 8, fields have the format YYYY-DDD, YYYY/DDD; for length 7, YYYYDDD; for length 6, YY-DDD, YY/DDD; for length 5, YYDDD; for length 3, DDD; for length 2, DD. For cases where the year is not in the date field, the year will default to the current year unless the user specifies a year during the data conversion process. If the user lets the year default and a Julian day is found that exceeds the current Julian day, it will be assumed that the data belongs to the previous year and so the year will be decremented.)

**F(D, type, length<Dc>,3)**

Fld-id 3 indicates only the month and day are recorded. For length 5, fields with format MM/DD, MM-DD, AND MM DD; for length 4, MMDD. The same rules about the missing year apply to the field descriptions for dates with fld id of 3 as the ones for the dates with fld id of 2.

**F(D, type, length<Dc>,4)**

Fld-id 4 indicates the same type of format as fld-id 1 but in a different order-month, day, year. For length 8, fields with format MM/DD/YY, MM-DD-YY, and MM DD YY; for length 6 MMDDYY.

You can also parse the date components individually:

F(YR, type, length)	Parse a year field. Length can be 2 or 4.
F(MN, type, length)	Parse a month field. If length is 2, expect a number from 1 to 12. If length is 3, expect a 3-character month abbreviation like jan, feb, etc.
F(DY, type, length)	Parse day of month.
.	
F(JDY, type, length)	Parse julian day-of-year.
F(JDY+, type, length)	Parse julian day-of-year just ending.

### The 'Increment-Day' Feature:

The 'D' and 'JDY' field-types may optionally have a plus sign after them. This feature allows us to handle EDL data that gives complete date information only at the end of a day. For example, suppose a file started like this:

```
001 20:00 22.1 12.5
001 21:00 22.2 12.5
001 22:00 22.1 12.5
001 23:00 22.3 12.4
004 2003 335 24:00 22.2 12.5
001 01:00 22.3 12.5
...
```

Notice that the line with the label "004" contains the year and the Julian day (335) that has just ended. Data prior to this line is day 335, data after this line is day 336. Hence we want to increment the day after parsing it. So use the JDY+ operator.

### 6.8.3 Time Fields

Field descriptions for times have a field type of 'T' and a data type of 'A' (ASCII). Thus, the form of a field description for a time is

**F(T, A, length<Dc><, sensor #, E>)**

The optional '*sensor #*' and '*E*' parameters signify that the time recorded is an event. This is used for recorders that record only the time whenever an event occurs e.g. the time is recorded whenever a tipping bucket tips. In this case, the recorded time is considered to be the data. When DECODES encounters a field description for a time and it has a sensor number and the 'E' parameter, DECODES will use the value 1 as the data value associated with that time.

The raw value of 1 can be converted to the desired units via an EU conversion in the script. For example, if a tipping bucket rain gage records the time whenever .01 inches of rain falls, convert the raw value of 1 to .01 with a linear EU conversion.

For length 8, times are expected with format HH-MM-SS or HH:MM:SS; for length 6, HHMMSS; for length 5, HH:MM, HH-MM; for length 4, HHMM; for length 3, HMM; and for length 2, MM.

You can also parse the time components individually:

```
F(HR, type, length<Dc>)
F(MIN, type, length<Dc>)
F(SEC, type, length<Dc>)
```

## 6.8.4 Time Interval Fields

Time interval fields have a field type of TI and a data type of 'A' (ASCII). The time interval field describes a field that contains a new time interval for recording data. This field description is useful for recorders that can adjust the recording interval from that set in the SENSORS entity to a new one when certain conditions occur. The form of a field description for a time interval is as follows. The data field format is the same as those for the time field description.

```
F(TI, A, length<Dc>, sensorNum)
```

## 6.8.5 Minute Interval and Offset Fields

Minute interval fields have a field type of MINT. The data type can be ASCII or any of the binary types. It is useful for parsing data where the time interval is given in a number of minutes.

```
F(MINT, A, length<Dc>, sensorNum)
```

Many GOES DCP messages contain a minute offset to the first sample. You can process these with field type 'MOFF':

```
F(MOFF, A, length<Dc>)
```

The offset sets the 'current time' to the message time, minus the parsed number of minutes. It also has the effect of truncating the seconds. So if message time is 14:22:39 and the minute offset in the message is 22, then the current time is set to 14:00:00.

You can use MOFF multiple times in the same message. Each time it sets current time relative to the unchanging message time.

Negative intervals can be specified by adding a minus sign after MINT:

```
F(MINT-, A, length<Dc>, sensorNum)
```

Use the negative interval only when the data is descending (i.e. most recent samples first) AND you are time-tagging based on times or time offsets found IN THE MESSAGE (not including the GOES header).

For example, look at the following GOES DCP message:

```
4804F5C804011203139G31-5HN060W0000177:HG
31#30+3.95500e+00+3.95700e+00+3.95700e+00+3.95700e+00+3.95700e+00+3.957
00e+00:HG 196#180+3.94900e+00:HG 206#180+3.96100e+00:VB
31#60+1.18576e+01+1.18620e+01+1.18509e+01:ZL$
```

After the initial :HG, we have 31 (minute offset to first sample) followed by #30 (minute interval of sample values – negative), followed by 6 sample values in exponential notation. Ignore the remainder of the message starting with the second ':HG'.

We can process this message with the following format statement:

```
4x,f(moff,a,3d'#'),x,f(mint-,a,2,1),6(f(s,a,12,1))
```

### 6.8.6 Format Label Fields

Format-label fields describe a data field that contains a code that is to be used as a format label to select a new format. DECODES extracts a label from the message data and jumps to a matching format statement.

The data pointer will remain at the character immediately following the extracted format-label.

Format-label fields allow DECODES to switch formats based upon a code found in the device data. For example, if a device records the data in different formats and also records a code that identifies the each format, a statement can be written for each code, using the code itself as a format label.

If DECODES cannot find a match for the label extracted from the data, it will attempt to switch to a format statement with the label 'ERROR'. If none exists, decoding of this message will be aborted.

The format of a field description for format labels is

**F(F, A, length<Dc>)**

Examples:

**F(F, A, 4)** - Format label field is 4 characters long.

**F(F, A, 8D',')** - Format label field is delimited by a comma and has at most 8 characters.

## 6.8.7 Sensor Value Fields

Sensor field descriptions have a field type of 'S'. They are used to extract data samples from the message. The format of a sensor field description is

**nF(S, data type, length<Dc>, sensor #)**

“Data type” can be any valid type listed above in section Field Operations

Examples:

**F(S, A, 6, 1)**      The Data will contain one 6-character ASCII sample for sensor number 1.

**F(S, A, 5D',', 2)**      The data is delimited by a comma and has at most 4 ASCII characters; the value was produced by sensor 2.

**3F(S, B, 3, 1)**      3 signed-binary samples for sensor number 1. Each sample is 3 characters long.

## 6.9 Special Decoding Features

### 6.9.1 How to Omit Specific Sensor Values

To omit all sensor values of a given data type, create a presentation group. Set the engineering units for the data time to the string “omit”.

To omit specific sensors from a configuration, add a config-sensor property called “omit” with a value of “true”.

To omit specific sensors from a specific platform, add a platform-sensor property called “omit” with a value of “true”.

### 6.9.2 Data Delimited by either a Plus or Minus Sign

Some platforms send data in a string of values that are delimited only by a sign. For example:

HG: +13.2+10.1+8.4+5.1+2.5+0.1-1.5-4.2

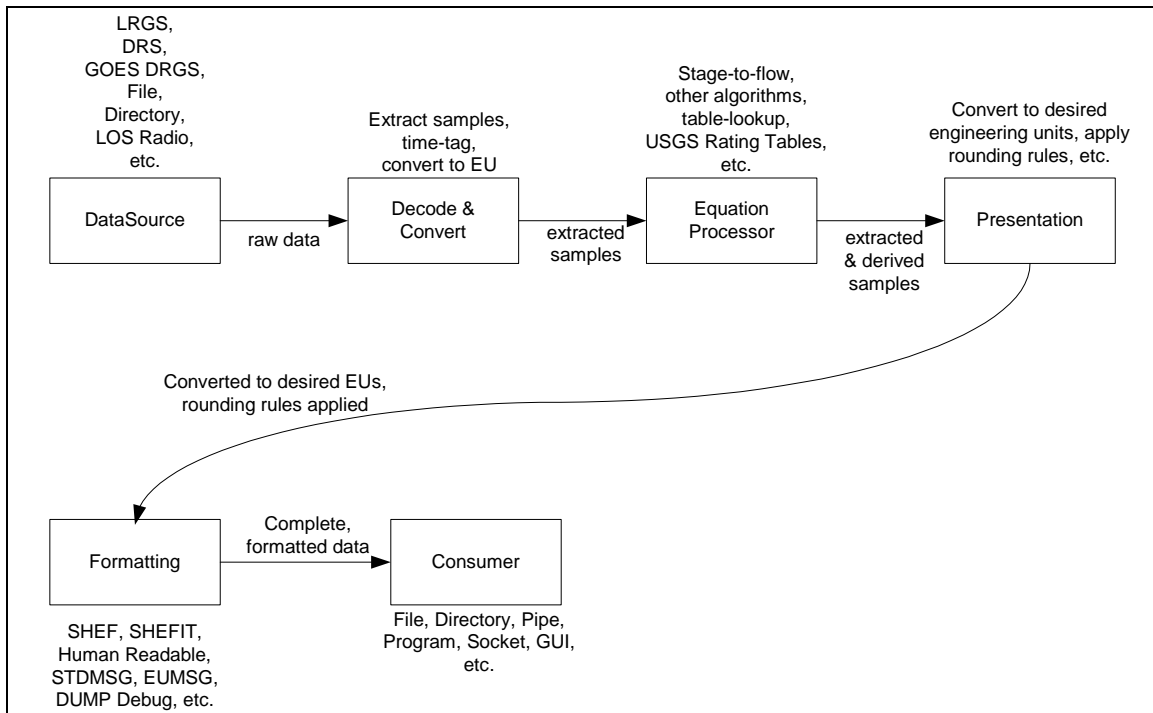
Notice that the length of the sample changes with the magnitude of the number. Above 10, we have 5 characters “+13.2”. Below 10 we have 4 characters “+8.4”. Also note that the sign can change.

To handle this, if the delimiter in your field operator is a sign (either ‘+’ or ‘-’), then either sign will work as a delimiter when parsing the message. Thus, we could parse the above line as follows:

4x, 8( F(S,A,6d'+',1) )

## 7. DECODES Routing Specifications

Figure 7-1 shows the data flow for a routing specification. Take a moment to study the components involved. This section will discuss how to run a routing specification and how to control each of the components shown in the figure.



**Figure 7-1: Data Flow for Routing Specifications.**



## 7.1 How to Run a Routing Specification

### Synopsis:

```
rs <options> spec-name
```

### Options:

-e	Run from the editable database (default is installed database)
-s <i>script</i>	Script name to be executed. This option may appear multiple times. This can also be accomplished with the “scriptname” property.
-m	Do NOT apply sensor min/max limits (default is to do so).
-n <i>netlist</i>	Add the named network list to the routing spec before executing it.
-S <i>since</i>	Override “since-time” specified in database routing spec record.
-U <i>until</i>	Override “until-time” specified in database routing spec record.
-o <i>filename</i>	Set the status monitor output properties file. See below.
-R	Remove redundant DCP data from output.
-E <i>DatabaseLoc</i>	Specify an Explicit XML database location. This allows you to run a routing spec in a database <i>other</i> than your editable or installed database.
-c	Enable computations (e.g. USGS RDB File Rating).
-C <i>CompConfigFile</i>	Specifies computation configuration file (default is \$DECODES_INSTALL_DIR/computations.conf)
-k <i>lockFile</i>	Use specified lock file to ensure only one instance runs and to provide a mechanism to kill the routing spec (by removing the lock file).
-p <i>name=value</i>	Adds (or overrides) a routing-spec property.
-L <i>connectSpec</i>	Specify LRGS data source on command line, overriding data source specified in database routing spec definition. The ‘connectSpec’ is in the form <i>host:port:user[:password]</i>

### Description:

This script starts a Java Virtual Machine running the specified routing spec. All of the parameters that control the action of the routing spec are specified in the database or the DECODES properties file. Hence there are no options to this command.

Routing Spec Properties can be used to control the execution of the spec, or to control the actions of various component objects. The properties which apply at the top level are:

- **scriptname:** This is a blank-separated list of script names to be executed. The default action is to execute any script. This property is equivalent to the -s command line argument.
- **nolimits:** Value is either true or false (default = false). This property is equivalent to the -m command line argument. It tells the spec to NOT apply sensor min/max limits, even if they are defined.

### Examples:

```
rs Atlanta-lrgs-input           Execute routing spec “Atlanta-lrgs-input”  
                                from the installed database.
```

```
rs -e test
```

*Execute routing spec “test” from the editable database.*

```
rs -e -s ST test
```

*Execute routing spec “test” from the editable database, but only process messages for ST (self-timed) scripts.*

Each routing spec writes trouble-shooting information to a separate log file. The file has the name of the routing spec with a “.log” extension. These files will be placed in the directory specified by the ‘RoutingStatusDir’ value in `decodes.properties`. If none is defined, the default of `$DECODES_INSTALL_DIR/routstat` will be used.

Thus look for the log file for routing spec ‘test’ in the file:

```
$DECODES_INSTALL_DIR/routstat/test.log.
```

### **7.1.1 Routing Spec Properties**

In the database editor you can enter properties that affect various aspects of a routing spec. Currently, none of these properties are used by the routing spec itself. Rather they are passed to the components of the routing spec like Data Source, Consumer, Output Formatter, etc. The properties used by components are described in the sub-sections that follow.

### **7.1.2 Adding Network List Names from the Command Line**

Use the `-n` argument to add additional network lists to the routing spec from the command line. This makes the routing spec more flexible. You could define a routing spec in the database with no network lists and supply them on the command line to process different data sets.

### **7.1.3 Overriding Time Range from the Command Line**

The `-S` and `-U` arguments (note, must be capital letters) can be used to override the time range specified in the database. For example, the following runs ‘myspec’ but the since time is replaced by “now - 1 day”:

```
rs -e -S 'now - 1 day' myspec
```

Note that the string must be enclosed in single quotes so that it is passed as a single argument. Also note that it must be separated from the `-S` by at least one space.

### 7.1.4 Status Output File

The routing spec will write its status periodically to a file. This allows you to check on the status of the specs running in the background.

By default, the output file will be called “*name.status*”, where *name* is the name of the routing spec. The file will be placed in the directory specified in the `decodes.properties` file. (Refer back to Table 3-1).

You can specify a particular file with the `-o` command line argument. For example, to have the status written to “`/tmp/mystat.status`”, use the following command line argument:

```
rs -o /tmp/mystat.status ... (other args here) ...
```

If you do not want the spec to write status, include the argument with a value of “-”. As follows:

```
rs -o - ... (other args here) ...
```

### 7.1.5 Optional Lock File

The `-k` argument allows you to specify a lock file for this instance of the routing spec. Lock files do two things:

1. Ensure only one instance with a given lock file can run: If the lock is busy, the routing spec will fail to start.
2. Provide an easy way to terminate a background routing spec: Simply delete the lock file.

While running, the process will ‘touch’ the lock file every 10 seconds. If the file was deleted, the process will terminate. So allow about 10 seconds after deleting a lock file before starting a new instance.

A lock file is “busy” if it exists and has been touched within the last 20 seconds.

## 7.2 Time Tagging Data Samples

The “DataOrder” property can be set to ‘A’ (for Ascending) or ‘D’ for Descending.

- Ascending means that the oldest samples are first in the message. Successive samples for the same sensor have an ascending time.
- Descending means that the newest samples are first in the message. Successive samples for the same sensor have a descending time.

The DataOrder property can appear in several entities. This is how DECODES determines the order for a given sensor:

- If there is a “DataOrder” property in the Equipment Model associated with the Platform Config. Set this as the default for all sensors. In most cases, this is all that is necessary.

- If there is a “DataOrder” property in the Equipment Model associated with the transport medium, this overrides the previous value. An example for using this would be a random message that reports time in a different order than self-timed messages.
- If the DataOrder value in the decoding script is not “undefined”, this value will be used.
- If there is a “DataOrder” property in the ConfigSensor, use it as the value for that particular sensor. Use this if sensors report data in different orders.
- If there is a DataOrder property in the PlatformSensor entity, use it as the value for that particular sensor.

So, in rising order of precedence, data order (Ascending or Descending) is determined by:

1. Platform Config’s EquipmentModel ‘DataOrder’ Property
2. Transport Medium’s EquipmentModel ‘DataOrder’ Property
3. DecodingScript value
4. ConfigSensor ‘DataOrder’ property
5. PlatformSensor ‘DataOrder’ property

### 7.3 Expanding Environment Variables

Several of the properties listed in the following sections allow embedded environment variables. This is particularly true for file and directory names. The following table list the substitutions that are done:

<i>String</i>	<i>Replaced with ...</i>
~	Current user's home directory.
\$HOME	Current user's home directory.
\$DATE	Current Date/Time in default format.
\$DATE( <i>format</i> )	Current Date/Time in user specified format (see below).
\$DECODES_INSTALL_DIR	The location where DECODES was installed.
\$user.dir	The current working directory.

The Date/Time format is specified with a string passed to the Java "SimpleDateFormat" class. See Sun's documentation at the following URL for a description of format options.

<http://java.sun.com/j2se/1.5.0/docs/api/java/text/SimpleDateFormat.html>

## 8. Data Sources

The following sections describe the semantics of data sources. See section 5.7 for instructions on how to modify a data source's parameters.

### 8.1 LRGS Data Source

LRGS Data Sources are used to connect to LRGS or DRS systems over the network. The LDDS Server must be running on the LRGS you want to connect to.

Properties for the LRGS Data Source may be placed in the Data Source record or the Routing Spec Record in your DECODES database. Properties defined in the Routing Spec record will override those of the same name defined in the Data Source record.

So, for example, if the Data Source record contains “username=joe”, but the Routing Spec record contains “username=ted”, THEN “ted” will be the username passed to the LRGS server.

Accepted properties are as follows:

- **host:** The host name or IP Address of the LRGS system to connect to. (Optional, If missing, the name of the data source object is used.)
- **port:** Port number for this LRGS's server. (Optional, default = 16003)
- **username:** registered user on the LRGS server (required)
- **password:** Some LRGS servers are configured to require passwords. If this is the case, you will need to enter the password here. **Warning! The password will be stored in clear text in the SQL database and XML files.**
- **single:** (Default=false) The newer LRGS servers have a new feature whereby many DCP messages can be returned for a single request. By default, DECODES will use this feature if the server supports it. To force the old (single message per request) behavior, add a property “single” with a value of either “on”, “true”, or “yes”.
- **sendnl:** (Default=true) – Old DRS servers do not support network list transfers. Set this to false when connecting to such servers. The data source will then assume that the network lists are already loaded on the DRS. You must then transfer the list using some other mechanism (e.g. FTP) prior to running the routing spec.
- **response.timeout:** (Default=60 seconds) This is the number of seconds to wait for a response from this server. See discussion of timeouts below.
- **searchcrit:** If supplied, this should be the full path-name to a search criteria file to be passed to the server. See below on how searchcrit information is processed.

Each time an LRGS is initialized, it is passed the new search criteria from the routing specification. This information includes the “since” and “until” times, network lists, and the routing spec properties.

The Routing Spec may contain a property called “lrgs.timeout”, set to a number of seconds. If so, this value will be used by the LRGS data source. The default timeout is 60 seconds.

The routing spec will exit with the LRGS Data Source determines that the specified “until time” has been reached. If no until time is specified, the routing spec will continue running indefinitely.

### **The “searchcrit” Property**

If you supply a “searchcrit” property in either the routing spec or data source record, it will be the full path-name to a search criteria file. This allows you to use the full range of search-criteria in a routing spec.

The information in the file will be somewhat modified before being passed to the server. If the routing spec contains a “since” or “until” time, these will override the values in the search criteria.

If the searchcrit names a network list, with or without the “.nl” extension, then:

- IF the list is contained in your DECODES database, it will be sent to the server.
- ELSE IF the list is found on your hard disk in either the current directory or a subdirectory called “netlist” (i.e. in “.” or “./netlist”), then it will be sent to the server.
- ELSE, assume that the list already resides on the server.

Also, any network lists specified directly in the routing spec will be sent to the server.

### 8.1.1 Timeouts in LRGS Data Sources

There are two timeout values that effect the operation of an LRGS Data Source:

The “response.timeout” property in the LRGS Data Source object controls how long to wait for a response from the server after sending a request. The purpose of this timeout is to catch connections that have failed. For example, the server is no longer responding or a WAN link has gone done.

The “lrgs.timeout” property *in the Routing Spec object*, specifies the maximum number of seconds to wait for the next message to arrive. This means, even if a link is up and the server is responding to each request in a timely fashion, wait no more than this many seconds for the next message. The purpose of this timeout is to catch problems upstream from the server.

The “lrgs.timeout” property is associated with the routing spec (not the Data Source) because it depends on what data you are retrieving. For example, if I am getting data from a single DCP that reports hourly, I might set lrgs.timeout to 3660 (1 hour and 1 minute).

In most cases, the “response.timeout” should be fairly low. The default value of 60 seconds should suffice.

When a timeout (of either type) occurs, the LRGS Data Source throws an exception and...

- If this LRGS is part of a Hot Backup Group, the group will attempt to connect to another LRGS.
- If this LRGS is the sole data source, the routing spec will terminate.



## 8.2 File Data Source

A File Data Source reads a series of DCP messages from a single file. It processes the file from beginning to end and returns each message found therein. After reaching the end of the file, the Data Source causes the routing spec to exit.

Accepted properties for a File Data Source are as follows:

<i>Name</i>	<i>Value Type</i>	<i>Description</i>
filename	path	If present, this value will be used as the file name to be read. It can be a complete path name or a filename relative to the current working directory. If this property is absent, the name of the data source will be assumed to be a file name. The value may also contain environment variables as described in section 7.3.
before	delimiter	A special string that delimits the beginning of a new message in the file. This string may contain binary and escaped characters such as \n (newline) or \001 (ASCII STX).
after	delimiter	special string the delimits the end of a message in the file.
MediumType	name	Specifies the type of data stored in the file, such as “GOES”, or “data-logger”.
MediumId	name	Specifies the transport medium ID of the platform that generated the messages in the file. Optional: Only use this if all the messages in the file came from the same platform, such as an EDL file. Typically, the MediumId can be constructed from information in the message header so specifying a property is not necessary.
LengthAdj	number	Some header types (like Vitel) report message length wrong. Use this kludge to adjust the length before attempting to read the message bodies.
OneMessageFile	Boolean	Default=false. When set to true, DECODES assumes that the entire file contains one message.

For added flexibility, the filename property may contain environment variables preceded with a dollar sign. For example, set the filename property to **\$FILENAME**. Then start the routing spec with the -D argument defining the filename, as follows:

```
rs -e -DFILENAME=/usr/local/mydata/cr10-1.dat specname
```

### 8.2.1 Delimiting Messages Within the File

The ‘before’ and ‘after’ strings are optional. Here is how DECODES interprets them:

- If neither ‘before’ or ‘after’ is specified, the entire file is assumed to contain a single message.

- If ‘before’ is specified, but ‘after’ is not. DECODES will scan the file for the ‘before’ string and return data following it, up to, but not including the next ‘before’ string. The final message terminates at end-of-file. Any data in the file prior to the first ‘before’ string will be ignored.
- If ‘after’ is specified, but ‘before’ is not. The first message starts at the beginning of the file and continues up to, but not including, the first occurrence of the ‘after’ string. Any data at the end of the file not terminated by the ‘after’ string will be ignored.
- If both ‘before’ and ‘after’ are specified, only completely delimited messages will be processed from the file.

### 8.3 Directory Data Source

A “Directory Data Source” allows you to designate one or more directories on your system into which data files are placed. This is typically used for EDL (Electronic Data Logger) files.

You use properties to specify the directories and other settings. The routing spec will continually “watch” the directories for new files to appear. When a file is found it is decoded.

The following properties are accepted. The property name is *not* case sensitive, but in some cases (e.g. a UNIX file name) the property value *is* case sensitive.

<i>Name</i>	<i>Value Type</i>	<i>Description</i>
DirectoryName	Path	The path name to the directory to be watched. The value may contain environment variables (see below).
FileExt	String	Only files with this extension will be processed from the directory. Other files will be ignored.
Recursive	Boolean	If true, then DirectoryName is taken as the root of a hierarchy of directories. All sub-directories (and sub-sub-directories, etc.) are also watched for files.
NameIsMediumId	Boolean	Some EDL files do not have a complete medium identifier in the header. Set this to true if the file-name itself is to be taken as the medium identifier. Note: If a FileExt is specified, it is stripped from the name before using it as a medium ID.
SubdirIsMediumId	Boolean	Use this with the Recursive flag if the sub-directory name is to be taken as the medium ID.
DoneDir	Path	If specified, files that have been successfully processed will be moved to this directory. One of DoneDir or DoneExt must be specified.
DoneExt	String	If specified, files that have been successfully processed will be renamed with this extension. One of DoneDir or DoneExt must be specified.
OneMessageFile	Boolean	Default=true. If true, DECODES assumes that each file in the directory contains a single message. Turn this feature off by adding a property explicitly set to false.
MediumType	name	Specifies the type of data stored in files in this directory, such as “GOES”, or “data-logger”.

**Table 8-1: Properties for Directory Data Source.**

### Setting up a Tree of Directories for EDL Files:

To set up a tree of directories to be watched, set 'DirectoryName' to the root of the tree, and set 'Recursive' to true. If you want to devote each sub-directory to a specific platform, set 'SubdirIsMediumId' to true. Then name each subdirectory with the transport identifier in the platform.

Example: I have two data-loggers. The platform records have medium IDs of "01435532-cr10-1" and "05523352-cr10-1". The file headers do not contain the STATION identifier. The data files will all end in ".dat". After processing, I want the files renamed with the extension ".done".

I can set up a tree as follows:

- Parent Dir: \$HOME/edl-data
  - Sub Dir: 01435532-cr10-1
  - Sub Dir: 05523352-cr10-1

I set up a DirectoryDataSource with the following parameters:

DirectoryName	\$HOME/edl-data
FileExt	.dat
Recursive	true
SubdirIsMediumId	true
DoneExt	.done

I then build a routing spec that uses this data source. When I run the routing spec, it watches for new files to appear. I place the data files in the appropriate sub-directory and they are immediately processed.

### Files with Errors:

If a file contains un-recoverable errors, we don't want the routing spec to abort, as it would if we were only processing a single file. When such an error occurs, DirectoryDataSource renames the file with the extensions ".err" and leaves it in the input directory. FAILURE messages will be generated in the log explaining the nature of the problem.

### Only Process Complete Files

We only want to process files that are complete. Consider the following scenario: I am copying a large EDL file from a floppy disk into the input directory. Before the copy is complete, the Directory Data Source grabs the (partial) file and processes it. There are two way to avoid this problem:

- Specify a FileExt property like ".dat". Copy the file in from the floppy disk *without* the extension, and then rename the file *with* the extension.
- Unix Only: Copy the file to a temporary directory on the same mounted disk partition. Then use the 'mv' command to move it into the input directory.

## 8.4 Hot Backup Group Data Source

A Hot Backup Group Data Source is primarily used for a set of LRGS connections. One connection may fail, in which case we want our routing spec to try another. This makes your routing spec more reliable, particularly if this is a real-time routing spec that runs continuously (i.e. no “Until Time”).

Currently there is only one property that is used by a Hot Backup Group:

- **recheck:** (default = 900 seconds, or 15 minutes) – If the currently active data source is not the first one in the list, the Hot Backup Group will attempt to connect to higher priority data sources at this period.
- **fudge:** (default = 120 seconds, or 2 minutes) – Amount of time to back-up after connecting to new data source.

The Hot Backup Group contains an *ordered* list of LRGS data sources. The group will prefer the members in the order they are listed.

Upon start-up, the group will attempt to connect to a LRGS, starting with the first one listed. Once a successful connection is made, this LRGS becomes *active*. The group then reads DCP messages from this source until...

- The active source fails (either a timeout or broken connection), or
- The active source is not first in the list *and* the recheck period expires.

When this happens, the group will try to connect to a source, once again starting from the first in the list.

When the group changes from one active source to another, it passes the new source the network lists and search criteria with one modification: The ‘since’ time is adjusted to:

LastMessageTime – fudge

... where LastMessageTime is the time of the last DCP message I received. The ‘fudge’ factor (default=120 seconds) can be controlled via a property setting.

The purpose of this fudge factor is to account for small variations in the system clocks of the LRGS members. If you have all your systems synchronized via NTP you can make the fudge factor very small.

Larger fudge factors may result in duplicate messages: A DCP message received from one LRGS and then after a switch, the same message received from the new LRGS.

## **8.5 Round Robin Group Data Source**

NOT YET IMPLEMENTED!

A round-robin group contains a list of other data sources.

The purpose of a round-robin group is to continually read data from all data sources in the group. This differs from a hot-backup group, which only uses one data source at a time

## 8.6 Socket Stream Data Source

A socket stream data source opens a socket and reads a one-way stream of data containing raw DCP messages. Some DRGS and DOMSAT products provide such a stream.

Accepted properties for `SocketStreamDataSource` are:

- `host` = the host name or IP address of the server
- `port` = the port number of the socket to be opened
- `lengthAdj` = a negative or positive number. The default value is -1. (See below)
- `delimiter` = A string that begins each message, use `\r` for carriage return and `\n` for linefeed. The default delimiter is `\r\n`. (See below)
- `endDelimiter` = A string that marks the end of each message. This is required if header is “noaaport”. The NOAAPORT message format determines the message length not from the header but from the beginning and end delimiters.
- `header` = GOES, VITEL, NOAAPORT, Vaisala. The default is GOES (See below)

### Delimiters and Length Adjustments

Each message must start with a 37-byte DOMSAT header. The last 5 bytes of the header is the number of message bytes to follow. Immediately following the message data, a delimiter is expected. The delimiter is not included in the message length.

The Vitel DRGS reports a message length which is actually 4 more than the number of bytes actually present in the message data. Each message is terminated by a carriage return and linefeed. Hence the proper settings for a Vitel DRGS are:

```
lengthAdj = -4
delimiter = \r\n
```

The DataWise DOMSAT system reports a length that is one greater than the number actually present. It terminates each message with 3 sets of carriage-return/linefeed. The proper settings for a DataWise DOMSAT socket stream are:

```
lengthAdj = 0
delimiter = \r\n\r\n\r\n
```

### How messages are parsed

The socket is opened. The input software expects the stream to start with a message header, followed by the message data, followed by the delimiter. This cycle repeats indefinitely until the socket is closed.

The input software can get out of sync in one of the following ways:

- Detecting an invalid 37-byte header (no DCP address, channel number, or message length).
- Failing to find the delimiter string

When this happens, the input software goes into “hunt mode”. It will read characters from the socket looking for the delimiter sequence. Once found it will again attempt to read the 37 byte header.

Look at the debug-log when running the routing spec. If your ‘lengthAdj’ and ‘delimiter’ parameters are correct you will never see the messages saying that the software has skipped data. If you do see these messages:

- Consult the manual for the server system to determine how messages are formatted.
- Make sure the delimiter string is correct as described above.
- Try adjustin lengthAdj downward, into negative numbers (incrementally).

### **Network Lists and Time Ranges**

Since a socket-stream is assumed to be a real-time data source, the input software will ignore the ‘since’ and ‘until’ times specified in the routing spec.

Network lists will be used to filter incoming data. Only messages whose DCP address is contained in one of the routing-specs network lists will be processed. If the routing spec contains no network lists, all data will be processed.

### **Header Format**

The “header” property should be one of “GOES”, “VITEL”, or “NOAAPORT”. The default is “GOES” if the property is missing. The Vitel header is slightly different in that it does not include the failure-code field, causing subsequent fields to be shifted one character to the left.

## 8.6.1 Using SocketStreamDataSource for NOAAPORT

NOAAPORT messages are received over a socket in the following format:

```
[SOH]\r\r\nNNN\r\r\nHHH[RS]DDD\r\r\n[ETX]
```

...where

- *[SOH]* is an ASCII Start-Of-Header character (octal \001)
- *NNN* is a NOAAPORT 3 digit sequence number
- *HHH* is a NOAAPORT Header (ignored)
- *[RS]* is an ASCII Record-Separator character (octal \036)
- *DDD* is the DCP message containing time stamp and other header fields before and after the message proper.
- *[ETX]* is an ASCII End-of-Text character (octal \003)

The *DDD* data field contains all the header fields and message-data that we need. We want to ignore everything else. Consequently use the following Data Source Properties:

- host
- port =
- delimiter = \036
- endDelimiter = \r\r\n\003
- header = NOAAPORT

The Socket Stream will then process only the *DDD* (data) field between the *[RS]* and *\r\r\n[ETX]*, and ignore everything else.

The Data Field itself will have the following format:

```
AAAAAAAA DDDHHMMSS ddd... SSFFNN CCCs
```

...where

- *AAAAAAAA* is the 8-hex-char DCP Address
- *DDHHMMSS* is the date/time stamp.
- *ddd...* is the actual message data
- *SS* is the signal strength
- *FF* is the Frequency offset
- *NN* is a placeholder for IFPD (it is always set to 'NN')
- *CCC* is the GOES Channel number, padded on the left with blanks (3 characters)
- *s* is the GOES Spacecraft (E or W)



## 9. Output Formatters

DECODES supports a variety of output formats including:

- SHEF – Standard Hydrometeorologic Exchange Format (.A lines)
- SHEFIT – Intermediate format defined by the USACE Hydrologic Engineering Center. Used to input data into the Corp’s CWMS database.
- Human Readable - Simple but compact time-sorted table format
- EMIT-ASCII - Compatible with EMIT when “ASCII” format is selected.
- EMIT-ORACLE - Compatible with EMIT when “ORACLE” format is selected.
- Dump – Used primarily for trouble-shooting, this format dumps all known information about samples, sensors, & platform.
- STDFMT - Standard format used by USGS for data-ingest into NWIS
- TransmitMonitor - Displays log of transmission quality parameters and battery voltage.

The following subsections contain more details on individual formats.

## 9.1 SHEF Output Format

The SHEF Output Formatter can produce either the “.A” or “.E” type lines:

- .E is normally used for regular interval data, such as is found in self-timed DCP messages. Figure 9-1 shows an example of the SHEF .A.
- .A is normally used for irregular interval data, such as is found in random DCP messages. Figure Figure 9-2 shows an example of SHEF .E.

The SHEF Formatter honors the following routing-spec properties:

<i>Name</i>	<i>Value Type</i>	<i>Default</i>	<i>Description</i>
dotAOnly	True/false	false	If true, force output to be .A lines only, even for self-timed (regular interval) data.
century	True/false	false	SHEF time stamps allow 4 digit or 2 digit years. The default is a 2 digit year. To force the century to be included, add this property set to “true”.
seconds	True/false	true	Likewise, seconds can be omitted in SHEF time stamps. By default they are included. To force them to be dropped, add a this property with a value of “false”.
useNesdisId	True/false	false	Normally the default Site Name is used in the SHEF output. To force the output to use the 8 hex-char NESDIS ID, set this to true.
fullShefCode	True/false	false	Normally the SHEF output will only include the 2-character physical element (PE) code entered with each sensor. If you want a full 7 digit code constructed by filling out the trailing 5 characters, set this to true.
defaultShefCode	7-char string	xxIRZZZ	If “fullShefCode” is set to true, you can control the characters used to fill-out the 7-character code.

```

.A BRFW3 011203 GMT+00:00 DH110000 /DUE /HG 38.36 :ft
.A BRFW3 011203 GMT+00:00 DH100000 /DUE /HG 38.35 :ft
.A BRFW3 011203 GMT+00:00 DH090000 /DUE /HG 38.34 :ft
.A BRFW3 011203 GMT+00:00 DH080000 /DUE /HG 38.35 :ft
.A BRFW3 011203 GMT+00:00 DH070000 /DUE /HG 38.35 :ft
.A BRFW3 011203 GMT+00:00 DH060000 /DUE /HG 38.35 :ft
.A BRFW3 011203 GMT+00:00 DH050000 /DUE /HG 38.35 :ft
.A BRFW3 011203 GMT+00:00 DH040000 /DUE /HG 38.35 :ft
.A BRFW3 011203 GMT+00:00 DH110000 /DUS /PC 6.26 :INCH
.A BRFW3 011203 GMT+00:00 DH100000 /DUS /PC 6.26 :INCH
.A BRFW3 011203 GMT+00:00 DH090000 /DUS /PC 6.26 :INCH
.A BRFW3 011203 GMT+00:00 DH080000 /DUS /PC 6.26 :INCH
.A BRFW3 011203 GMT+00:00 DH070000 /DUS /PC 6.26 :INCH
.A BRFW3 011203 GMT+00:00 DH060000 /DUS /PC 6.26 :INCH
.A BRFW3 011203 GMT+00:00 DH050000 /DUS /PC 6.26 :INCH
.A BRFW3 011203 GMT+00:00 DH040000 /DUS /PC 6.26 :INCH

```

**Figure 9-1: Example of SHEF .A Output**

```

.E SSIM5 020212 GMT DH150000 /DUS /VB/ DIH+1 /14.344 :V
.E LFKM5 020212 GMT DH080000 /DUE /HG/ DIH+1 /2.79/2.79/2.79/2.79/2.79/2.79/2.79/2.79 :ft
.E LFKM5 020212 GMT DH150000 /DUE /VB/ DIH+1 /14.344 :VOLT
.E VRNN8 020212 GMT DH150000 /DUE /VB/ DIH+1 /13.876 :VOLT
.E BRFW3 020212 GMT DH080000 /DUE /PC/ DIH+1 /6.26/6.26/6.26/6.26/6.26/6.26/6.26/6.26 :in
.E BRFW3 020212 GMT DH150000 /DUS /VB/ DIH+1 /14.5 :V
.E DURW3 020212 GMT DH080000 /DUE /HG/ DIH+1 /1.75/1.72/1.63/1.6/1.55/1.49/1.49/1.49 :ft
.E DURW3 020212 GMT DH150000 /DUS /VB/ DIH+1 /13.84 :V
.E HOMN8 020212 GMT DH160000 /DUS /VB/ DIH+1 /14.11 :V

```

**Figure 9-2: Example of SHEF .E output**

## 9.2 SHEFIT Output Format

Figure 9-3 shows an example of the HEC SHEFIT output format.

CE459D7E20011203110000	0	0	0	0	0	0	HP	RZZ	1055.530	Z	-1.00	0	0	0
CE459D7E20011203100000	0	0	0	0	0	0	HP	RZZ	1055.530	Z	-1.00	0	0	0
CE459D7E20011203090000	0	0	0	0	0	0	HP	RZZ	1055.530	Z	-1.00	0	0	0
CE459D7E20011203080000	0	0	0	0	0	0	HP	RZZ	1055.530	Z	-1.00	0	0	0
CE459D7E20011203070000	0	0	0	0	0	0	HP	RZZ	1055.530	Z	-1.00	0	0	0
CE459D7E20011203060000	0	0	0	0	0	0	HP	RZZ	1055.530	Z	-1.00	0	0	0
CE459D7E20011203050000	0	0	0	0	0	0	HP	RZZ	1055.530	Z	-1.00	0	0	0
CE459D7E20011203040000	0	0	0	0	0	0	HP	RZZ	1055.530	Z	-1.00	0	0	0
CE459D7E20011203030000	0	0	0	0	0	0	HP	RZZ	1055.530	Z	-1.00	0	0	0
CE459D7E20011203020000	0	0	0	0	0	0	HP	RZZ	1055.520	Z	-1.00	0	0	0
CE459D7E20011203010000	0	0	0	0	0	0	HP	RZZ	1055.520	Z	-1.00	0	0	0
CE459D7E20011203000000	0	0	0	0	0	0	HP	RZZ	1055.520	Z	-1.00	0	0	0
CE459D7E20011203110000	0	0	0	0	0	0	PC	RZZ	.000	Z	-1.00	0	0	0
CE459D7E20011203100000	0	0	0	0	0	0	PC	RZZ	.000	Z	-1.00	0	0	0
CE459D7E20011203090000	0	0	0	0	0	0	PC	RZZ	.000	Z	-1.00	0	0	0
CE459D7E20011203080000	0	0	0	0	0	0	PC	RZZ	.000	Z	-1.00	0	0	0
CE459D7E20011203070000	0	0	0	0	0	0	PC	RZZ	.000	Z	-1.00	0	0	0
CE459D7E20011203060000	0	0	0	0	0	0	PC	RZZ	.000	Z	-1.00	0	0	0

Figure 9-3: Example of SHEFIT Output Format.

### 9.3 Human Readable Output Format

The Human Readable Formatter is designed, well, for humans. It displays the message data in the simple table format shown in Figure 9-4. It also honors the following properties:

<i>Name</i>	<i>Value Type</i>	<i>Default</i>	<i>Description</i>
displayEmpty	True/false	false	Normally, empty columns will be omitted. Add this property and set it to true to cause a column to be displayed even for sensors that have no data.

Message for Platform NWSHB5-HOMN8				
	elev	PC	battery	
	HP	PC	VB	
	ft	in	V	
12/03/2001 00:00:00	1055.53	0.0		
12/03/2001 01:00:00	1055.53	0.0		
12/03/2001 02:00:00	1055.53	0.0		
12/03/2001 03:00:00	1055.53	0.0		
12/03/2001 04:00:00	1055.53	0.0		
12/03/2001 05:00:00	1055.53	0.0		
12/03/2001 06:00:00	1055.53	0.0		
12/03/2001 07:00:00	1055.53	0.0		
12/03/2001 08:00:00	1055.53	0.0		
12/03/2001 09:00:00	1055.52	0.0		
12/03/2001 10:00:00	1055.52	0.0		
12/03/2001 11:00:00	1055.52	0.0	13.876	
Message for Platform NWSHB5-WTSM5				
	pool	tail	battery	
	HP	HT	VB	
	ft	ft	VOLT	
12/03/2001 00:00:00	900.0	935.5		
12/03/2001 01:00:00	900.0	935.49		
12/03/2001 02:00:00	900.0	935.5		
12/03/2001 03:00:00	900.0	935.51		
12/03/2001 04:00:00	900.0	935.54		
12/03/2001 05:00:00	900.0	935.61		
12/03/2001 06:00:00	900.0	935.65		
12/03/2001 07:00:00	900.0	935.67		
12/03/2001 08:00:00	900.0	935.67		
12/03/2001 09:00:00	900.0	935.65		
12/03/2001 10:00:00	900.0	935.64		
12/03/2001 11:00:00	900.0	935.61	12.004	

Figure 9-4: Example of Human Readable Output Format.

## 9.4 EMIT-ASCII Format

If the routing spec contains a string property called ‘delimiter’, this will be used to delimit between columns. The default is a single space.

The EMIT-ASCII formatter produces an output that is compatible with the old EMIT program when “ASCII” was selected as the output format. This format has 12 blank-delimited fields as follows:

- Hex DCP Address
- EPA Sensor Code (0 if none is assigned)
- Sensor Number
- Time Stamp in the format: YYDDD/HH:MM:SS
- Sample Value (formatted as specified by Presentation Group)
- ‘I’ if this is a self-timed message (meaning interval data); or ‘R’ if this is a random message.
- DCP Name (the preferred site name as specified by your properties file is used)
- Sensor Name
- SHEF Code (or ‘XX’ if none is specified)
- Recording interval for this sensor (in seconds)
- ‘I’
- Engineering Units

Following all sample data, a single line with ‘ZZZZ’ is printed. Figure 9-5 shows a single message in EMIT-ASCII format.

If you have used station or sensor names that have embedded spaces, you can use an additional property ‘useQuotes’ set to TRUE. This will cause the station and sensor names to be enclosed in single quotes.

CE459D7E	0	1	01337/11:00:00	1055.53	I	HOMN8	elev	HP	3600	I	ft
CE459D7E	0	1	01337/10:00:00	1055.53	I	HOMN8	elev	HP	3600	I	ft
CE459D7E	0	1	01337/09:00:00	1055.53	I	HOMN8	elev	HP	3600	I	ft
CE459D7E	0	1	01337/08:00:00	1055.53	I	HOMN8	elev	HP	3600	I	ft
CE459D7E	0	1	01337/07:00:00	1055.53	I	HOMN8	elev	HP	3600	I	ft
CE459D7E	0	1	01337/06:00:00	1055.53	I	HOMN8	elev	HP	3600	I	ft
CE459D7E	0	1	01337/05:00:00	1055.53	I	HOMN8	elev	HP	3600	I	ft
CE459D7E	0	1	01337/04:00:00	1055.53	I	HOMN8	elev	HP	3600	I	ft
CE459D7E	0	1	01337/03:00:00	1055.53	I	HOMN8	elev	HP	3600	I	ft
CE459D7E	0	1	01337/02:00:00	1055.52	I	HOMN8	elev	HP	3600	I	ft
CE459D7E	0	1	01337/01:00:00	1055.52	I	HOMN8	elev	HP	3600	I	ft
CE459D7E	0	1	01337/00:00:00	1055.52	I	HOMN8	elev	HP	3600	I	ft
CE459D7E	00045	2	01337/11:00:00	0.0	I	HOMN8	PC	PC	3600	I	in
CE459D7E	00045	2	01337/10:00:00	0.0	I	HOMN8	PC	PC	3600	I	in
CE459D7E	00045	2	01337/09:00:00	0.0	I	HOMN8	PC	PC	3600	I	in
CE459D7E	00045	2	01337/08:00:00	0.0	I	HOMN8	PC	PC	3600	I	in
CE459D7E	00045	2	01337/07:00:00	0.0	I	HOMN8	PC	PC	3600	I	in
CE459D7E	00045	2	01337/06:00:00	0.0	I	HOMN8	PC	PC	3600	I	in
CE459D7E	00045	2	01337/05:00:00	0.0	I	HOMN8	PC	PC	3600	I	in
CE459D7E	00045	2	01337/04:00:00	0.0	I	HOMN8	PC	PC	3600	I	in
CE459D7E	00045	2	01337/03:00:00	0.0	I	HOMN8	PC	PC	3600	I	in
CE459D7E	00045	2	01337/02:00:00	0.0	I	HOMN8	PC	PC	3600	I	in
CE459D7E	00045	2	01337/01:00:00	0.0	I	HOMN8	PC	PC	3600	I	in
CE459D7E	00045	2	01337/00:00:00	0.0	I	HOMN8	PC	PC	3600	I	in
CE459D7E	70969	3	01337/11:00:00	13.876	I	HOMN8	battery	VB	3600	I	V
ZZZZ											

Figure 9-5: Example of EMIT-ASCII format.

## 9.5 EMIT-Oracle Format

This format is similar to EMIT-ASCII but more compact. It was originally designed to input data into an Oracle database, hence the name. It is, however, a generally useful format in its own right, very easy to parse with a computer program.

The 'delimiter' property is supported in the same way as for EMIT-ASCII.

The EMIT-ORACLE formatter produces an output that is compatible with the old EMIT program when "ORACLE" was selected as the output format. This format has 7 blank-delimited fields as follows:

- Hex DCP Address
- SHEF Code (or 'XX' if none is specified)
- Sensor Number
- Time Stamp in the format: YYDDD/HH:MM:SS
- Sample Value (formatted as specified by Presentation Group)
- 'I' if this is a self-timed message (meaning interval data); or 'R' if this is a random message.
- Engineering Units

Following all sample data, a single line with 'ZZZZ' is printed. Figure 9-6 shows a single message in EMIT-Oracle format.

```
CE459D7E HP 1 01337/11:00:00 1055.53 I ft
CE459D7E HP 1 01337/10:00:00 1055.53 I ft
CE459D7E HP 1 01337/09:00:00 1055.53 I ft
CE459D7E HP 1 01337/08:00:00 1055.53 I ft
CE459D7E HP 1 01337/07:00:00 1055.53 I ft
CE459D7E HP 1 01337/06:00:00 1055.53 I ft
CE459D7E HP 1 01337/05:00:00 1055.53 I ft
CE459D7E HP 1 01337/04:00:00 1055.53 I ft
CE459D7E HP 1 01337/03:00:00 1055.53 I ft
CE459D7E HP 1 01337/02:00:00 1055.52 I ft
CE459D7E HP 1 01337/01:00:00 1055.52 I ft
CE459D7E HP 1 01337/00:00:00 1055.52 I ft
CE459D7E PC 2 01337/11:00:00 0.0 I in
CE459D7E PC 2 01337/10:00:00 0.0 I in
CE459D7E PC 2 01337/09:00:00 0.0 I in
CE459D7E PC 2 01337/08:00:00 0.0 I in
CE459D7E PC 2 01337/07:00:00 0.0 I in
CE459D7E PC 2 01337/06:00:00 0.0 I in
CE459D7E PC 2 01337/05:00:00 0.0 I in
CE459D7E PC 2 01337/04:00:00 0.0 I in
CE459D7E PC 2 01337/03:00:00 0.0 I in
CE459D7E PC 2 01337/02:00:00 0.0 I in
CE459D7E PC 2 01337/01:00:00 0.0 I in
CE459D7E PC 2 01337/00:00:00 0.0 I in
CE459D7E VB 3 01337/11:00:00 13.876 I V
ZZZZ
```

Figure 9-6: Example of Emit-Oracle Output Format.

## 9.6 Dump Formatter

DumpFormatter is useful for testing and trouble-shooting. It dumps the raw message, performance measurements, and decoded data to an output interface. Figure 9-7 shows an example of this format.

```
=====  
Start of message for platform NWSHB5-HOMN8  
Time Stamp: 12/02/2001 16:08:11  
Raw Data:  
CE459D7E01336210811G44-  
4NN031E9200077B1HAVq@@@Avq@@@Avq@@@Avq@@@Avq@@@Avq@@@Avq@@@Avp@@@Avp  
@@@Avp@@@N  
  
Performance Measurements:  
DcpAddress=CE459D7E  
Spacecraft=E  
UplinkCarrier=92  
Channel=31  
SignalStrength=44  
Length=77  
ModulationIndex=N  
Quality=N  
Time=12/02/2001 21:08:11  
FailureCode=G  
FrequencyOffset=-4  
  
Decoded Data:  
  
Sensor 1: elev, EU=ft(feet), DataType=SHEF-PE:HP  
Begin=12/02/2001 16:53:33, End=12/03/2001 06:00:00  
Number of Samples=12  
Sample[0]=12/03/2001 06:00:00: 1055.53 ' 1055.53'  
Sample[1]=12/03/2001 05:00:00: 1055.53 ' 1055.53'  
Sample[2]=12/03/2001 04:00:00: 1055.53 ' 1055.53'  
Sample[3]=12/03/2001 03:00:00: 1055.53 ' 1055.53'  
Sample[4]=12/03/2001 02:00:00: 1055.53 ' 1055.53'  
Sample[5]=12/03/2001 01:00:00: 1055.53 ' 1055.53'  
Sample[6]=12/03/2001 00:00:00: 1055.53 ' 1055.53'  
Sample[7]=12/02/2001 23:00:00: 1055.53 ' 1055.53'  
Sample[8]=12/02/2001 22:00:00: 1055.53 ' 1055.53'  
Sample[9]=12/02/2001 21:00:00: 1055.52 ' 1055.52'  
Sample[10]=12/02/2001 20:00:00: 1055.52 ' 1055.52'  
Sample[11]=12/02/2001 19:00:00: 1055.52 ' 1055.52'  
Sensor 2: PC, EU=in(inches), DataType=SHEF-PE:PC  
Begin=12/02/2001 16:53:33, End=12/03/2001 06:00:00  
Number of Samples=12  
Sample[0]=12/03/2001 06:00:00: 0 '0.0 '  
Sample[1]=12/03/2001 05:00:00: 0 '0.0 '  
Sample[2]=12/03/2001 04:00:00: 0 '0.0 '  
Sample[3]=12/03/2001 03:00:00: 0 '0.0 '  
Sample[4]=12/03/2001 02:00:00: 0 '0.0 '  
Sample[5]=12/03/2001 01:00:00: 0 '0.0 '  
Sample[6]=12/03/2001 00:00:00: 0 '0.0 '  
Sample[7]=12/02/2001 23:00:00: 0 '0.0 '  
Sample[8]=12/02/2001 22:00:00: 0 '0.0 '  
Sample[9]=12/02/2001 21:00:00: 0 '0.0 '  
Sample[10]=12/02/2001 20:00:00: 0 '0.0 '  
Sample[11]=12/02/2001 19:00:00: 0 '0.0 '  
Sensor 3: battery, EU=V(volts), DataType=SHEF-PE:VB  
Begin=12/02/2001 16:53:33, End=12/03/2001 06:00:00  
Number of Samples=1  
Sample[0]=12/03/2001 06:00:00: 13.876 ' 13.876
```

Figure 9-7: Example of Dump Output Format



## 9.7 USGS STDFMT Output Formatter

This formatter is used by USGS for ingesting data into the National Water Information System (NWIS). For documentation on this format see Appendix E in the NWIS User Guide, which can be found at:

<http://wa.water.usgs.gov/realtime/adaps/adaps.book.html>

Figure 9-8 shows an example of STDFMT output. Each DCP message is placed in a separate STDFMT envelope.

```
BE STDDCP
DB 1 1
SD USGS      03323500  ON
SE   8  STAGE00065  11 73F010000
TM 20021030130000
UF   8 10.390 10.390 10.390 10.390 10.390 10.390 10.390 10.390
SE   3  H2O T00010  11 73F010000
TM 20021030130000
UF   8 10.600 10.600 10.600 10.700 11.100 11.400 11.600 11.800
SE   9  BATVT70969  11 63F010000
TM 20021030130000
UF   8 3.740 3.740 3.770 3.850 4.210 4.100 4.000 4.110
EE
BE STDDCP
DB 1 1
SD USGS      03324500  ON
SE   2  STAGE200065  11 63F010000
TM 20021030130000
UF   8 4.540 4.540 4.540 4.540 4.540 4.540 4.540 4.540
SE   5  PREC 200045   6 83F010000
TM 20021030130000
UF   8 116.700 116.700 116.700 116.700 116.700 116.700 116.700 116.700
SE   3  H2O T100010  11 73F010000
TM 20021030130000
UF   8 13.000 13.000 13.000 13.000 13.000 13.000 13.000 13.000
SE   8  BATVTB70969  11 73F010000
TM 20021030130000
UF   8 15.310 15.210 14.910 14.720 14.830 14.710 14.710 14.830
EE
```

Figure 9-8: Example of USGS STDFMT Output.

## 9.8 Transmit Monitor Formatter

The Transmit Monitor format provides a log of transmission quality measurements in an easy-to-use row column format. The following columns are used by default:

- Message Time Stamp in the form MM/DD/YYYY-HH:MM:SS
- DCP Address (Transport Medium ID)
- Site Name
- Failure Code
- SignalStrength
- Message Length
- GOES Channel Number
- Frequency Offset
- Modulation Index
- Battery Voltage

An example of the default format is shown in Figure 9-9.

10/30/2002-20:03:33	CE7718EE	03324500	G	50	209	23	-4	N	N	14.83
10/30/2002-20:16:50	CE77835E	03360000	G	50	97	23	-5	N	N	13.88
10/30/2002-20:29:25	CE777D08	03327500	G	50	161	23	-2	N	N	14.37
10/30/2002-21:03:11	CE14B3F8	03324000	G	50	145	17	0	H	N	13.70
10/30/2002-21:07:22	CE14C568	03275000	G	50	113	17	-1	N	N	13.74
10/30/2002-22:21:29	CE6D361C	03335500	G	49	113	41	-4	N	F	14.12
10/31/2002-00:03:33	CE7718EE	03324500	G	49	209	23	-3	N	N	14.72
10/31/2002-00:05:30	CE772D74	03375500	G	49	105	23	-3	N	N	13.3
10/31/2002-00:06:27	CE7730D0	03276000	G	49	145	23	2	N	N	14.8
10/31/2002-00:16:50	CE77835E	03360000	G	49	97	23	-5	N	N	14.23
10/31/2002-00:29:25	CE777D08	03327500	G	50	161	23	-2	N	N	13.99
10/31/2002-01:03:11	CE14B3F8	03324000	G	50	145	17	0	H	N	13.70
10/31/2002-02:21:29	CE6D361C	03335500	G	50	113	41	-4	N	N	14.11
10/31/2002-04:16:50	CE77835E	03360000	G	49	97	23	-5	N	N	13.88
10/31/2002-04:29:25	CE777D08	03327500	G	49	161	23	-2	N	N	13.79
10/31/2002-05:03:11	CE14B3F8	03324000	G	49	145	17	0	H	N	13.70
10/31/2002-05:05:41	CE14D61E	03357500	G	50	145	17	-9	N	N	14.70
10/31/2002-05:07:22	CE14C568	03275000	G	50	113	17	-1	N	N	13.57
10/31/2002-06:21:29	CE6D361C	03335500	G	50	113	41	-4	N	N	14.17

**Figure 9-9: Example of Transmit Monitor Format.**

You can control the contents of the transmit monitor format by adding properties to the routing specification:

- The string property “delimiter” has a default value of a single space character. This is used to separate columns in the output. To ingest this data into a SQL database, for example, you may wish to use a comma as a delimiter.
- The Boolean property “justify” defaults to ‘true’. This causes each column to be either right or left justified within the column width. The example above shows justified columns.

The string property “columns” is a blank or comma-separated list of columns that you wish to see in the output. Table 9-1 shows the column names that can be included in this string. The default value for the string is:

```
"time id name FailureCode SignalStrength Length Channel FrequencyOffset
ModulationIndex Quality batt"
```

<i>Column Name</i>	<i>Description</i>
time	Message time stamp in the format MM/DD/YYYY-HH:MM:SS
id	Transport ID (i.e. DCP address for GOES messages)
name	Site name
FailureCode	1-character code for GOES messages: 'G' means good message, '?' means parity errors.
Length	Length of the raw message in bytes
Channel	GOES Channel number
FrequencyOffset	A sign plus a digit, taken from the DOMSAT message header, this indicates the frequency offset of the raw message, as reported by DAPS. The digit indicates the amount of the offset in units of 50Hz.
ModulationIndex	'N' for Normal, 'L' for Low, 'H' for High
Quality	'N' (normal) = Error rate better than $10^{-6}$ , 'F' (fair) = Error rate between $10^{-4}$ and $10^{-6}$ 'P' (poor) = Error rate worse than $10^{-4}$
SignalStrength	in dB.
Spacecraft	'E' (East), or 'W' (West)
UplinkCarrier	Uplink Carrier Status (not implemented in DAPS-I)
batt	Battery voltage if available. The most recent sample contained in the message will be printed. This looks for a sensor with a name that starts with "batt". If none found it looks for any sensor with a datatype equivalent to VB.

**Table 9-1: Column Names supported by Transmit Monitor Formatter.**

The string property "colwidths" is used to control the width and justification of each column. It should be a blank or comma-separated list of numbers, one for each column. A positive number means right-justified. A negative number means left-justified. The default value of this property is:

19, 8, 10, 1, 5, 3, 2, 2, 2, 5

Example: Cause the formatter to print a comma-separated list of messages. For each message we only want the time, DCP Address, and battery voltage.

Add the following properties to the routing spec:

- delimiter = , (i.e. a single comma)
- justify = false
- columns = time id batt
- colwidths = 19, 8, 7

## 10. Consumers

Consumers receive the formatted data created by DECODES and send it somewhere. There are currently 4 types of consumers implemented within DECODES:

- PipeConsumer is used to send data from a DECODES routing spec into some other program in real time.
- FileConsumer sends all data from a routing spec into a single file. The file is closed when the routing spec is complete.
- DirectoryConsumer creates separate files for each message in a specified directory.
- StringBufferConsumer is used internally by GUI programs that display decoded data interactively.

### 10.1 Pipe Consumer

The Consumer Argument should be one of:

- 'stdout' – send data to standard output.
- 'stderr' – send data to standard error.
- *command* - an arbitrary command line. The command will be executed and the data will be piped to the command's standard input.

PipeConsumer will use the following routing-spec properties to control its actions:

<i>Property Name</i>	<i>Default</i>	<i>Description</i>
ConsumerBefore	none	An encoded string that is written to the file preceding each message. The string may contain UNIX-style escape sequences such as \n \r \t, and octal binary characters encoded as \002, etc.
ConsumerAfter	none	An encoded string that is written to the file after each message.

## 10.2 File Consumer

The Consumer Argument should be the file name template. The file will be opened when the routing spec starts. All data from the routing spec will be placed in the file. The file will be closed when finished.

The file name template may contain a variable of the form `$DATE(format)` where *format* describes how the date/time stamp is to be formatted in the file name. It can contain any format handled by the Java class `java.text.SimpleDateFormat`, although since it is used as a filename, it should not contain spaces or other illegal characters.

See <http://java.sun.com/j2se/1.4.1/docs/api/> for complete docs on `SimpleDateFormat`. Click on “java.text” in the upper left frame. Then click on `SimpleDateFormat` in the lower left frame.

For example, if Consumer Arg is “data-\$DATE(yyyyMMdd-HHmms)”, this might result in a filename `data-20031213-120000`.

This consumer should therefore only be used with routing specs that run for a finite period of time. That is, the ‘until’ time should be specified if reading from an LRGS.

FileConsumer will use the following routing-spec properties to control its actions:

<i>Property Name</i>	<i>Default</i>	<i>Description</i>
ConsumerBefore	none	An encoded string that is written to the file preceding each message. The string may contain UNIX-style escape sequences such as <code>\n</code> <code>\r</code> <code>\t</code> , and octal binary characters encoded as <code>\002</code> , etc.
ConsumerAfter	none	An encoded string that is written to the file after each message.
file.overwrite	false	Set this property to true if you want the consumer to overwrite files that already exist. The default behavior is to append to the existing file.
cmdAfterFile	none	A command that DECODES will execute after each file is generated and placed in the directory. The command may contain <code>\$FILENAME</code> which will be replaced with the name of the file just completed.
cmdTimeout	Integer	Number of seconds for <code>cmdAfterFile</code> to complete. Default=60 seconds.

### 10.3 Directory Consumer

The Consumer Argument should be a directory name. The routing spec will create a separate file in this directory to hold the data generated for each message. The file name will be in the following format:

*SiteName-YYYYMMDDHHmmSS*

Hence when looking at a sorted directory listing you will see each platform's files together in time order.

The Site Name used will be the default site name type defined in your DECODES properties file.

DirectoryConsumer will use the following routing-spec properties to control its actions:

<i>Property Name</i>	<i>Default</i>	<i>Description</i>
ConsumerBefore	none	An encoded string that is written to the file preceding each message. The string may contain UNIX-style escape sequences such as \n \r \t, and octal binary characters encoded as \002, etc.
ConsumerAfter	none	An encoded string that is written to the file after each message.
cmdAfterFile	none	A command that DECODES will execute after each file is generated and placed in the directory. The command may contain \$FILENAME which will be replaced with the name of the file just completed.
cmdTimeout	Integer	Number of seconds for cmdAfterFile to complete. Default=60 seconds.
filename	none	A file-name template to override the default described above. (see below)

Files will be constructed in a temporary location and then moved to the named directory. Therefore, you can write a program to scan the directory for new files and be assured that all files in the directory are complete.

If you supply a filename property, it will be used to construct the filename, overriding the default described above. The template may contain variables of the following form:

- \$DATE(*format*) - See the description of this in section 10.2.
- \$TRANSPORTID - will be replaced by the DCP address.
- \$SITENAME - will be replaced by the site name.

## 11. Special Considerations for EDL Files

EDL (Electronic Data Logger) files can be processed by DECODES routing specs just as easily as DCP messages. This section highlights some of the differences that you'll need to be aware of in setting up your database.

### 11.1 How does DECODES find the Platform Record?

If file contains USGS header with a complete values for STATION and DEVICE, then DECODES can construct a transport medium ID as follows:

```
station-devname-devnum
```

Example: Suppose the file header contains:

```
//STATION 01234567  
//DEVICE CR10 1
```

...then the medium ID would be “01234567-CR10-1”. Your platform record would need to have a Transport Medium record with this value.

If your files do not contain a complete USGS header, then you can supply it on the command line with a -D argument. For example:

```
rs -Dfilename=myfile -DMediumID=01234567-CR10-1 myspecname
```

(station number with device number), for example “//STATION cr10

MediumID property set on command line for files that are missing STATION or device number in the file. Example:

```
rs -DMediumID=0143563-cr10-3 specname
```

### 11.2 Time Zones for Dates & Times in EDL Files

Time zone abbreviations can be one of:

- Standard time zone name like EST or America/Chicago.
- Custom Java time zone in the form “GMT-HH:MM”. For example, Eastern US that never uses daylight time could be “GMT-05:00”.
- A sign followed by a minute offset, followed by a flag indicating whether or not daylight time applies: Y, N, or M (see below).

Time zones (as of DECODES 6.1) are stored in the transport medium record (see Figure 5-4). If none is specified here, the time zone specified in the site record is used. If none there, then “UTC” is assumed.

The daylight time flag can take on three values:

- Y means that daylight time applies according to the normal domestic U.S. rules. That is, daylight time starts at 2 AM on the first Sunday of April, and ends at 2 AM on the last Sunday of October.

- N means that daylight time never applies.
- M means Manual. That is, the platform is set manually to the new time zone by the operator when the site is visited to collect the data.

In the case of manual daylight change, there are special rules: The time will be changed on or after the period starts. That is, the change to daylight time must occur after 2 AM on the first Sunday of April. The change back to normal time must occur after 2 AM on the last Sunday of October. Also, the operator must make the change after collecting the data, so that all data within a single EDL file has the same offset.

Note that the use of daylight time in general, and manual change in particular, is error-prone. You are strongly discouraged from this practice. The ideal situation is to program your recorder always in UTC, and to let DECODES make the conversions as needed when the data is parsed.



## 12. Specific Scenarios

### 12.1 How To Create a New Platform Specification

#### Create a Site for the New Platform

First create a Site for this platform. Recall that in DECODES a “Site” refers to the location. The Platform resides at the Site.

1. Start the Database Editor by typing ‘dedit’.
2. Press the ‘Sites’ tab.
3. Make sure the site you want to create doesn’t already exist. Press the column headers to sort by the various name types. If the site already exists, skip ahead to “Create a Configuration”.
4. Press the ‘New’ button at the bottom of the Site List Panel. This creates a new site record and opens it. You now see a Site Edit Panel with the newly created site.
5. A site must have at least one valid name. The new panel shows a site with your default name-type (probably NWSHB5) and the name “NewName”. Change these to the proper (unique) name for the new site.
6. On the right side of the Site Edit Panel you can enter other descriptive information about the site, such as Latitude, Longitude, Nearest City, etc. At a minimum you should enter the correct time zone for this site.
7. At the bottom of the Site Edit Panel, press ‘Commit’ and the ‘Close’. You should now see your new site in the site list.

#### Create an Equipment Model Record for the New Platform

Equipment Model records hold information about each vendor’s platform. Your platform’s configuration will be associated with an Equipment Model.

8. Press the ‘Equipment’ tab at the top of the editor window. You now see the Equipment Model List Tab.
9. Each equipment model is given a unique abbreviated name. For example “SU8004D” is the name for a Sutron 8004 DCP. Does the equipment model for your new platform already exist? If so, make a note of it and skip ahead to “Create a Configuration”.
10. Press “New” at the bottom of the Equipment Model List Panel. You are prompted to enter a new name. There should be no spaces in the name and it must be unique. A new Equipment Model record is created and opened. You now see the open Equipment Model Edit Panel.
11. In the edit panel type the descriptive information about the equipment. The Type value must be set to “DCP.”
12. Press the “Add” button in the Properties area. Create a property called “DataOrder” (no spaces). The value should be either **A** or **D**. ‘A’ means

ascending, meaning that in messages from this platform, the oldest samples are transmitted first.

13. Press “Commit” and “Close” when you are finished with the Equipment Model.

### **Create a Configuration for the New Platform**

14. Next you need to create a configuration for the new platform. Press the ‘Configs’ tab at the top of the editor. You should now see the Configuration List Panel.

Recall that in DECODES, most of the information about how to decode and time-tag data is part of the configuration. A configuration can be shared by several platforms. For example if you installed 8 Sutron 8200 platforms with exactly the same sensors, and they all are programmed to generate identically-formatted messages; then you only need to create one configuration that all 8 platforms can share.

Do you already have a configuration for a platform identical to the new platform? If so, you can use it. Find its name in the list and make a note of it. Then skip ahead to “Create the Platform Record”.

15. Press the New button at the bottom of the Configuration List Panel. You are prompted to enter a unique name for the configuration.

## ***Rules and Conventions for Configuration Naming***

The only hard rules for configuration names are:

The name must be unique. No two configurations can have the same name.

The name cannot contain any spaces. We recommend limiting the name to alphanumeric and hyphens.

You should establish an agency-wide convention for naming configurations. You want to be able to exchange configurations with other districts (and other agencies) without fear of a name clash.

The USGS has a well established naming convention that other agencies are encouraged to follow:

*EquipmentModelName-Organization-Sequence*

...where:

*EquipmentModelName* is the abbreviation of the Equipment Model record associated with this configuration. Example “SU8004D” for Sutron 8004 DCP.

*Organization* is an abbreviation that uniquely identifies your organization and district. For example “ACEMD” might mean the U.S. Army Corps of Engineers Maryland District.

*Sequence* is simply a sequence number.

The following are examples of existing configuration names:

SU8200D-ILEX-005

HA555D-ACEAL-017

HA570D-ACETN-015

### ***Enter the Sensor and Formatting Information***

After creating a new configuration, a Config Edit Panel will be opened. This screen contains a lot of important information.

16. Press the 'Select' button to the right of "Equipment Model". Select the equipment model from the pop-up list.
17. In the text area, type a brief description for this configuration. The description might contain the number of sensors, SHEF codes, etc.
18. Press the 'Add' button in the Sensors area of the screen. The "Edit Config Sensor Dialog" appears. Fill out the form. Enter a name and data-type for the sensor. Enter the correct sampling time and interval (this is important for correctly time-tagging samples from each sensor). When finished, press OK.
19. Repeat the previous step for each sensor.
20. Press the 'Add' button in the Decoding Scripts area. The "Edit Decoding Script Dialog" appears. A Decoding Script tells DECODES how to extract samples from the raw message.
21. Enter a script name in the upper right of the dialog. Commonly used names are "ST" for self-timed GOES messages and "RD" for random GOES messages.
22. Enter the format statements manually. Order is important. The script will start with the first statement. Use the "Up" and "Dn" buttons to move format statements, if necessary.
23. In the "Sensor Unit Conversions" area of the screen, enter the units and conversions for each sensor. Enter the units abbreviation (see standard EU abbreviations table below).
24. If no conversion is necessary, leave algorithm set to "None". This is appropriate if the DCP reports values that are already in engineering units. For a linear conversion, select Algorithm=linear. Then enter A and B coefficients for the equation:  
$$\mathbf{EU} = \mathbf{A} * \mathbf{RAW} + \mathbf{B}$$
25. Test your script. Load a raw DCP message into the "Sample Message" text area. The "Load" button lets you load from a file. Even more convenient: Open a LRGS Browser window and cut & paste. The sample area must start with the DCP address at the beginning of the DOMSAT header in the DCP message. Any spaces or other characters before the DCP address will cause decoding to fail.
26. Press the "Decode" button. This will apply the format statements and unit conversions to the raw data in the sample area. See the results in the Decoded Data area.
27. Tweak the format statements and unit conversions, then press "Decode" again. Repeat until decoding is correct. Then press OK.

28. If your platform sends both self-timed and random messages, you will need to create a separate Decoding Script for each. We recommend calling the Self-Timed Script “ST” and the Random Script “RD”.

### Create the Platform Record

You are finally ready to create the platform!

29. Press the “Platforms” tab at the top of the editor.
30. Press the “New” button at the bottom of the screen. This creates a new (empty) platform record and opens it in a Platform Edit Panel.
31. Press the “Choose” button to the right of “Site”. Select the new site that you created above.
32. Type a brief description for this platform. This will show up in the list of platforms for your own easy reference.
33. Press the “Choose” button to the right of “Config”. Select the configuration that you prepared above. Click OK when the pop-up tells you about the dangers of changing the configuration assignment. You should now see a list of sensors in the “Platform Sensor Information” area.
34. Press the “Add” button on the right of the Transport Media area. Fill out the dialog. A **Transport Medium** is the glue that associates an incoming message with your platform records and your decoding scripts.
35. Under Medium Type, select “goes-self-timed”.
36. Under Medium Identifier, type the 8-hex-digit DCP address.
37. Under Decoding Script, select the name of the self timed script (e.g. “ST”).
38. Enter the channel numbers, and if this is for a self-timed message, enter the time and interval values. Then press OK.
39. If this platform also transmits random, press Add again, select “goes-random” for Medium Type and “RD” for Decoding Script. Re-enter the DCP address.
40. At the bottom of the Platform Edit Panel press “Commit” and then “Close”.

### **Add the new Platform to a Network List**

41. Press the “Netlists” tab at the top of the editor.
42. Highlight the list you want to edit and press “Open”.
43. In the Network List Edit panel, press the “Add” button on the right.
44. Select your new site from the pop-up list and press OK.
45. Press “Commit” and “Close” at the bottom of the screen.

### **Testing the new Platform in a Routing Spec**

46. Run a routing spec that uses the network list that you modified.
47. In the output data, you should see data from your new platform.

## 13. Reference List Editor

The DECODES database contains information to populate pull-down lists. You will probably never need to modify this information.

However, if you need to expand DECODES functionality you will need to use the Reference List Editor.

To start the program, type 'rledit' at the command prompt. The initial screen is shown in Figure 13-1. Along the top, you see four tabs for the four types of reference lists:

- Enumerations: Used to populate pull-down lists in the database editor, and also to expand DECODES functionality in some cases.
- Engineering Units: This tab contains the units known to DECODES.
- Engineering Unit Conversions: This tab contains the conversions between units.
- Data Type Equiv: This tab contains the known data types, and assertions as to equivalence.

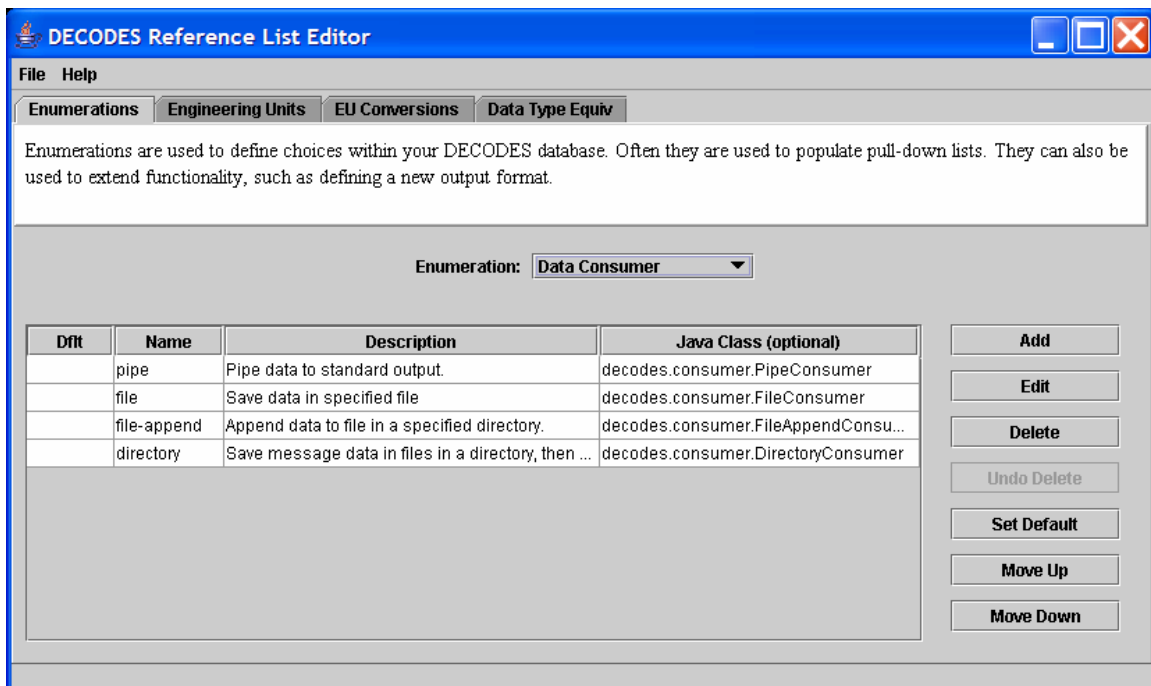


Figure 13-1: Reference List Editor Enumerations Tab.

## 13.1 Enumerations

There are several enumeration sets within DECODES. Select one from the pull-down listed labeled “Enumeration”. Notice that when you select a different enumeration, the table below is populated with the values in that set.

Each set has a particular purpose:

- Data Consumer – These are shown in the dbedit routing spec panel. This set associates a name like ‘pipe’ with a Java class that implements the consumer, like ‘decodes.consumer.PipeConsumer’.
- Data Source Type – These values are shown in the dbedit Data Source panel. Each DECODES data source is associated with a type. This set tells DECODES which Java class to use for each type.
- Data Type Standard – This set defines the data type coding standards that you use. Common values are ‘shef-pe’ used by USACE, and ‘epa-code’ used by USGS.
- Measures – This is used for engineering units. Every EU measures some physical quantity like area, length, flow, volume, etc. This is a list of those physical quantities.
- Output Format - These are shown in the dbedit routing spec panel where you select the format for output data. This set associates each name with its Java class.
- Script Type – DECODES was designed to support several types of scripts, although currently only the ‘standard’ script is used.
- Site Name Type – Sites may have many names, but only one of each type. This set defines the name type columns that appear in dbedit.
- Transport Medium Type – Each TM has a type like GOES-Self-Timed, or “Data-Logger”. This set determines the values shown in the pull down list in dbedit.
- Unit Conversion Algorithm – currently contains four values for the different conversion types: none, linear, poly-5, and usgs-standard.

Using the buttons on the right you can add, edit, or delete enumeration values.

The ‘Set Default’ button places an asterisk next to the selected value. In certain cases, the default value is used in the absence of a user selection.

The order of values in the list determines the order they will appear in a pull-down list. Hence you can use the Move-Up and Move-Down buttons to change the order shown here.



## 13.2 Engineering Units

The Engineering Units tab is shown in Figure 13-2. This list defines all of the known EUs in DECODES.

You can click on the column headers to sort by:

- Abbreviation
- Full Name
- Family (i.e. English, Metric, or Universal)
- Measures (the physical quantity being measured by the EU)

You can use the buttons to the right to add, edit, or delete an EU.

Abbreviations must not contain embedded spaces.

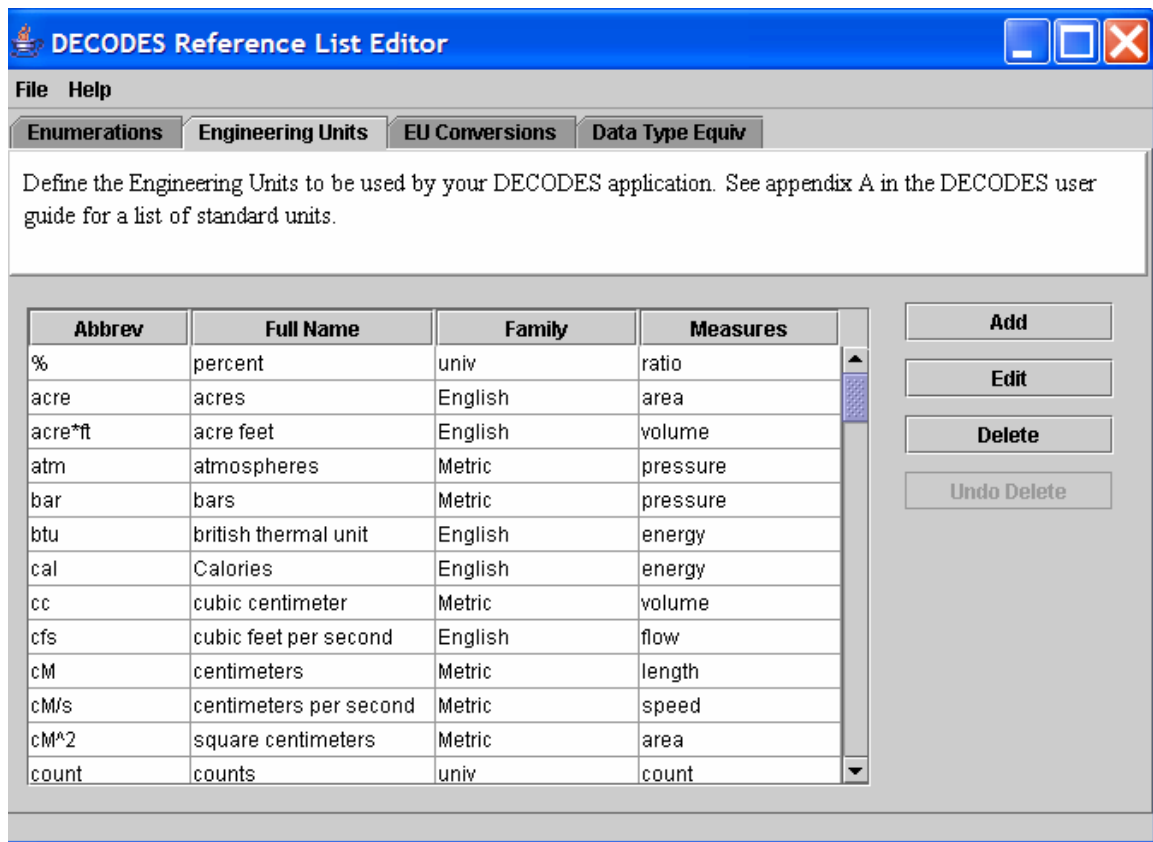


Figure 13-2: Reference List Editor Engineering Units Tab.

## 13.3 Engineering Unit conversions

The EU conversions tab is shown in Figure 13-3.

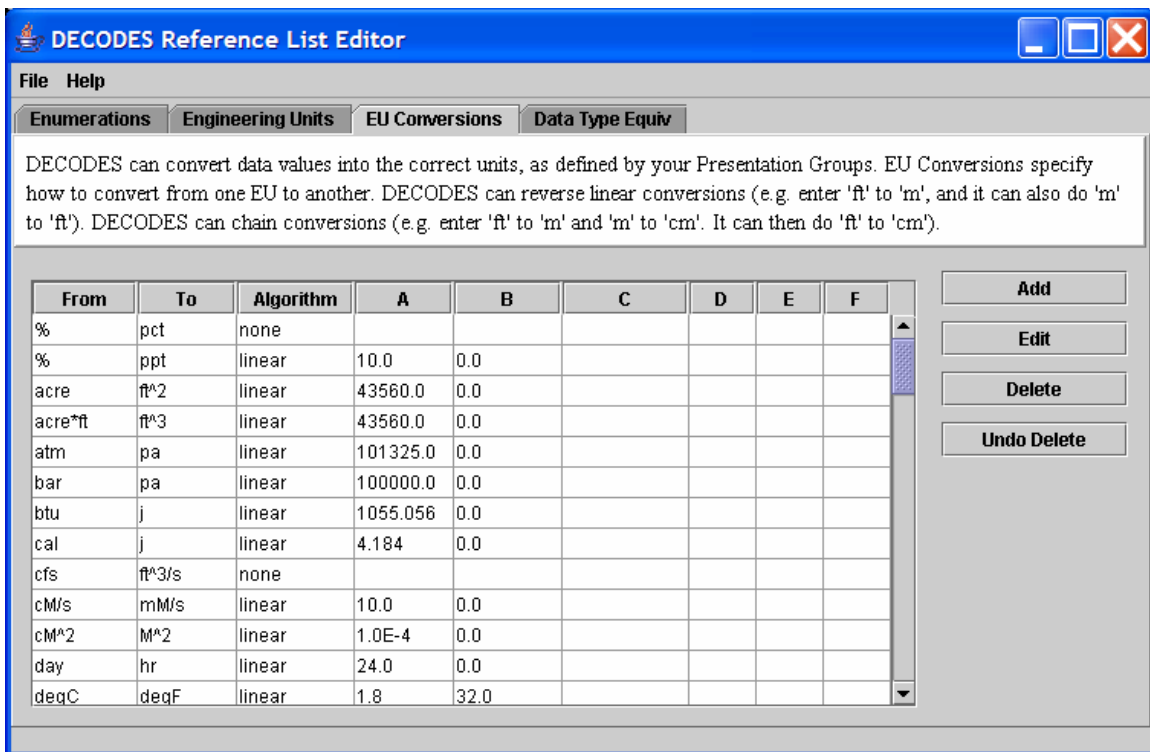


Figure 13-3: Reference List Editor EU Conversions Tab.

This list shows the standard conversions. Each line specifies how to convert *from* one EU abbreviation *to* another. There are four possibilities for Algorithm:

- none – This means that the two units are to be considered synonyms. Examples: % is a synonym for pct, and cfs is a synonym for ft^3/s.
- linear – Uses the equation  $y = Ax + B$ , where  $y$  is the EU we are converting to, and  $x$  is the EU we are converting from.
- usgs – Uses the equation  $y = A * (B + x)^C + D$
- poly-5 – 5<sup>th</sup> order polynomial:  $y = Ax^4 + Bx^4 + Cx^3 + Dx^2 + Ex + F$

To add, edit, or delete, use the buttons to the right of the table.

DECODES can invert “none” and “linear” algorithms. Hence if we specify how to convert from “cal” to “j”, we don’t need to specify how to convert from “j” to “cal”.

DECODES can also combine conversions. Suppose you specify:

- in -> ft
- ft -> m
- m -> mm

Then DECODES can combine these if it needs to convert “in” to “mm”.

### 13.4 Data Type Equivalencies

Figure 13-4 shows the tab for Data Type Equivalencies. Recall that DECODES allows you to specify multiple data type codes for each sensor. So, for a stream stage sensor you might enter HG. Then, if you select USGS-STDMSG for your output format, DECODES must convert HG to the equivalent EPA numeric parameter code.

The equivalencies table allows DECODES to do just that.

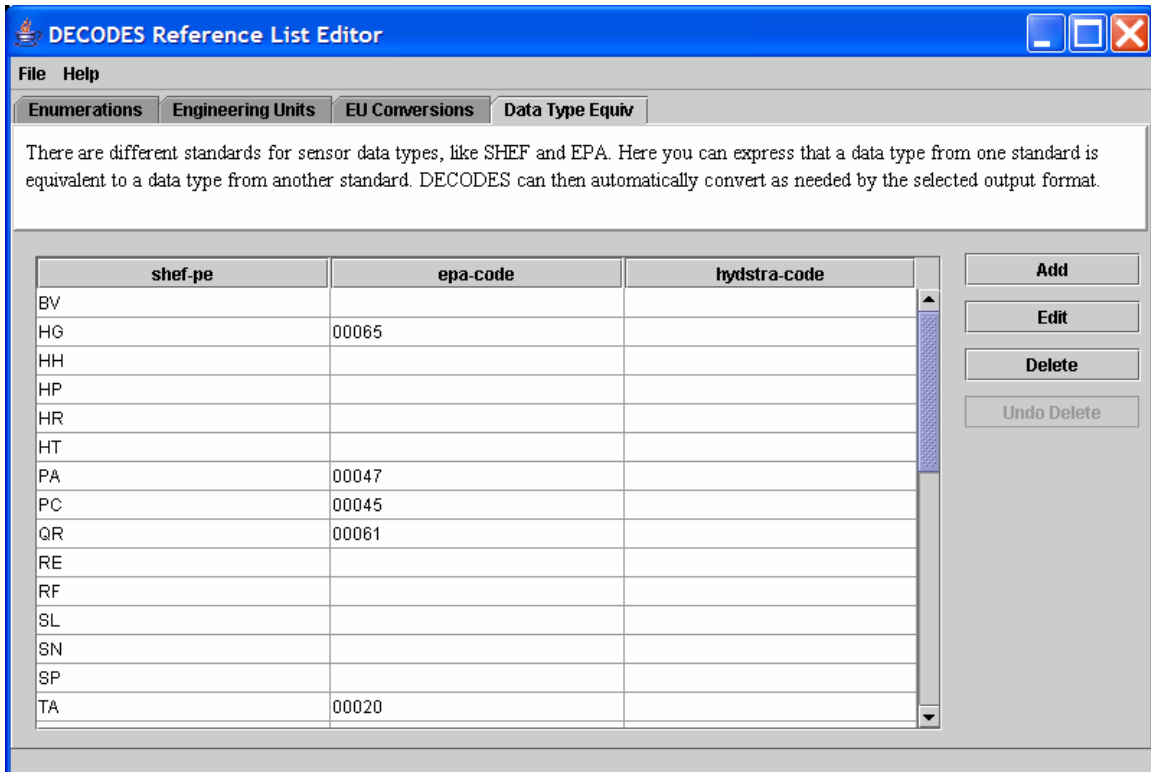


Figure 13-4: Reference List Editor - Data Type Equivalencies Tab.

## 14. Rating Computation using USGS RDB Files

This feature is new in DECODES 6.2. The USGS publishes rating tables in a format called “RDB”. USGS distributes these files to other federal agencies. DECODES can use USGS RDB files to do calculations like:

- Stage to Discharge
- Elevation to Volume

It does the calculation on-the-fly. When a raw message is processed, it produces a time series containing your *independent* variable (e.g. STAGE). The RDB Process can then use these to derive a separate time series containing your *dependent* variable (discharge). The output of the routing spec will contain both.

You will need USGS RDB files with the shifts already calculated. An example is shown in Figure 14-1. Note the inclusion of the SHIFT column in the data. This is necessary.

This example also shows an “expanded” rating, meaning that the table includes the stored values (marked with an asterisk) *and* interpolated values at every .01 increment of the independent variable. DECODES is capable of doing its own logarithmic interpolation between stored points, so you do not need an expanded rating.

```
# //UNITED STATES GEOLOGICAL SURVEY      http://water.usgs.gov/
# //NATIONAL WATER INFORMATION SYSTEM    http://water.usgs.gov/data.html
# //DATA ARE PROVISIONAL AND SUBJECT TO CHANGE UNTIL PUBLISHED BY USGS
# //RETRIEVED: 2003-10-31 10:10:47
# //FILE TYPE="NWIS RATING"
# //DATABASE NUMBER=1 DESCRIPTION=" Standard data base for this site."
# //STATION AGENCY="USGS " NUMBER="05495000 " TIME_ZONE="CST" DST_FLAG=Y
# //STATION NAME="Fox River at Wayland, MO"
# //DD NUMBER=" 7" LABEL="Discharge, in cfs"
# //PARAMETER CODE="00060"
# //RATING SHIFTED="20031031100000 CST"
# //RATING ID="19.0" TYPE="STGQ" NAME="stage-discharge"
# //RATING REMARKS=""
# //RATING EXPANSION="logarithmic"
# //RATING BREAKPOINT1=1.00
# //RATING OFFSET1=0.80 OFFSET2=1.00
# //RATING_INDEP ROUNDING="2223456782" PARAMETER="Gage height IN feet"
# //RATING_DEP ROUNDING="2222233332" PARAMETER="Discharge IN cfs"
# //RATING_DATETIME BEGIN=20021001000000 BZONE=CDT END=23821230090000 EZONE=CST
INDEP      SHIFT      DEP      STOR
16N        16N        16N      1S
0.80       0.12       0.12     *
0.81       0.12       0.13
0.82       0.12       0.14
0.83       0.12       0.15
0.84       0.12       0.16
0.85       0.12       0.17
0.86       0.12       0.18
0.87       0.12       0.19
0.88       0.12       0.20
0.99       0.12       0.44
1.00       0.12       0.49     *
1.01       0.12       0.53
1.02       0.12       0.57
1.03       0.12       0.62
1.04       0.12       0.66
. . .
```

Figure 14-1: USGS RDB Rating File Example

## 14.1 Store the RDB Files in a Known Directory

Set up a directory under the DECODES installation to hold the RDB files. We recommend making a sub-directory called “rdb-files”.

Place your RDB files in this directory as you get them from the USGS. Many agencies are using WGET or RSYNC utilities to make sure they have the latest ratings.

The rating files can have any name. Typically, USGS will give them a name that contains the USGS Station Identifier (a long number from 7 to 15 digits) and the USGS database descriptor number, which uniquely identifies the sensor on that station. For example, the above rating file might be called “05495000-7.rdb”.

## 14.2 Associate RDB Files with Platform Sensors

Next you need to tell DECODES which parameters on which sites are associated with which RDB files. You do this by setting Platform Sensor Properties.

Open dbedit and click on the Platforms tab. Select the desired platform and open it. Select the sensor for the independent variable. For example, if this is a stage (HG) to flow (QR) conversion, the independent variable would be stage (HG). Then press the “Sensor Properties” button.

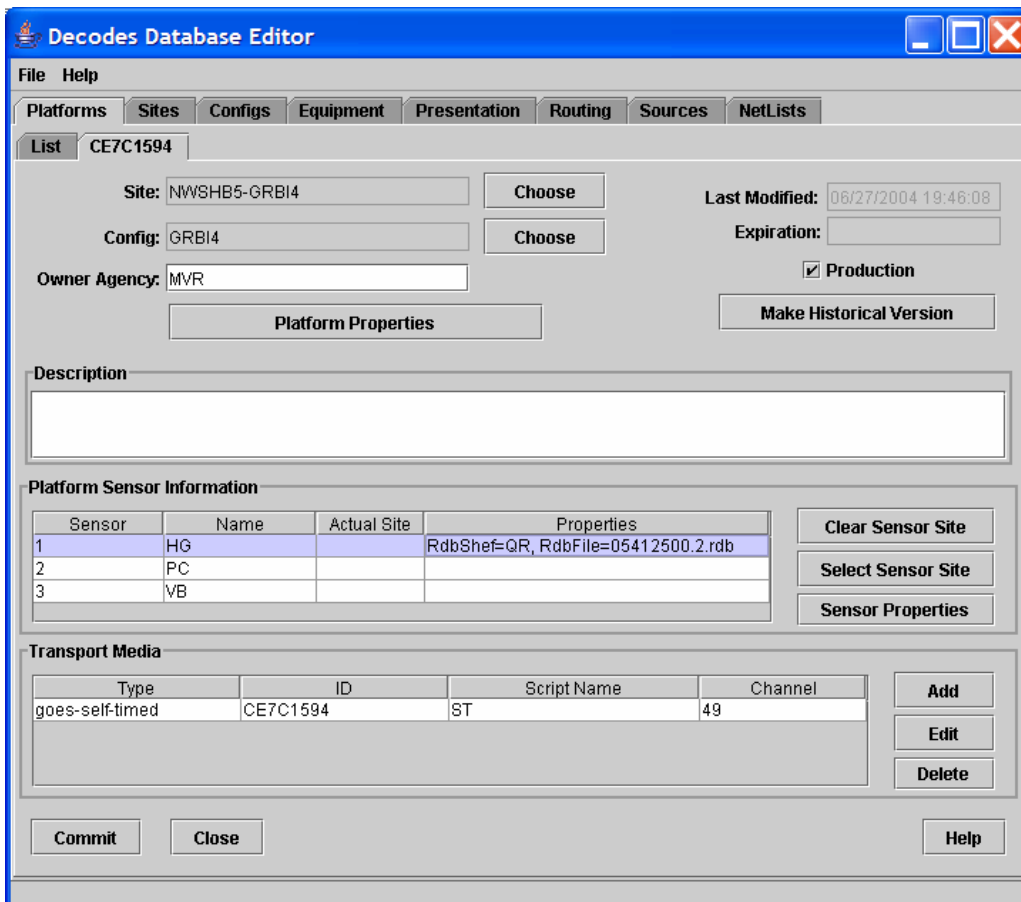
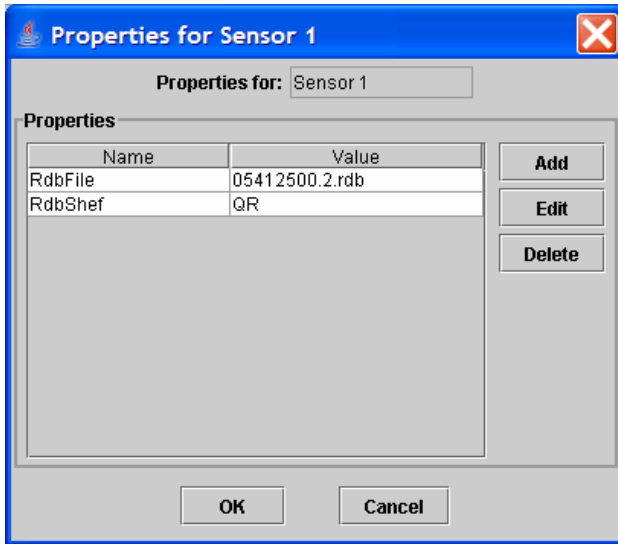


Figure 14-2: Select Platform Sensor and press "Sensor Properties".



**Figure 14-3: Platform Sensor Properties Dialog.**

In this dialog create two properties:

- RdbFile is set to the name of the RDB file to do the conversion. You do not need to specify a complete path name.
- RdbShef is set to the SHEF code for the output (dependent) variable. The example shows QR for river discharge.

### **14.3 Configure the Computation Processor**

The RDB processor is one part of a general purpose module that in the future will be able to perform many types of computations on your data. You configure the computation processor with an XML file that is placed in your DECODES directory. The default name for the file is “computations.conf”. It is included in the DECODES 6.2 release.

There is an entry in this file for each type of computation you want to support. Currently, two types supported. Your file should look much like the following:

```
<ComputationProcessor>
  <CompResolver class="decodes.comp.RdbRatingCompResolver">
    <Property name="dir">$DECODES_INSTALL_DIR/rdb-files</Property>
  </CompResolver>
  <CompResolver class="decodes.comp.TabRatingCompResolver">
    <Property name="dir">$DECODES_INSTALL_DIR/tab-files</Property>
  </CompResolver>
</ComputationProcessor>
```

**Figure 14-4: Example "computations.conf" file.**

This file installs two resolvers: one for RDB files and one for simple table files. For each one you can specify a property called ‘dir’, which is the name of the directory to contain the RDB or table files. The default is to use a subdirectory under the DECODES installation. If this is acceptable to you, then do not make any changes.

## 14.4 Simple ASCII Table Files

Rating can also be done with simple ASCII table files. An example would be the following file, which converts lake elevation into storage in Acre-Feet:

```
# CRVI4 - Elevation vs Storage (Ac-Ft) 28June2004
645, 0
650, 2
660, 100
670, 3120
680, 17720
690, 70730
700, 183710
710, 372680
715, 501670
720, 644470
```

**Figure 14-5: Example of Simple ASCII Table File.**

Just as you did for RDB file, you specify linkages by adding Platform Sensor Properties:

- TabFile – The name of the table file to use on this sensor.
- TabShef – The SHEF code for the output value.
- TabName – The name for the output value.
- TabEU – The engineering units for the output value.

## 14.5 Run Your Routing Spec with Computations Enabled

Simply add the “-c” argument to the command line of your routing spec. For example, if my routing spec is called mvmDSS in the editable database, use this command:

```
rs -e -c mvmDSS
```

## 15. The DECODES Platform Wizard

The Platform Wizard is a Graphical User Interface (GUI) tool for entering or editing your platform meta-data. It guides you step-by-step through the process of entering site, equipment, configuration, and platform data.

Everything that the Platform Wizard does can also be accomplished with the DECODES Database Editor described in Chapter 5, and you may prefer to continue using that tool. The wizard provides a more directed approach.

The platform wizard has nine panels. Each will be described in the following subsections.

- Start Panel
- Site Panel
- Platform Sensors Panel
- Equipment Model Panel
- Decoding Script Panels for self-timed, random, and EDL
- Platform Specific Information Panel
- “Save Your Work” Panel



## 15.1 Platform Wizard Start Panel

When you start the platform wizard, you see the initial panel shown in Figure 15-1. Here you specify what types of messages this platform can generate: GOES Self Timed or Random, or EDL (Electronic Data Logger) files. You also specify the identifying information for the message. For example, the figure shows that we want to create a platform record for a GOES Self Timed DCP with address “CE4816DE” that transmits on channel 73.

If this is a NEW platform, press Next to continue to the next panel.

If this panel already exists in your database, press the “My Editable Database” button to cause the GUI to be initialized with the information you’ve already entered.

Future versions will allow you to initialize the GUI from remote DECODES databases, the National Weather Service HADS System, and from your NEWSID PDT (Platform Description Table) records.

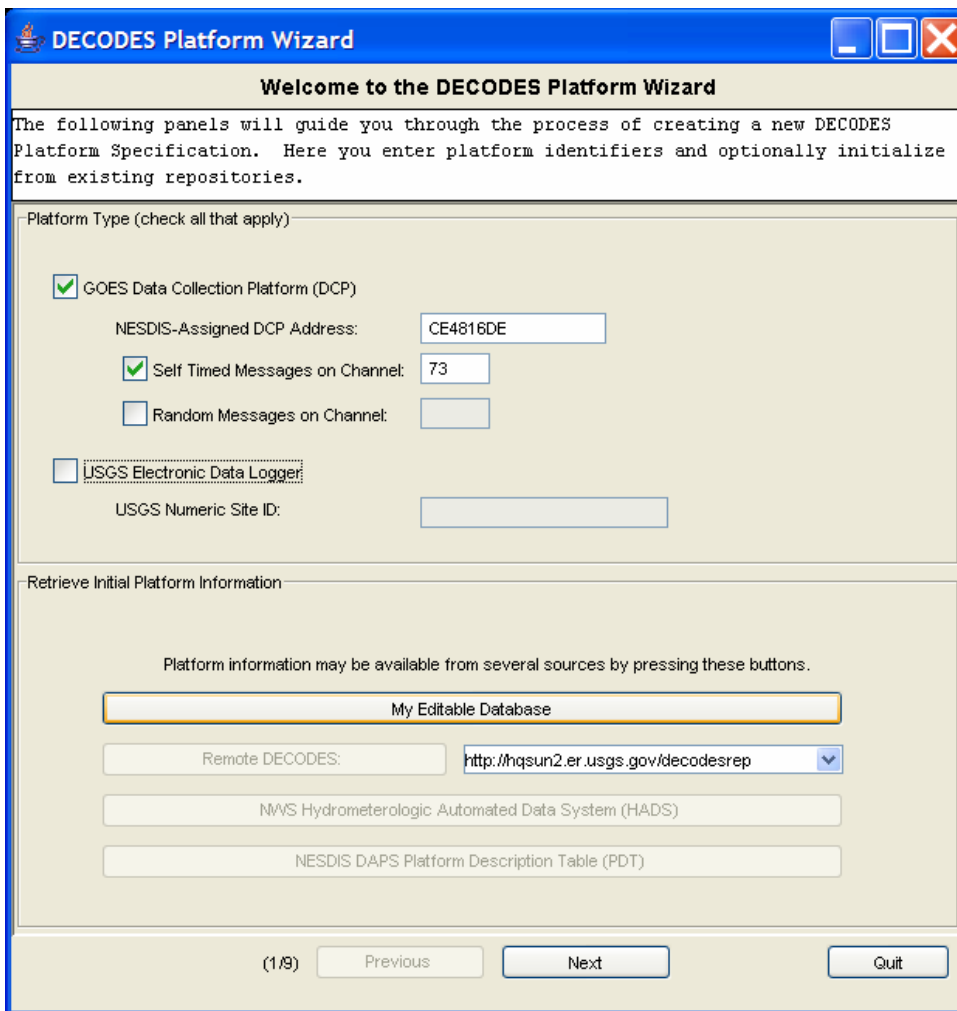


Figure 15-1: Platform Wizard Start Panel.

## 15.2 Platform Wizard Site Panel

The Site Panel is shown in Figure 15-2. Here you enter information about the location. We have also entered the NWSHB5 (National Weather Service Handbook 5) name “VNDI2” for this site.

The screenshot shows the 'DECODES Platform Wizard' window with the 'Site Location Information' tab selected. The window title bar includes standard Windows window controls (minimize, maximize, close). The main content area is titled 'Site Location Information' and contains the following elements:

- A text box with the instruction: "Enter descriptive information about the site location. At least one name must be entered for the site."
- A 'Select Site:' field containing 'local=CE4816DE', with 'Select' and 'New' buttons to its right.
- A 'Names for this Site' table with two columns: 'Type' and 'Name'. It lists two entries: 'nesdis' with name 'CE4816DE' and 'nwshb5' with name 'VNDI2'. Below the table are 'Add Name' and 'Delete Name' buttons.
- A 'Location Information' section with the following fields:
  - Latitude: N 38°57'55"
  - Longitude: W 89°05'20"
  - Elevation: (empty text box)
  - Elev Units: ft (feet) (dropdown menu)
  - Nearest City: VANDALIA
  - Time zone: (dropdown menu)
  - State: IL
  - Country: (empty text box)
  - Region: (empty text box)
- A 'Description, Directions, Etc.' section with a text box containing the text: 'KASKASKIA RIVER AT VANDALIA'.
- Navigation buttons at the bottom: '(2/9)', 'Previous', 'Next', and 'Quit'.

Figure 15-2: Platform Wizard Site Panel.

### 15.3 Platform Wizard Sensors Panel

The next panel, shown in Figure 15-3, allows you to enter information about each sensor on this platform. This Platform has three sensors: Stage, Precip, and Battery. Each reports a value every 15 minutes within a message.

DECODES Platform Wizard

**Define Platform Sensors**

Define the sensors on this platform and add a description.

Platform Configuration: MVSHR-SB4-001    Select    New

Name: MVSHR-SB4-001    Description: 15 min Stage, Precip, & Battery

Equipment Model: MVSHR

Num Platforms: 0

Sensor	Name	Data Type	Mode	Sampling Times
1	STAGE	SHEF-PE:HG	Fixed	00:00:00 15 minutes
2	PRECIP	SHEF-PE:PC	Fixed	00:00:00 15 minutes
3	Battery	SHEF-PE:VB	Fixed	00:00:00 15 minutes

Buttons: Delete Sensor, Edit Sensor, Add a sensor., Add DCP PMs

Navigation: (3/9) Previous Next Quit

Figure 15-3: Platform Wizard Sensors Panel.

## 15.4 Platform Wizard Equipment Model Panel

This panel allows you to enter information for the equipment model.

The screenshot shows a window titled "DECODES Platform Wizard" with a sub-panel "Select Equipment Model". The window contains the following elements:

- Instructions: "If you want to use an existing equipment model and platform config, select them here. Otherwise, select '<new>'."
- Form fields: "Select Equipment Model:" (text box with "MVSHR"), "Name:" (text box with "MVSHR"), "Type:" (dropdown menu with "DCP"), "Company:" (text box with "Handar"), and "Model:" (text box with "570").
- Buttons: "Select" and "New" next to the "Select Equipment Model:" field.
- Description: A large text area labeled "Description" which is currently empty.
- Properties: A table with columns "Name" and "Value".

Name	Value
PlatformType	DCP
RetrievalMethod	DCP

Buttons "Add", "Edit", and "Delete" are located to the right of the table.
- Navigation: "(4/9)", "Previous", "Next", and "Quit" buttons at the bottom.

Figure 15-4: Platform Wizard Equipment Model Panel.

## 15.5 Platform Wizard Decoding Script Panel

Refer back to the start panel in Figure 15-1. There you specified up to three ways to get data from this platform: GOES Self-Timed, GOES Random, and EDL. You will be presented with a separate Decoding Script Panel for each of the three message types. In Figure 15-5 we see the panel for GOES Self-Timed Messages.

The “Load” button along the right makes it easy to retrieve a message of the specified type. This is then shown in the “Sample Message” area. Press the “Decode” button to apply your format statements to the sample message. The results are shown in the “Decoded Data” area at the bottom.

The figure shows a simple ASCII DCP message with 8 fifteen-minute samples for each of the three sensors. Notice how the format statement makes use of the ‘w’ operator and delimited lengths in the ‘F’ field operators. This allows us to correctly handle the 4<sup>th</sup> data line, where an extra space appears before the stage value, and the battery value is only 4 characters long.

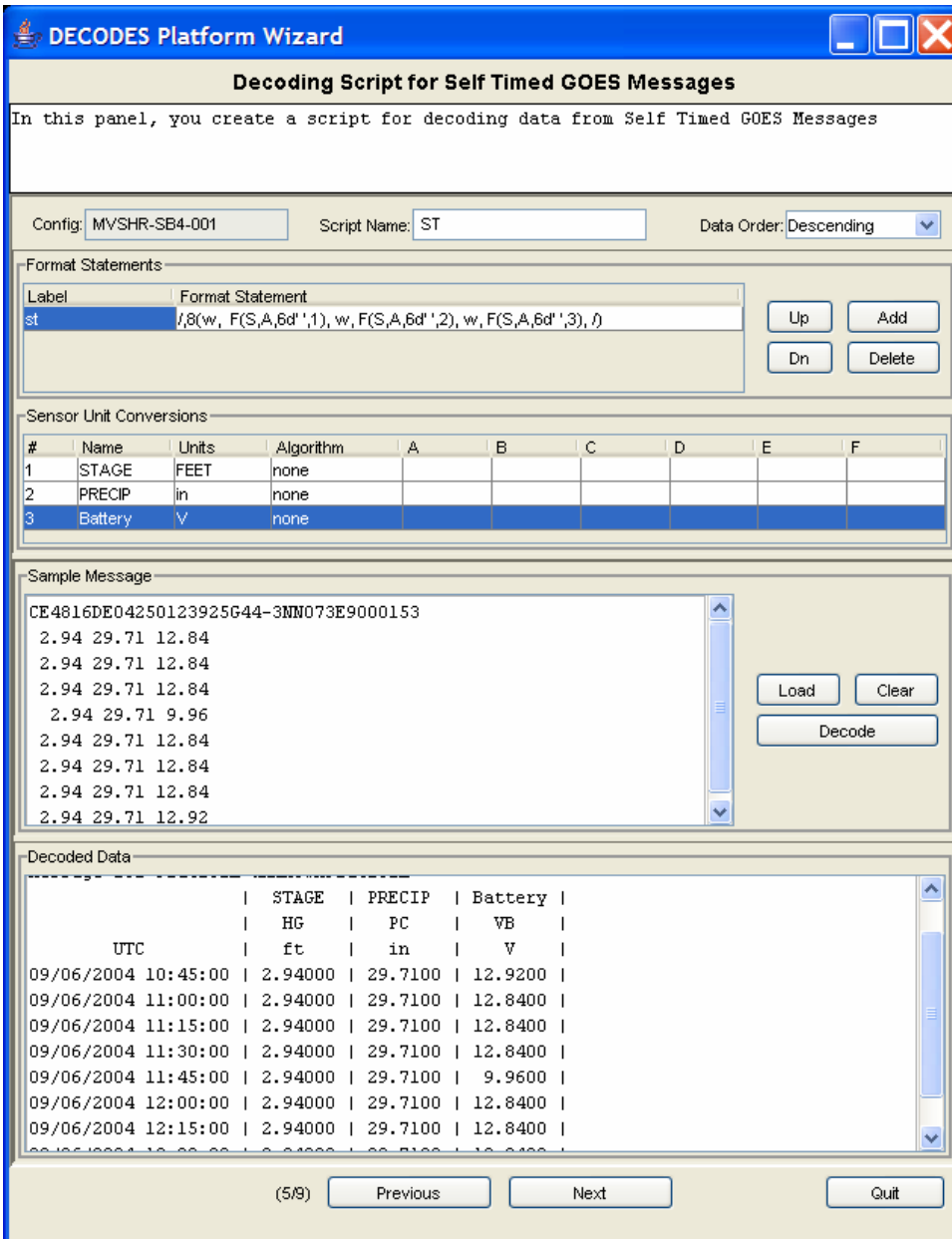


Figure 15-5: Platform Wizard Decoding Script Panel.

## 15.6 Platform Specific Information

The next panel, shown in Figure 15-6 allows you to enter platform-specific information. Make sure that you have one “Transport Medium” record at the bottom for each of the 3 ways of retrieving data. Our example shows a single record for GOES Self Timed.

**Platform-Specific Information**

Enter platform-specific information in this panel, including a description, sensor properties, and the specifics of your transport media.

Site:

Config:  Last Modified:

Owner Agency:  Expiration:

Description

Platform Sensor Information

Sensor	Name	Actual Site	Properties
1	STAGE		
2	PRECIP		
3	Battery		

Transport Media

Type	ID	Script Name	Channel
goes-self-timed	CE4816DE	ST	73

(8/9)

Figure 15-6: Platform Wizard "Platform Specific Info" Panel.

## 15.7 Save Your Work

The final panel, shown in Figure 15-7, allows you to save your work to the editable database, or to a separate XML file.

First press the “Validate Platform” button. If there are any inconsistencies in the data you entered, this will be explained on the screen. You will be directed back to other panels to correct the errors. When finished, return to this panel and press the Validate button again.

The example screen shows that our sample platform validated successfully. So we pressed the “Write to Editable Database” button. This was also successful.

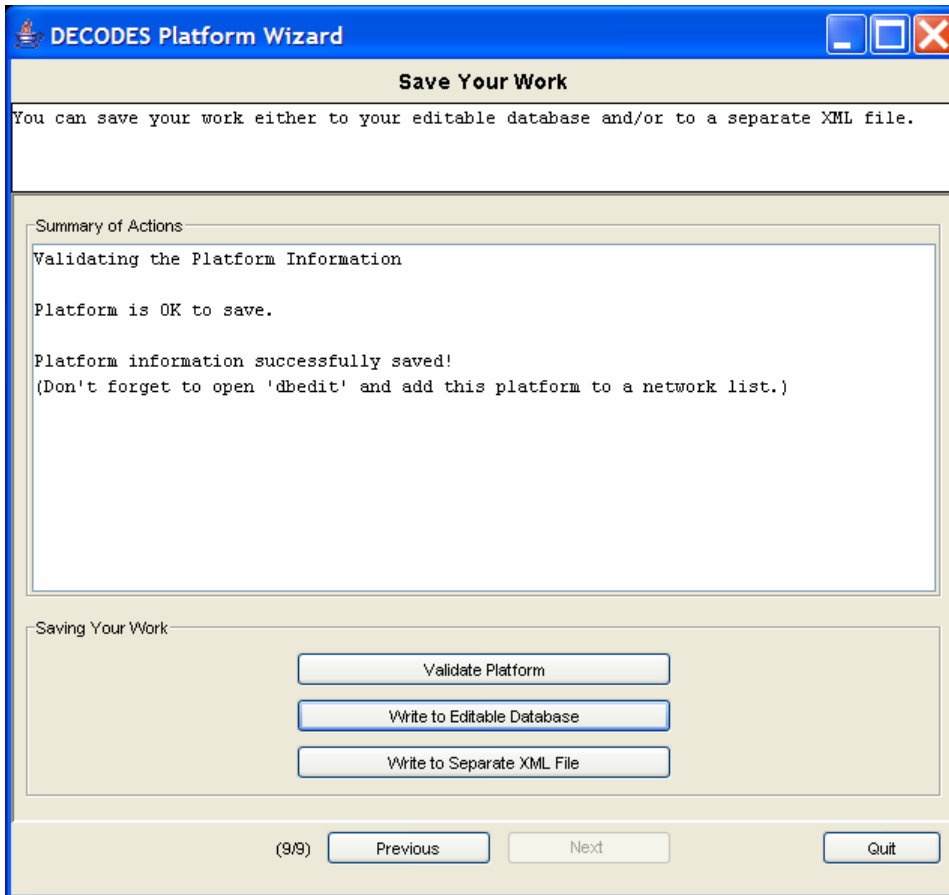


Figure 15-7: Platform Wizard "Save Your Work" Panel.

## 16. The Interactive Decoding Wizard

As of Version 7, DECODES contains a tool for interactive decoding of files. This is most useful for EDL (Electronic Data Logger) files, but may also be used for files containing GOES DCP messages.

Start the Decoding Wizard from the shortcut provided or by typing the command 'decwiz' at the command prompt.

By default the decoding wizard will use your production database, if one is present. To force it to use the edit-database, add the `-e` option to the command line:

```
decwiz -e
```

The Decoding Wizard has three panels, shown in the following three figures.



## 16.1 The File Scanning Panel

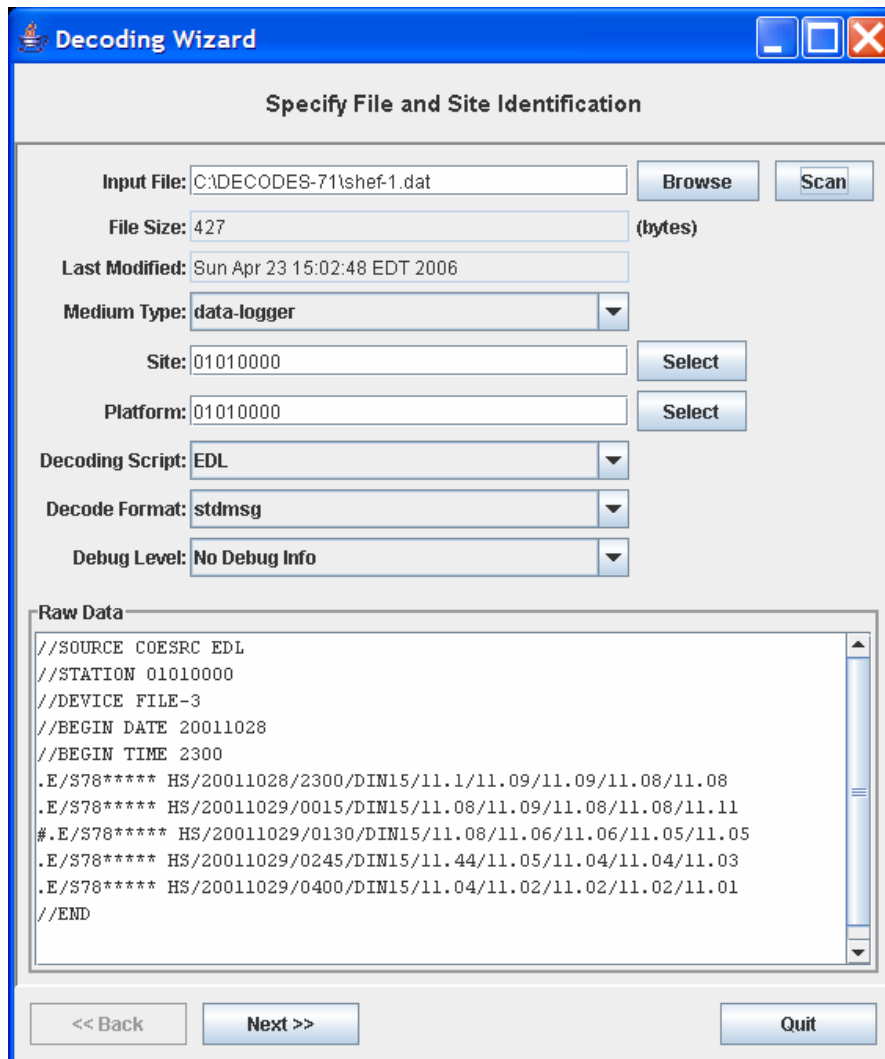
First select a file to decode by typing the name in the 'Input File' area, or by pressing the 'Browse' button. Then press the 'Scan' button.

The wizard loads the file into the 'Raw Data' area at the bottom of the screen. It also shows the file size in bytes and the last-modify time of the file.

It then scans the header of the file, trying to associate it with a platform in your database. If successful, it will fill in the 'Medium Type', 'Site', 'Platform', and 'Decoding Script' fields. It might *not* be successful if the header is completely or partially missing, or if it can not find a matching record in your database. If this happens, you may choose the site, platform, and script records manually.

You may also select the decoding format you would like to use (e.g. 'stdmsg' or 'Human-Readable'), and the verbosity level for debugging information.

After you have done this, press the 'Next >>' button at the bottom of the screen.



The screenshot shows the 'Decoding Wizard' window with the following fields and buttons:

- Input File:** C:\DECODES-71\shf-1.dat (with 'Browse' and 'Scan' buttons)
- File Size:** 427 (bytes)
- Last Modified:** Sun Apr 23 15:02:48 EDT 2006
- Medium Type:** data-logger (dropdown menu)
- Site:** 01010000 (with 'Select' button)
- Platform:** 01010000 (with 'Select' button)
- Decoding Script:** EDL (dropdown menu)
- Decode Format:** stdmsg (dropdown menu)
- Debug Level:** No Debug Info (dropdown menu)

The 'Raw Data' section contains the following text:

```
//SOURCE COESRC EDL
//STATION 01010000
//DEVICE FILE-3
//BEGIN DATE 20011028
//BEGIN TIME 2300
.E/S78***** HS/20011028/2300/DIN15/11.1/11.09/11.09/11.08/11.08
.E/S78***** HS/20011029/0015/DIN15/11.08/11.09/11.08/11.08/11.11
#.E/S78***** HS/20011029/0130/DIN15/11.08/11.06/11.06/11.05/11.05
.E/S78***** HS/20011029/0245/DIN15/11.44/11.05/11.04/11.04/11.03
.E/S78***** HS/20011029/0400/DIN15/11.04/11.02/11.02/11.02/11.01
//END
```

At the bottom of the window are three buttons: '<< Back', 'Next >>', and 'Quit'.

Figure 16-1: Decoding Wizard - File Scanning Panel.

## 16.2 The Decoding and Time Shift Panel

Press the 'Decode Data' button at the top. The wizard will use the platform, site, and script records that you selected to decode the raw data. The results will be shown in the format that you selected on the previous screen.

After decoding, if the results are not what you expect, press the 'Trace Log' button. A separate dialog is then displayed showing you all of the steps that DECODES went through to decode the raw data.

The 'Decoding Summary' area shows a summary of the decoding information. For each sensor it shows several statistics including start/end times, number of samples found, number of various types of errors, etc. A separate line is printed if an 'excessive' gap is found in the data. 'Excessive' means more than the maximum number of missing samples specified at the top of the screen.

In the bottom area, you may specify a time shift if necessary. Specify the beginning and ending time of the shift in the time-format shown, and the amount of the shift. Then press the 'Decode Data' button again to see the results of the shift.

When you are satisfied with the results, press the 'Next >>' button at the bottom of the screen.

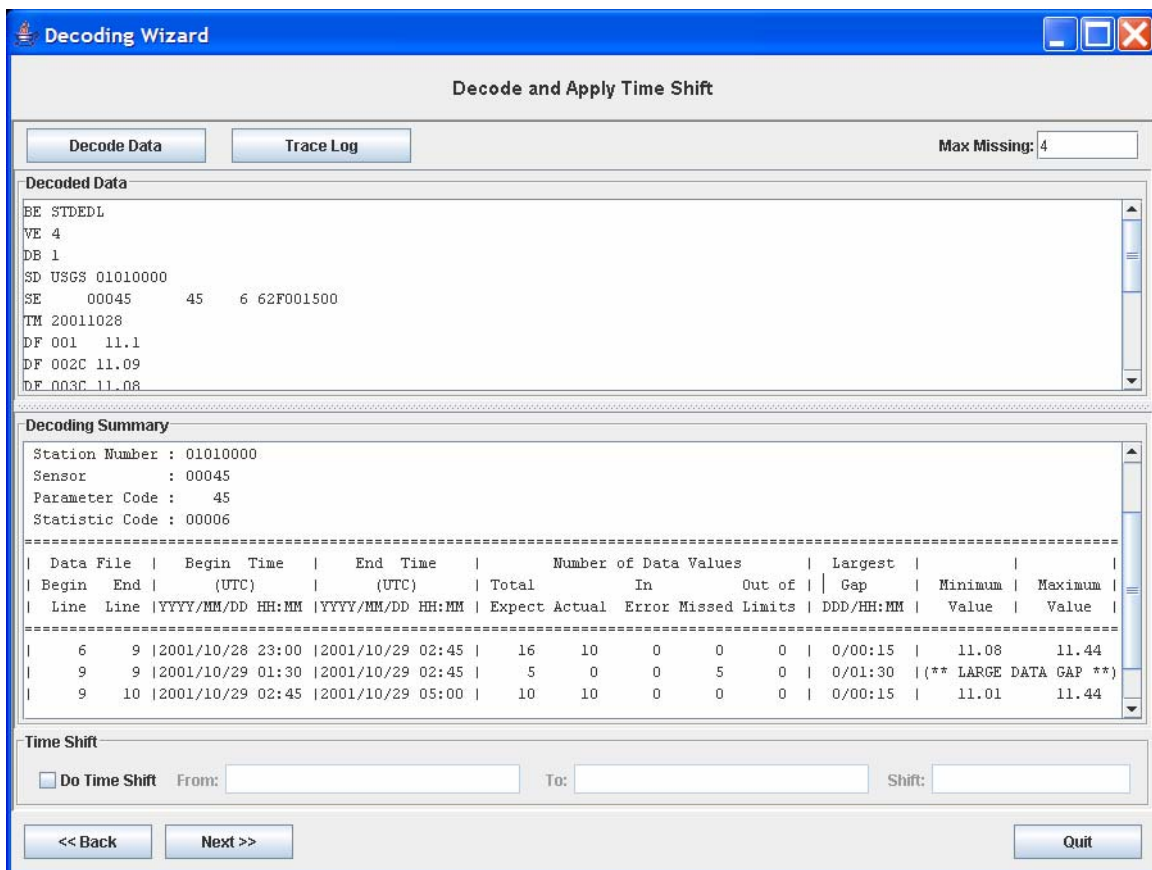


Figure 16-2: Decoding Wizard - Decoding and Time Shift Panel.

### 16.3 The Save-Results Panel

On this panel you may separately save the raw data, decoded data, and summary to different files. Leave the fields blank if you do not want to save. In the pull-down at the right, you may choose to overwrite or append-to the selected file.

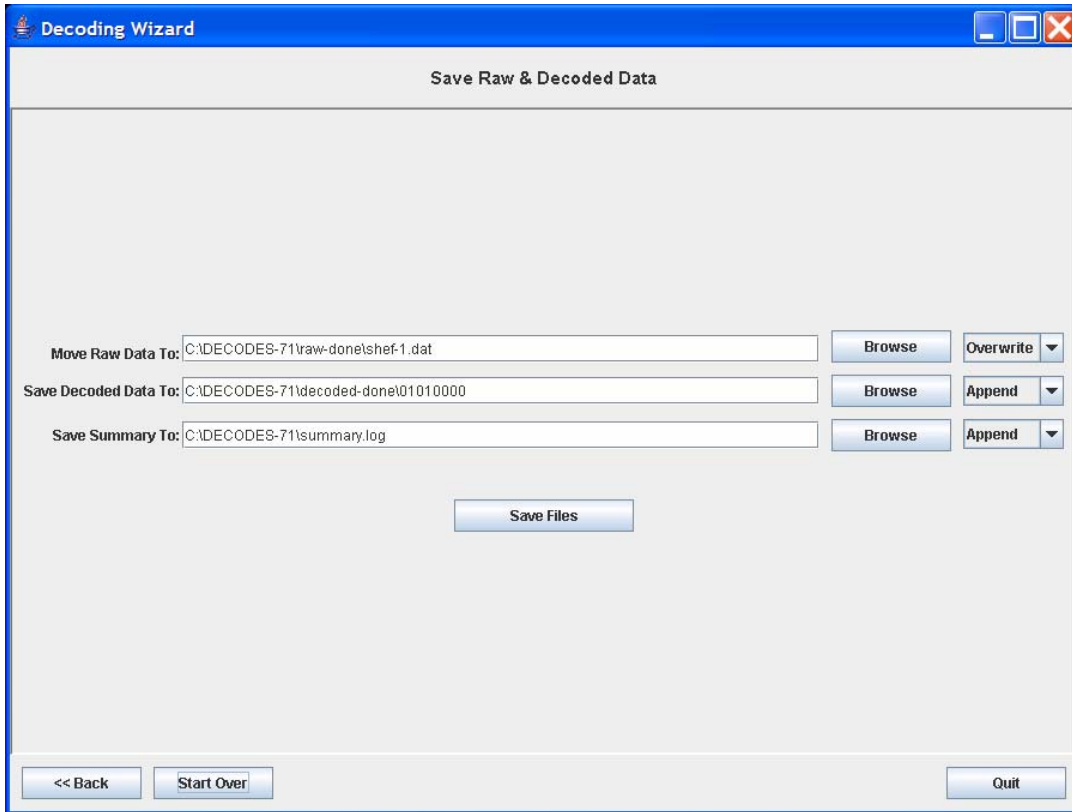


Figure 16-3: Decoding Wizard - Save Results Panel.

## 17. USGS National Water Information System (NWIS) Integration

As of version 7, DECODES can be integrated with an NWIS database. This works as follows:

- You set the Editable database type and location to point to your Ingres NWIS database.
- All of the DECODES meta-data is stored in tables within NWIS.
- Site records use information from the NWIS “SITEFILE” table.

### 17.1 Configure DECODES for your NWIS Database

First, you must set the edit database values in your “decodes.properties” file:

```
EditDatabaseType=NWIS
EditDatabaseLocation=nwis://hostname/dbname#dbnum
```

- *hostname* is the IP address or host name of the database server
- *dbname* is the logical Ingres database name.
- *dbnum* is the default database number

Then you must create an encrypted file in your home directory containing your NWIS login name and password. This file should be protected so that only you can access it in any way. Here is the recommended way to create the file:

```
cd $HOME
touch .nwis.auth
chmod 600 .nwis.auth
```

Now edit the file with the DECODES command “setNwisUser”. You will be asked for username and password. These will be encrypted and stored in the .nwis.auth file.

## 17.2 Initializing NWIS for DECODES

At the time of this writing, the stock NWIS installation needs to have some additions before it will work with DECODES. These may be corrected in future releases of NWIS.

**Add Surrogate Key Rows for DECODES Tables.** This is done by executing the following SQL commands with the appropriate GRANT permissions:

```
insert into surrogate values('enum_id', -2147483647, '');
insert into surrogate values('equipmentmodel_id', -2147483647, '');
insert into surrogate values('datatype_id', -2147483647, '');
insert into surrogate values('platform_id', -2147483647, '');
insert into surrogate values('platformconfig_id', -2147483647, '');
insert into surrogate values('decodesscript_id', -2147483647, '');
insert into surrogate values('routingspec_id', -2147483647, '');
insert into surrogate values('datasource_id', -2147483647, '');
insert into surrogate values('networklist_id', -2147483647, '');
insert into surrogate values('presentationgroup_id', -2147483647, '');
insert into surrogate values('datapresentation_id', -2147483647, '');
insert into surrogate values('unitconverter_id', -2147483647, '');
```

**Import the standard ‘Reference List’ data into the DECODES tables.** In the DECODES installation directory, you can run the following commands:

```
bin/dbimport edit-db/enum/EnumList.xml
bin/dbimport edit-db/eu/EngineeringUnitList.xml
bin/dbimport edit-db/datatype/DataTypeEquivalenceList.xml
```

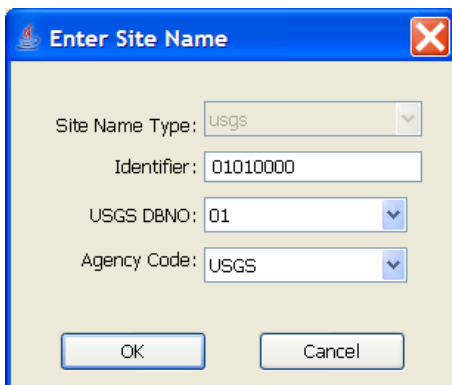
You can make more customizations to the reference lists with the Reference List Editor, as described in section 13.

## 17.3 NWIS Mapping for Sites and Site Names

Most of the DECODES meta-data is simply stored in NWIS as it would be in any other database. However, ‘Site’ records needed to be modified to conform to the NWIS implementation.

Most of the Site meta-data is stored in the NWIS SITEFILE\_## table. This includes the latitude, longitude, and other descriptive information.

DECODES ‘SiteName’ records have been enhanced to account for NWIS rules. When you add or edit a site name record, you now see the dialog shown in Figure 17-1.



**Figure 17-1: Site Name Edit Dialog.**

For SiteName records with type 'USGS', the following rules apply:

- The 'Identifier' is the USGS Site Number (typically 8 to 15 digits).
- The USGS DBNO is the Database Number containing this site record.
- The Agency code specifies which agency owns this site record.

## 17.4 Additional NWIS Sensor Parameters

NWIS requires you to specify two additional parameters for each sensor:

- Statistics Code
- Database Descriptor Number (DDNO)

Decodes stores the Statistics Code in the Platform Configuration record. Go to the 'Configs' tab and open the desired configuration. Select a sensor and press the 'Edit' button. You see the dialog as shown in Figure 17-2. Enter the USGS Stat Code here.

The 'Edit Config Sensor' dialog box contains the following fields and controls:

- Configuration: shef-1
- Sensor: 1
- Sensor Name: 00045
- Standard: epa-code (dropdown)
- Code: 00045
- Data Types: Standard: shef-pe (dropdown)
- Code: (empty)
- Standard: hydstra-code (dropdown)
- Code: (empty)
- USGS Stat Code: 00006
- Valid Range - Min: (empty)
- Max: (empty)
- Recording Mode: Fixed (dropdown)
- 1st Sample Time: 00:00:00 (HH:MM:SS)
- Sampling Interval: 00:15:00 (HH:MM:SS)
- Equipment Model: (empty) [Select]
- Properties table:

Name	Value

[Add] [Edit]
- [OK] [Cancel]

Figure 17-2: Edit Config Sensor Dialog.

The DDNO is site-specific. We enter this in the Platform Sensor record. Go to the 'Platforms' tab and open the desired platform. Select a sensor and press the 'Edit Sensor Info' button. You see the dialog as shown in Figure 17-3. Enter the USGS DDNO for this sensor here.

If you are connected to an NWIS database, the pull-down list should show you all valid DDNOs for this sensors parameter code at this site. If you are working from an XML or non-NWIS database, the pull-down list will be empty and you must type the value directly.

**Platform Sensor Parameters**

Sensor Number: 1      Sensor Name: 00045      Param Code: 00045

Actual Site: (inherited)      Select      Clear

Min and Max defined here override min/max in configuration.

Platform Specific Min:      Config Min:      Platform Specific Max:      Config Max:

USGS DDNO: 06

**Additional Properties**

Name	Value
------	-------

Add      Edit      Delete

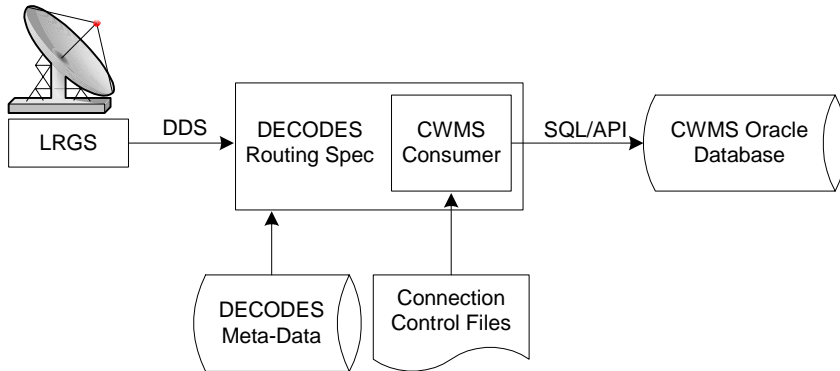
OK      Cancel

**Figure 17-3: Edit Platform Sensor Dialog.**



## 18. USACE CWMS Interface

The USACE (U.S. Army Corps of Engineers) CWMS (Corps Water Management System) stores uses a time-series database to store water-level and related data. The DECODES software suite has a module allowing it to place incoming data directly into the CWMS database. This obviates the need for intermediate flat-files used in the pass.



**Figure 18-1: CWMS Interface Data Flow.**

The “CWMS Consumer” is selected as the output (consumer) module in the routing spec. It receives the decoded data from the routing spec and stores it in the CWMS Oracle Database. The CWMS Consumer uses the new SQL API (Application Program Interface) published by HEC to write time-series data directly to CWMS.

We have tried to make the CWMS Consumer as automatic as possible. It has hard-coded defaults for time-series storage parameters that can be used in most cases. For special cases, the consumer allows you to provide properties in the DECODES database to override the defaults.

Control files tell DECODES how to connect and authenticate to the CWMS database.

This section will explain how to set up DECODES to store data directly into a CWMS database.

## 18.1 What You Will Need

In order to use the CWMS consumer, you must have a working CWMS database with the 1.4 HEC CWMS API Installed. You must have a valid CWMS username & password with permission to write time series data.

The Consumer uses the “STORE\_TS” PL/SQL stored procedure to store the data. Refer to the CWMS Oracle API User’s Manual for more information about the “STORE\_TS” procedure.

Oracle provides a JDBC driver that you will need. At the time of this writing, the latest version is a file called “ojdbc14.jar”. By default the CWMS Data Consumer uses the Oracle “thin” driver which does not require additional libraries besides the jar file. We have tested the consumer and verified that it works with the “thin” driver.

*If for special reasons, you are required to use the “OCI” driver, then you will need additional native library files. You must install these files in the machine where DECODES will run. The native libraries are required for the Oracle JDBC OCI driver. Refer to the Oracle web site (<http://www.oracle.com>) to find the correct “oci” Driver libraries for your operating system.*

You will also need to add this Jar file to your CLASSPATH variable. For example, on a UNIX system, if you place the file in /usr/local/lib, then in your shell startup file (.profile, .bash\_profile, etc.) place the lines:

```
CLASSPATH=/usr/local/lib/ojdbc14.jar:$CLASSPATH
export CLASSPATH
```

On a windows system use the System control panel to set CLASSPATH.

## 18.2 Set up DECODES for CWMS

CWMS requires some additions to the DECODES Database:

- New “CWMS” Site Name Type
- New “CWMS” Parameter Data Type
- Several Engineering Units (CWMS is very particular about what EUs it will accept).
- Unit Converters to convert from other DECODES units to the ones that CWMS recognizes.
- A new “CWMS” Data Consumer Type
- A “Null” Output Formatter

We have prepared an XML file containing these items. To import these items into your DECODES database, open a terminal window. Then CD to the DECODES\_INSTALL\_DIR directory. Then:

```
bin/dbimport -r to_import/cwms-import.xml
```

(If you are working on a Windows machine, substitute backslash for slash in the above).

## 18.3 CWMS Connection Parameters

Two files are required: A Properties file stores the CWMS connection and default parameters. An encrypted file stores the username and password to use when connecting to CWMS.

### 18.3.1 The CWMS Properties File

Create a text file in the \$DECODES\_INSTALL\_DIR called “decodes-cwms.conf”. This is a text file containing ‘name=value’ pairs, one per line. Table 18-1 explains the parameters, whether or not they are required, and what the default value is. The parameter name is *not* case sensitive.

<i>Name</i>	<i>Default Value</i>	<i>Description</i>
TimeZone	GMT	Optional time zone in which CWMS Database will represent date/time stamps.
dbUri	No default value provided	Required parameter in the form: host:portnumber:SID This tells DECODES the location of the CWMS database. Example: 155.76.210.137:1521:MVRT)
jdbcOracleDriver	jdbc:oracle:thin:@	Optional Oracle JDBC Driver String. The default driver is “thin” but you can change it to “oci”. If “oci” is used native code will have to be installed. No need to modify this property.
cwmsVersion	Raw	Optional: This is used as the default “Version” part of the time-series descriptor.
cwmsOfficeId	No default value provided	Required: This is the CWMS office ID passed to the API “store_ts” procedure. Typically this is your 3-character district abbreviation. Example: MVR
DbAuthFile	\$DECODES_INSTALL_DIR/ .cwmsdb.auth	Optional: Set this if you want to stored the database authentication file in a different location.
shefCwmsParamFile	\$DECODES_INSTALL_DIR/ shefCwmsParam.prop	Optional: Set this if you want to store the SHEF to CWMS mapping in a different file.

**Table 18-1: CWMS Connection Parameters.**

### 18.3.2 Encrypted Username/Password File

The CWMS Consumer will look for a file called “.cwmsdb.auth” in the directory \$DECODES\_INSTALL\_DIR. This file will contain the needed login information in an encrypted form.

A script called “setCwmsUser” has been prepared to facilitate creating or modifying the file. This script must be run in a terminal session:

```
cd $DECODES_INSTALL_DIR
bin/setCwmsUser
    (enter username & password when prompted).
chmod 600 .cwmsdb.auth
```

If this is a Windows system, open a DOS (“cmd”) window and type:

```
cd %DECODES_INSTALL_DIR%
bin\setCwmsUser
```

The program will ask you for a username and password. These will be encrypted and stored in the file.

After creating the file for the first time, you should set its permissions so that only you have access to it:

```
chmod 600 .cwmsdb.auth
```

**Note:** *The file should be owned by the user who will run the DECODES routing spec. The routing-spec will need permission to read this file.*

### 18.3.3 Optional CWMS Parameter Mapping File

DECODES must build a time-series descriptor that contains a valid CWMS “Parameter Type”. Since most of the Corps is currently using DECODES with SHEF codes, we have provided a way to automatically map SHEF codes to CMWS Parameter Types.

**Note:** *See section 18.4.1 for a more complete description on how DECODES builds the descriptor. You can specify CWMS data-types directly in the DECODES database, bypassing SHEF altogether.*

DECODES can do the mappings listed in Table 18-2 automatically. If these are sufficient for you, then you do not need to create a mapping file.

<i>SHEF Code</i>	<i>CWMS Param Type</i>
PC	Precip
HG	Stage
HP	Stage-Pool
HT	Stage-Tail
VB	Volt
BV	Volt
HR	Elev
LF	Stor
QI	Flow-In
QR	Flow
TA	Temp-Air
TW	Temp-Water
US	Speed-Wind
UP	Speed-Wind
UD	Dir-Wind

**Table 18-2: Built-in SHEF to CWMS Parameter Code Mapping.**

If the above defaults are *not* adequate, you may provide a mapping file to override or supplement them. Prepare a text file “shefCwmsParam.prop” and place it in \$DECODES\_INSTALL\_DIR. This is a Java properties file, containing name=value pairs, one per line. For example, to have SHEF “HP” map to CWMS Param Type “Stage”, add a line as follows:

```
HP=Stage
```

## 18.4 How DECODES Uses the CWMS API

DECODES uses a stored procedure in the API called “STORE\_TS”. This procedure requires several arguments to be passed. This section will explain how DECODES determines these arguments.

### 18.4.1 The CWMS Time Series Descriptor

A CWMS Time-Series descriptor has six parts. Each part is separated with a period:

*Location . Param . ParamType . Interval . Duration . Version*

We have designed the DECODES CWMS Consumer for convenience and flexibility: For convenience, DECODES can build the descriptor automatically, using information that it already has in the DECODES database. For flexibility, you can explicitly set part or all of the descriptor in special circumstances.

The following subsections describe each part of the descriptor.

#### 18.4.1.1 Location

The *Location* corresponds to a DECODES site name. DECODES allows each site to have multiple names of different types (see section 2.2.4). It also allows each site to specify which name-type to use by default (see the “SiteNameTypePreference” parameter in Table 3-1).

So, if you have CWMS set up with the same names that you use in DECODES, then you do not need to do anything else.

The consumer will build the location as follows:

- If a site-name with type “CWMS” exists, use it.
- Otherwise, use the default site name.

See section 18.4.3 below for instructions on creating an explicit CWMS site-name-type.

#### 18.4.1.2 Param

The ‘Param’ part must exactly-match one of the CWMS parameter names given in section 7 of the CWMS Oracle API Draft.

You can specify an explicit “CWMS” data-type that will be used here. For instructions on doing this see section 18.4.3 below.

If now CWMS data-type is specified, then the Consumer will attempt to map it from a SHEF code. The consumer will use the mapping specified in the file described in section 18.3.3, or a default mapping listed in Table 18-2 if the SHEF code is not found in the file.

### **18.4.1.3 ParamType**

By default the consumer will set *ParamType* to “Inst”. You can override this by adding a sensor property to the DECODES database called “CwmsParamType”.

Set a Config Sensor Property if you want the value to be applied to all platforms using shared configuration. Use a Platform Sensor Property to apply the value to a single platform.

Other valid settings for ParamType include: “Ave”, “Max”, “Min”, or “Total”.

### **18.4.1.4 Interval**

The *Interval* part specifies the period at which this parameter is measured. DECODES already has this information in each sensor record. It will build the appropriately-formatted string.

### **18.4.1.5 Duration**

The *Duration* part should be “0” for data with a ParamType of “Inst”. DECODES will handle this automatically. For other types (specified by a sensor property), DECODES will build a duration string matching the sensor period. The user can override this choice by adding a sensor property called “CwmsDuration”.

### **18.4.1.6 Version**

The *Version* is usually set to a constant value. Specify this value in the “cwmsVersion” parameter as described in section 18.3.1, or just use the default of “Raw”.

If you need to override the Version setting for particular parameters, add a sensor property called “CwmsVersion” containing the desired value.

## **18.4.2 The CMWS Office ID**

The value for the CWMS office ID is set in the CMWS properties file. See Table 18-1.

You can also specify this as a routing-spec property called “CwmsOfficeId”. This gives you flexibility: The properties file can contain the default. Individual routing specs may override the default if they process data from another office.

## **18.4.3 The “Store Rule”**

The store rule value it is used by the STORE\_TS procedure to control how to handle the insertion of data samples that already exist in the CWMD database.

By default, the consumer will set the store rule to “Replace All”. You may override this by adding a routing-spec property with the desired setting. The valid values are:

- Replace All
- Delete Insert
- Replace With Non Missing
- Replace Missing Values Only
- Do Not Replace

Refer to the API User’s Manual for more information on the store rule field.

#### 18.4.4 Override Protection

This value determines how CWMS will override existing data in the database. By default, the consumer sets this to 1 (true). To set it to false (0), add a routing-spec property called “OverrideProt” set to a value of “0”.

Refer to the API User’s Manual for more information on the override protection field.

#### 18.4.5 Version Date

NOT USED ON CURRENT CWMS DATABASE. Default value is null. Refer to the CWMS Oracle API User’s Manual for more information on this field

### 18.5 Create the Routing Spec

Open the DECODES database editor and create a new routing spec in the normal manner. For Consumer Type, select “cwms”. For Output Format, select “null”.

As stated above, the properties shown in Table 18-3 may be used to override the built-in defaults. Property names are *not* case-sensitive.

Name	Description
CwmsOfficeId	Overrides setting in decodes-cwms.conf file.
StoreRule	Overrides built-in default of “Replace All”
OverrideProt	Overrides built-in default of 0 (false). Set to 1 for true.
VersionDate	NOT USED ON CURRENT CWMS DATABASE VERSION. Default value null. Refer to the CWMS Oracle API User’s Manual for more information.

**Table 18-3: CWMS Routing Spec Properties.**

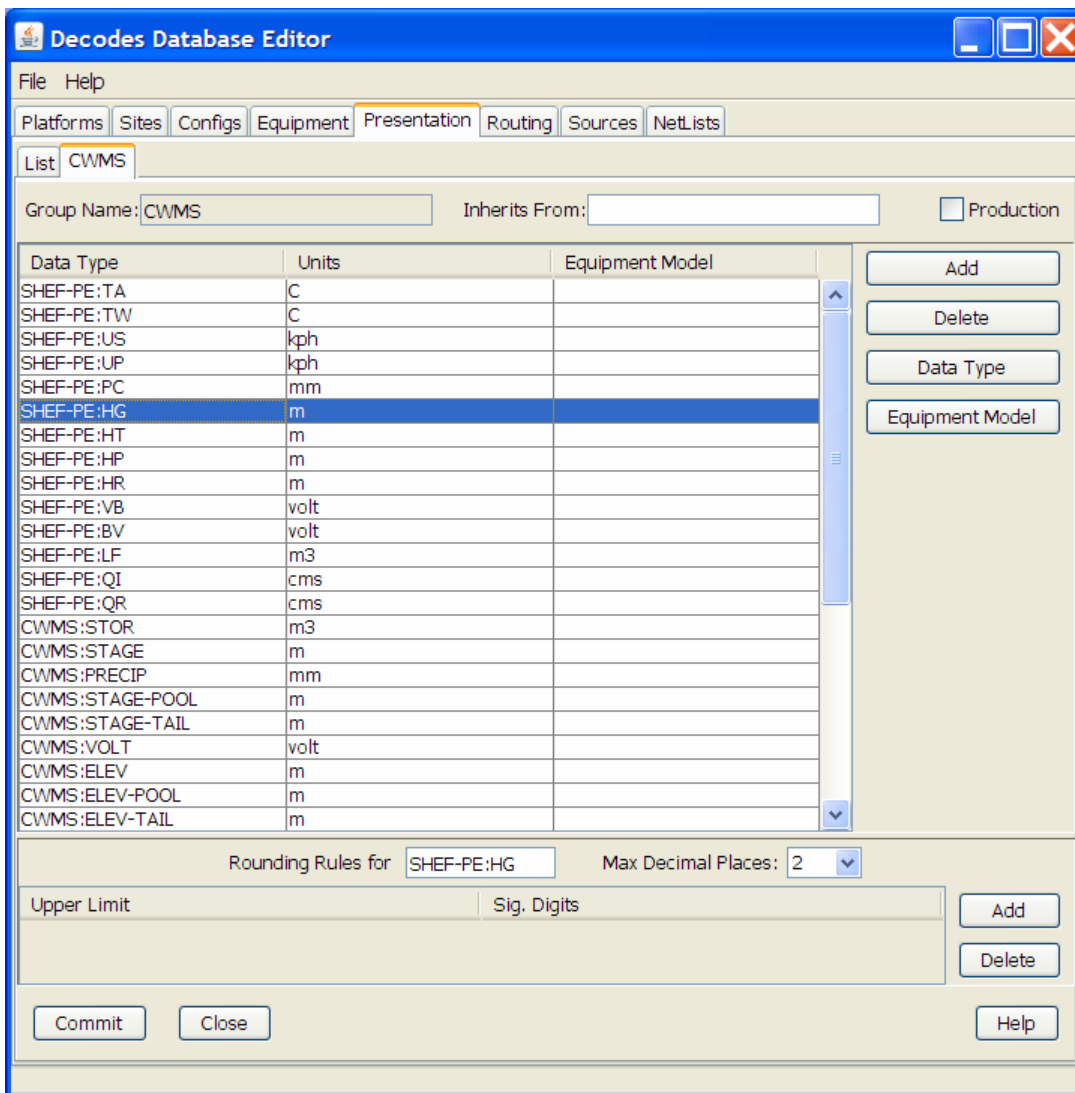
We also recommend that you select the “CWMS” presentation group. This will ensure that your data is converted into EUs that CWMS will accept.



## 18.6 Engineering Units

The sensor engineering-units need to be in compliance with the CWMS Oracle Database, otherwise the sensor data will not be accepted by CWMS. We have prepared a presentation group that will automatically convert your data into CWMS EUs. You simply have to select the presentation group in the routing spec.

Figure 18-2 shows the database editor with the CWMS presentation group open. See how the presentation group asserts which units should be used for each parameter type. When you apply the presentation group to a routing spec, DECODES will automatically convert the data into the correct units.



**Figure 18-2: Database Editor Showing the CWMS Presentation Group.**

Now, recall from section 5.6.1 that you can also use the presentation group to omit certain parameter types from the output. For example, if you do not store battery voltage in the CWMS database, change the units for VB to 'omit'.

## 18.7 Troubleshooting

The DECODES Routing Spec sends log messages to a file in the “routstat” directory under \$DECODES\_INSTALL\_DIR. Find the file there with the same name as your routing spec and an extension “.log”. For example if your Routing Spec is called “cwms\_rs”, the log file name will be: cwms\_rs.log.

The remainder of this section will provide examples of possible log messages, explaining what each means and what to do to correct the situation. A ‘FATAL’ message will result in the termination of the routing spec.

```
FATAL 03/06/07 16:56:46 CwmsConsumer Cannot load configuration from
'$DECODES_INSTALL_DIR/decodes-cwms.conf': java.io.IOException:
CwmsDbConfig Cannot open config file 'C:\DCSTOOL/decodes-cwms.conf':
java.io.FileNotFoundException: C:\DCSTOOL\decodes-cwms.conf (The system
cannot find the file specified)]
```

This fatal message means that the decodes-cwms.conf file was not found under the required directory. Make sure that the decodes-cwms.conf file is located under the DECODES installed directory.

```
WARNING 03/06/07 16:31:26 CwmsConsumer Cannot read DB auth from file
'C:\DCSTOOL/.cwmsdb.auth': java.io.FileNotFoundException:
C:\DCSTOOL\cwmsdb.auth (The system cannot find the file specified)
```

This warning message means that the authentication file, which contains the encryption of the username and password for the Database connection, is not on the right directory. Make sure that the .cwmsdb.auth file is located under the DECODES installed directory.

```
FATAL 03/06/07 16:31:26 CwmsConsumer Error getting JDBC ORACLE
connection using driver 'jdbc:oracle:thin:@' to database at
'155.76.210.137:1521:MVRT' for user "': java.sql.SQLException: invalid
arguments in call
```

CWMS Data Consumer will log Database connection fatal messages if:

- The wrong username/password was sent to it, which in this case make sure that the authentication file (.cwmsdb.auth) is on the right directory and contains the right username and password (this is the sample log shown above)
- The wrong CWMS Database connection information was supplied; in this case make sure that the DbUri property on the decodes-cwms.conf file contains the right Database connection information
- The CWMS Database server is down, in this case call the CWMS Database system administrator

WARNING 03/06/07 17:03:17 CwmsConsumer Cannot read properties file  
'C:\DCSTOOL\shfCwmsParam.prop': java.io.FileNotFoundException:  
C:\DCSTOOL\shfCwmsParam.prop (The system cannot find the file specified)

This warning message means that the shfCwmsParam.prop file was not found under the DECODES installed directory. However, this file is not required. If the user has decided not to use this file no action need to be taken. If not, make sure that this file exists under the DECODES installed directory.

WARNING 03/06/07 15:30:59 CwmsConsumer Platform Site Name nwshb5-STBI4, Platform Agency MVR, DCP Address CE2DC544, sensor HG Error while inserting sensor data in cwms\_ts.store\_ts CWMS procedure  
:java.sql.SQLException: ORA-20010: INVALID\_OFFICE\_ID: "tttMVR" is not a valid CWMS office id

This warning message means that the office that was set on the decodes-cwms.conf file is not valid for the CWMS Database. Make sure that the decodes-cwms.conf file contains the correct office value on the cwmsofficeid property.

WARNING 03/05/07 16:22:40 CwmsConsumer Platform Site Name nwshb5-STBI4, Platform Agency MVR, DCP Address CE2DC544, sensor VB Error while inserting sensor data in cwms\_ts.store\_ts CWMS procedure  
:java.sql.SQLException: ORA-20210: WARNING(cwms\_loc.get\_ts\_code): STBI4.Volt.Inst.1Hour.0.raw FOR OFFICE: MVR NOT FOUND

This warning message means that the time-series descriptor does not exists in the CWMS Database. Make sure that the CWMS Database contains the time-series descriptors specified in the warning message. In this case 'STBI4.Volt.Inst.1Hour.0.raw' for office MVR.

FAILURE 02/23/07 15:20:13 RoutingSpec(CWMSTEST) Error on data consumer 'cwms': decodes.consumer.DataConsumerException: CwmsConsumer Error while inserting sensor data in cwms\_ts.store\_ts CWMS procedure  
:java.sql.SQLException: ORA-20103: Requested unit conversion is not available

This warning message means that the CWMS Database does not recognize the unit value that CWMS Data Consumer sent. Make sure that the sensor unit is accepted by the CWMS Database, you may need to create a DECODES presentation group to convert units if the CWMS Database does not handle the current sensor unit. Refer to the DECODES Presentation group on the DECODES User Manual for more information.

WARNING 03/05/07 16:34:36 CwmsConsumer Platform Site Name nwshb5-CRVI4, Platform Agency MVR, DCP Address CE637FAC, sensor YA Cannot find CWMS or SHEF datatype -- skipping.

This warning message means that the time-series descriptor was not created for that particular sensor. Change the sensor data type to cwms with the correct cwms code (this is done on the Edit Config Sensor dialog) or add the mapping of that sensor data type code on the shefCwmsParam.prop file.

## Appendix A: Engineering Unit List

DECODES is delivered with a fairly complete list of engineering units that are used in hydrometeorologic applications. You may define additional EUs by editing the XML file. Please email any proposed changes/additions to the decodes Email forum so that, if appropriate, they can be included in a subsequent release.

When you enter unit values for sensors, use the abbreviation. Case is ignored so ft, Ft, FT all refer to feet.

### Sorted By Name

Name	Abbr	Family	Measures
acre feet	acre*ft	English	volume
acres	acre	English	area
atmospheres	atm	Metric	pressure
bars	bar	Metric	pressure
british thermal unit	btu	English	energy
Calories	cal	English	energy
centimeters	cM	Metric	length
centimeters per second	cM/s	Metric	speed
counts	count	univ	count
cubic centimeter	cc	Metric	volume
cubic feet	ft <sup>3</sup>	English	volume
cubic feet per second	cfs	English	flow
cubic feet per second	ft <sup>3</sup> /s	English	flow
cubic inches	in <sup>3</sup>	English	volume
cubic meter	m <sup>3</sup>	Metric	volume
cubic meters per second	m <sup>3</sup> /s	Metric	flow
days	day	univ	time
degrees Celsius	degC	Metric	temperature
degrees Fahrenheit	degF	English	temperature
degrees Kelvin	degK	Metric	temperature
dyn	dyn	Metric	force
ergs	erg	English	energy
feet	ft	English	length
feet per second	ft/s	English	speed
fluid ounce	floz	English	volume

foot-pounds per second	ft*lb/s	English	power
gallon	gal	English	volume
grams	G	Metric	mass
grams per liter	g/L	Metric	concentration
horsepower	hp	English	power
hours	hr	univ	time
inches	in	English	length
inches of mercury	inHg	English	pressure
inches per second	in/s	English	speed
joules	j	Metric	energy
Kilocalories	kcal	English	energy
kilograms	kG	Metric	mass
kilojoules	kj	Metric	energy
kiloliter	kL	Metric	volume
kilometers	kM	Metric	length
kilometers per hour	kM/hr	Metric	speed
kilopascals	kpa	Metric	pressure
kilowatts	kW	Metric	power
liter	L	Metric	volume
meters	M	Metric	length
meters per second	M/s	Metric	speed
Metric ton	mt	Metric	mass
micrograms	uG	Metric	mass
micrograms per liter	uG/L	Metric	concentration
microliter	uL	Metric	volume
micrometers	uM	Metric	length
MicroMHOs per centimeter	uMHOs/cm	metric	conductance
MicroMHOs per centimeter	uMHO	metric	conductance
MicroMHOs per centimeter	uMHOs	metric	conductance
miles	mi	English	length
miles per hour	mi/hr	English	speed
miles per hour	mph	English	speed
millibars	mbar	Metric	pressure
milligrams	mG	Metric	mass
milligrams per liter	mG/L	Metric	concentration
milliliter	mL	Metric	volume

millimeters	mM	Metric	length
millimeters of mercury	mmHg	Metric	pressure
millimeters per second	mM/s	Metric	speed
minutes	min	univ	time
nautical miles	nmi	English	length
nautical miles per hour	nmi/hr	English	speed
nautical miles per hour	knots	Metric	speed
newtons	N	Metric	force
ounces	oz	English	mass
parts per million	ppm	univ	ratio
parts per thousand	ppt	univ	ratio
pascals	pa	Metric	pressure
percent	pct	univ	ratio
percent	%	univ	ratio
pH	pH	univ	acidity
pint	pt	English	volume
pound-force	lbf	English	force
pounds	lb	English	mass
pounds per square inch	psi	English	pressure
quart	qt	English	volume
second	sec	univ	time
square centimeters	cM <sup>2</sup>	Metric	area
square feet	ft <sup>2</sup>	English	area
square inches	in <sup>2</sup>	English	area
square kilometers	kM <sup>2</sup>	Metric	area
square meters	M <sup>2</sup>	Metric	area
square miles	mi <sup>2</sup>	English	area
square millimeters	mM <sup>2</sup>	Metric	area
square yards	yd <sup>2</sup>	English	area
tons	ton	English	mass
volts	V	Metric	emf
watts	W	Metric	power
weeks	week	univ	time
yards	yd	English	length
yards per second	yd/s	English	speed

### Sorted By Abbreviation

Name	Abbr	Family	Measures
percent	%	univ	ratio
acres	acre	English	area
acre feet	acre*ft	English	volume
atmospheres	atm	Metric	pressure
bars	bar	Metric	pressure
british thermal unit	btu	English	energy
Calories	cal	English	energy
cubic centimeter	cc	Metric	volume
cubic feet per second	cfs	English	flow
centimeters	cM	Metric	length
centimeters per second	cM/s	Metric	speed
square centimeters	cM^2	Metric	area
counts	count	univ	count
days	day	univ	time
degrees Celsius	degC	Metric	temperature
degrees Fahrenheit	degF	English	temperature
degrees Kelvin	degK	Metric	temperature
dyn	dyn	Metric	force
ergs	erg	English	energy
fluid ounce	floz	English	volume
feet	ft	English	length
foot-pounds per second	ft*lbf/s	English	power
feet per second	ft/s	English	speed
square feet	ft^2	English	area
cubic feet	ft^3	English	volume
cubic feet per second	ft^3/s	English	flow
grams	G	Metric	mass
grams per liter	g/L	Metric	concentration
gallon	gal	English	volume
horsepower	hp	English	power
hours	hr	univ	time
inches	in	English	length
inches per second	in/s	English	speed



square inches	in <sup>2</sup>	English	area
cubic inches	in <sup>3</sup>	English	volume
inches of mercury	inHg	English	pressure
joules	j	Metric	energy
Kilocalories	kcal	English	energy
kilograms	kG	Metric	mass
kilojoules	kj	Metric	energy
kiloliter	kL	Metric	volume
kilometers	kM	Metric	length
kilometers per hour	kM/hr	Metric	speed
square kilometers	kM <sup>2</sup>	Metric	area
nautical miles per hour	knots	Metric	speed
kilopascals	kpa	Metric	pressure
kilowatts	kW	Metric	power
liter	L	Metric	volume
pounds	lb	English	mass
pound-force	lbf	English	force
meters	M	Metric	length
meters per second	M/s	Metric	speed
square meters	M <sup>2</sup>	Metric	area
cubic meter	m <sup>3</sup>	Metric	volume
cubic meters per second	m <sup>3</sup> /s	Metric	flow
millibars	mbar	Metric	pressure
milligrams	mG	Metric	mass
milligrams per liter	mG/L	Metric	concentration
miles	mi	English	length
miles per hour	mi/hr	English	speed
square miles	mi <sup>2</sup>	English	area
minutes	min	univ	time
milliliter	mL	Metric	volume
millimeters	mM	Metric	length
millimeters per second	mM/s	Metric	speed
square millimeters	mM <sup>2</sup>	Metric	area
millimeters of mercury	mmHg	Metric	pressure
miles per hour	mph	English	speed

Metric ton	mt	Metric	mass
newtons	N	Metric	force
nautical miles	nmi	English	length
nautical miles per hour	nmi/hr	English	speed
ounces	oz	English	mass
pascals	pa	Metric	pressure
percent	pct	univ	ratio
pH	pH	univ	acidity
parts per million	ppm	univ	ratio
parts per thousand	ppt	univ	ratio
pounds per square inch	psi	English	pressure
pint	pt	English	volume
quart	qt	English	volume
second	sec	univ	time
tons	ton	English	mass
micrograms	uG	Metric	mass
micrograms per liter	uG/L	Metric	concentration
microliter	uL	Metric	volume
micrometers	uM	Metric	length
MicroMHOs per centimeter	uMHO	metric	conductance
MicroMHOs per centimeter	uMHOs	metric	conductance
MicroMHOs per centimeter	uMHOs/cm	metric	conductance
volts	V	Metric	emf
watts	W	Metric	power
weeks	week	univ	time
yards	yd	English	length
yards per second	yd/s	English	speed
square yards	yd^2	English	area

**Sorted By Family, Name**

<b>Name</b>	<b>Abbr</b>	<b>Family</b>	<b>Measures</b>
acre feet	acre*ft	English	volume
acres	acre	English	area
british thermal unit	btu	English	energy
Calories	cal	English	energy
cubic feet	ft^3	English	volume
cubic feet per second	cfs	English	flow
cubic feet per second	ft^3/s	English	flow
cubic inches	in^3	English	volume
degrees Fahrenheit	degF	English	temperature
ergs	erg	English	energy
feet	ft	English	length
feet per second	ft/s	English	speed
fluid ounce	floz	English	volume
foot-pounds per second	ft*lbf/s	English	power
gallon	gal	English	volume
horsepower	hp	English	power
inches	in	English	length
inches of mercury	inHg	English	pressure
inches per second	in/s	English	speed
Kilocalories	kcal	English	energy
miles	mi	English	length
miles per hour	mi/hr	English	speed
miles per hour	mph	English	speed
nautical miles	nmi	English	length
nautical miles per hour	nmi/hr	English	speed
ounces	oz	English	mass
pint	pt	English	volume
pound-force	lbf	English	force
pounds	lb	English	mass
pounds per square inch	psi	English	pressure
quart	qt	English	volume
square feet	ft^2	English	area
square inches	in^2	English	area

square miles	mi <sup>2</sup>	English	area
square yards	yd <sup>2</sup>	English	area
tons	ton	English	mass
yards	yd	English	length
yards per second	yd/s	English	speed
atmospheres	atm	Metric	pressure
bars	bar	Metric	pressure
centimeters	cM	Metric	length
centimeters per second	cM/s	Metric	speed
cubic centimeter	cc	Metric	volume
cubic meter	m <sup>3</sup>	Metric	volume
cubic meters per second	m <sup>3</sup> /s	Metric	flow
degrees Celsius	degC	Metric	temperature
degrees Kelvin	degK	Metric	temperature
dyn	dyn	Metric	force
grams	G	Metric	mass
grams per liter	g/L	Metric	concentration
joules	j	Metric	energy
kilograms	kG	Metric	mass
kilojoules	kj	Metric	energy
kiloliter	kL	Metric	volume
kilometers	kM	Metric	length
kilometers per hour	kM/hr	Metric	speed
kilopascals	kpa	Metric	pressure
kilowatts	kW	Metric	power
liter	L	Metric	volume
meters	M	Metric	length
meters per second	M/s	Metric	speed
Metric ton	mt	Metric	mass
micrograms	uG	Metric	mass
micrograms per liter	uG/L	Metric	concentration
microliter	uL	Metric	volume
micrometers	uM	Metric	length
MicroMHOs per centimeter	uMHO	metric	conductance
MicroMHOs per centimeter	uMHOs	metric	conductance

MicroMHOs per centimeter	uMHOs/cm	metric	conductance
millibars	mbar	Metric	pressure
milligrams	mG	Metric	mass
milligrams per liter	mG/L	Metric	concentration
milliliter	mL	Metric	volume
millimeters	mM	Metric	length
millimeters of mercury	mmHg	Metric	pressure
millimeters per second	mM/s	Metric	speed
nautical miles per hour	knots	Metric	speed
newtons	N	Metric	force
pascals	pa	Metric	pressure
square centimeters	cM <sup>2</sup>	Metric	area
square kilometers	kM <sup>2</sup>	Metric	area
square meters	M <sup>2</sup>	Metric	area
square millimeters	mM <sup>2</sup>	Metric	area
volts	V	Metric	emf
watts	W	Metric	power
counts	count	univ	count
days	day	univ	time
hours	hr	univ	time
minutes	min	univ	time
parts per million	ppm	univ	ratio
parts per thousand	ppt	univ	ratio
percent	%	univ	ratio
percent	pct	univ	ratio
pH	pH	univ	acidity
second	sec	univ	time
weeks	week	univ	time

**Sorted By Measured Quantity, Name**

<b>Name</b>	<b>Abbr</b>	<b>Family</b>	<b>Measures</b>
pH	pH	univ	acidity
acres	acre	English	area
square centimeters	cM <sup>2</sup>	Metric	area
square feet	ft <sup>2</sup>	English	area
square inches	in <sup>2</sup>	English	area
square kilometers	kM <sup>2</sup>	Metric	area
square meters	M <sup>2</sup>	Metric	area
square miles	mi <sup>2</sup>	English	area
square millimeters	mM <sup>2</sup>	Metric	area
square yards	yd <sup>2</sup>	English	area
grams per liter	g/L	Metric	concentration
micrograms per liter	uG/L	Metric	concentration
milligrams per liter	mG/L	Metric	concentration
MicroMHOs per centimeter	uMHO	metric	conductance
MicroMHOs per centimeter	uMHOs	metric	conductance
MicroMHOs per centimeter	uMHOs/cm	metric	conductance
counts	count	univ	count
volts	V	Metric	emf
british thermal unit	btu	English	energy
Calories	cal	English	energy
ergs	erg	English	energy
joules	j	Metric	energy
Kilocalories	kcal	English	energy
kilojoules	kj	Metric	energy
cubic feet per second	cfs	English	flow
cubic feet per second	ft <sup>3</sup> /s	English	flow
cubic meters per second	m <sup>3</sup> /s	Metric	flow
dyn	dyn	Metric	force
newtons	N	Metric	force
pound-force	lbf	English	force
centimeters	cM	Metric	length
feet	ft	English	length
inches	in	English	length

kilometers	kM	Metric	length
meters	M	Metric	length
micrometers	uM	Metric	length
miles	mi	English	length
millimeters	mM	Metric	length
nautical miles	nmi	English	length
yards	yd	English	length
grams	G	Metric	mass
kilograms	kG	Metric	mass
Metric ton	mt	Metric	mass
micrograms	uG	Metric	mass
milligrams	mG	Metric	mass
ounces	oz	English	mass
pounds	lb	English	mass
tons	ton	English	mass
foot-pounds per second	ft*lb/s	English	power
horsepower	hp	English	power
kilowatts	kW	Metric	power
watts	W	Metric	power
atmospheres	atm	Metric	pressure
bars	bar	Metric	pressure
inches of mercury	inHg	English	pressure
kilopascals	kpa	Metric	pressure
millibars	mbar	Metric	pressure
millimeters of mercury	mmHg	Metric	pressure
pascals	pa	Metric	pressure
pounds per square inch	psi	English	pressure
parts per million	ppm	univ	ratio
parts per thousand	ppt	univ	ratio
percent	%	univ	ratio
percent	pct	univ	ratio
centimeters per second	cM/s	Metric	speed
feet per second	ft/s	English	speed
inches per second	in/s	English	speed
kilometers per hour	kM/hr	Metric	speed

meters per second	M/s	Metric	speed
miles per hour	mi/hr	English	speed
miles per hour	mph	English	speed
millimeters per second	mM/s	Metric	speed
nautical miles per hour	nmi/hr	English	speed
nautical miles per hour	knots	Metric	speed
yards per second	yd/s	English	speed
degrees Celsius	degC	Metric	temperature
degrees Fahrenheit	degF	English	temperature
degrees Kelvin	degK	Metric	temperature
days	day	univ	time
hours	hr	univ	time
minutes	min	univ	time
second	sec	univ	time
weeks	week	univ	time
acre feet	acre*ft	English	volume
cubic centimeter	cc	Metric	volume
cubic feet	ft^3	English	volume
cubic inches	in^3	English	volume
cubic meter	m^3	Metric	volume
fluid ounce	floz	English	volume
gallon	gal	English	volume
kiloliter	kL	Metric	volume
liter	L	Metric	volume
microliter	uL	Metric	volume
milliliter	mL	Metric	volume
pint	pt	English	volume
quart	qt	English	volume



