SNOTEL GOES Manual, Revision 2

Revision Date: 2/17/2011

Authored by
Richard Brown and Austin Beard

INDEX

Application Overview

msiDCSDecodeR2, provided by Micro Specialties, Inc (MSI), is designed to automate the retrieval of GOES data reports from the National Environmental Satellite Data Information System (NESDIS). These reports are presumed to have been generated by a Campbell Scientific (CSI) data logger, and communicated through the GOES system via a CSI TX412 data modem, or via a MicroComm GTX data modem. The application can be configured to collect data from any number of GOES data platforms, and will generate SNOTEL-compatible data messages intended to be automatically ingested by SNOTEL. Data messages received from GOES, are automatically processed and appended to an output text file named mbdata.txt. An external, cron-timed batch process should be provided by NWCC, that will execute msiDCSdecode and feed the content of mbdata.txt into the SNOTEL system. The batch process should rename mbdata.txt to avoid having the application write to it while the batch is running, then delete it when its content has been ingested into SNOTEL, as this application will never delete or shrink mbdata.txt.

New Features and Revisions in Revision 2a 2/9/2010

- **Decodes all GTX sensor labels**
- **Recognizes sensor values generated internal to the GTX such as the 'Clock', 'Battery', and values from the equation editor.**
- **Understands GTX min/max/avg (NXG) flags**
- **Will decode multiple values from a single GTX TPL Data Parameter entry based on the 'Value Count' of that parameter.Allows more flexibility in site group definitions.**
- **Keeps track of successful data retrieval events to help prevent data gaps.**
- **Exports most GOES procedures to new GOES python module.**
- **New command line option to “fresh start” retrieval history.**

New Features and Revisions in Revision 2b - 2/16/2011

- **ddsClient now logs only retrieve dates more recent than the last entry in the log. This is intended to remove duplicate entries in the log.**
- **Sorts the data records (mbdata.txt) by the data time stamp.**
- **ddsClient now restricts attempted data retrieval to within 30 days of the current date. Dates in the future are not allowed.**
- **ddsClient now attempts to get data from the primary LRGS server, but will switch to the backup server if the primary does not respond. The ddsClient module now defines the IPs for the primary and backup LRGS servers.**
- **Provides a command line option to use a specified LRGS (host) IP.**

Directories and Files

All **Python modules**, **log files**, **data files** and **configuration files** should reside within the same directory, hereafter referred to as the Application Directory. Log files and data files are automatically appended as a product of the msiDCSdecode application. Log and data files are never automatically deleted by the application.

Python Modules

The entire GOES Decoder application is written in Python, and is directly portable to any computing platform, Unix, Linux, or Windows, which has a Python Interpreter installed. The application consists of several Python modules, which should be placed in the application directory:

Revision 2a – 2/9/2010

- **msiDCSdecodeR1.py rev 2.8.2010** – The application's main module, which collects data from NESDIS, decodes the data streams, and formats data messages to be ingested by SNOTEL. msiDCSdecodeR1 imports the following modules:
- **ddsClient.py rev 1.29.2010** – Objects and methods which implement a client to the NESDIS LRGS server.
- **goesSiteDefs.py rev 2.2.2010** – Objects and methods which operate on the individual data site definitions (sitedefs.txt).
- **gtxDefs.py rev 2.9.2010** - Objects and methods pertaining to the definitions of individual MicroComm GTX data modem sites.
- **GOES.py rev 2.2.2010** – Objects and methods pertaining to NESDIS GOES message structure.

Revision 2b – 2/17/2011

- **msiDCSdecodeR1.py rev 2.17.2011** – The application's main module, which collects data from NESDIS, decodes the data streams, and formats data messages to be ingested by SNOTEL. msiDCSdecodeR1 imports the following modules:
- **ddsClient.py rev 2.15.2011** – Objects and methods which implement a client to the NESDIS LRGS server.
- **goesSiteDefs.py rev 2.2.2010** – Objects and methods which operate on the individual data site definitions (sitedefs.txt).
- **gtxDefs.py rev 2.9.2010** - Objects and methods pertaining to the definitions of individual MicroComm GTX data modem sites.

- **GOES.py rev 2.2.2010** – Objects and methods pertaining to NESDIS GOES message structure.

Configuration Files

The operation of msiDCSdecode and its supporting modules is controlled and configured by several configuration files. All configuration files are simple, plain-text files, which may be edited by any plain-text editor such as vi or notepad.

sitedefs.txt – Defines the site specifics and message grouping for all sites defined within the application.
This configuration file uses keyword:value mapping to define configuration values. The following keywords are valid:

Keyword	Values
site:	A generic name for the data site, ie: Sidney, Montana
snotID:	The site's SNOTEL site ID, ie: 2120
goesID:	The site's GOES DCP address, ie:DA5014EA
gmtOffset:	The time offset from GMT for this site, ie: -8 (PST)
modemType:	The GOES modem used, ie: tx312 or gtx
group:	SNOTEL group definition: grpNum, startIndex..endIndex ie: group:1,2..17
enabled	Enables data collection for this site

SiteDef Notes:

- Lines beginning with the pound character (#) are ignored.
- The scope of each site definition is from the **site** keyword, to the next **site** keyword.
- **The order of keywords is not important, except that all keywords following a "site" keyword, apply to that site.** Groups, for instance, may be defined in any order.
- Each site may have multiple **group** definitions.
- GTX **sensor labels**, defined within the GTX TPL file can now be used freely and interchangeably with sensor array indices to define sensor groups.
- Group definitions may contain multiple sensor sequences like 3..6 which is equivalent to 3,4,5,6 or batt..pill, or 3..pill.
- The sensor sequence in a Group Definition may be in any random order now. For example a group may be defined with this order: 8..10,7,2,3..6,12.
- If a **group** definition has the added parameter of "**midnight**", that group is expected only at the midnight (0000) hour. Each GOES message is expected to contain in its data array[0], the Julian Date produced by the data logger, and in array[1], the local standard time produced by the data logger. **Midnight groups are only processed when array[1] is 0000 (midnight). At all other times, midnight groups are ignored, whether or not the values are present in the array.**
- The gmtOffset is used to calculate the site's Local Standard Time. Daylight Time is not supported. The MBD data message's reception time is delivered as Local Standard Time, as well as the message's data time. **Note that the reception time is based on the DCP's GOES transmission-time window, and will always be the same time of the hour. For this reason, the difference between the reception time and the data time cannot be utilized as a measure of the site's communication performance**.

Example site definition for a TX312 type site:

```
site:Sidney                 ## Site Name
snotID:2120                 ## SNOTEL ID
goesID:DA5092FE             ##  GOES ID
gmtOffset:-7                ##  local time is 7 hours behind GMT
modemType:tx312             ## tx312 Modem Type
group:1,2..16               ## Group 1 uses GOES data fields 2 through 16
group:2,17..32              ## Group 2 uses GOES data fields 17 through 32
group:3,33..44              ## Group 3 uses GOES data fields 33 through 44
group:4,46..55,midnight     ## Group 4 is only received at midnight
#enabled                    ## This site is not enabled
```

Example site definition for a GTX type site:

```
site:Upper Bethlehem
snotID:2123
goesID:DA5014EA
gmtOffset:-8
modemType:gtx
enabled
## Primary sensors by label, single sequence:
group:1,Batt..avDP
## Soils by label, 4 sequences for clarity:
group:2,1V1..1B4,2V1..1V4,3V1..3V4,4V1..4V4
group:3,5V1..5V4
## Soils by GOES message index:
group:4,41..45
```

TPL files - Microcom data definition file. Named per SNOTEL ID (nnnn.tpl), these files define the Microcom GTX datafields used in the data extraction process. Each site, transmitting data through a Microcom GTX data modem, must have a tpl file named for its SNOTEL ID, placed in the application directory. MsiDCSdecode uses the tpl file to understand the structure of each GTX GOES data message. If the configuration of the data modem is modified, an updated tpl file must be placed in the working directory.

New in revision 2, msiDCSdecode understands GTX Min-Max-Avg (NXG) processing, and will produce text labels for all NXG processed data points. The GTX NXG process may produce up to 3 data points, one for each flag from the set: [NXG]. GTX will allow you to include recent samples and NXG values within the GOES message data stream. For example:
The GTX tpl statement: **IL4=DYSI** defines the label DYSI for internal sensor 4 (IL4).
The GTX tpl statement: **MMA=0,I4,24:00:00,00:00:00,NXG** tells the GTX to calculate minimum (N), maximum (X) and average (G) values for internal (I) sensor number 4.
The GTX tpl statement: **TDP=5,I4,2NXG,…** tells the GTX to include in the data stream:

- Two (2) recent sensor sample from the I4 sensor
- The minimum (N) value of I4
- The maximum (X) value of I4
- The average (G) of I4 samples

msiDCSdecode will produce the following labels form the above statement set:

- DYSI The most recently sampled data, sample number 1. When n-1=0 there is no numbered suffix added to the label.
- DYSI-2 The second sample (number 2) has an n-1=2-1=1, therefore a ‘-1’ suffix is added to the label
- DYSI-N The minimum values in the sample set
- DYSI-X The maximum value in the sample set
- DYSI-G The average of the samples in the sample set

These labels can then be used in sensor group definitions within the sitedefs.txt configuration file.

Log Files

Log files contain operational messages or application error messages, which may have been generated as a result of events encountered by msiDCSdecode.

- **DCSactivity.log** -The activity log file contains normal operational messages which have been generated during normal operation of the msiDCSdecode application. These messages may be helpful in determining whether the application is performing correctly.

- **DCSproblem.log** – The problem log file contains error messages, which may have been generated as a result of events or problems encounterd by msiDCSdecode.

- **Retrieve.log** – Recent successful data retrieval times in GMT. The methods in the ddsClient use these times to determine what data needs to be retrieved from NESDIS. These times may be helpful in determining the duration of networking interruptions.

- **DCSrmp.log** – The application automatically appends a Remote Maintenance Parameter (RMP) message to the DCSrmp.log file for each new data message received from each site. The application never deletes or shrinks the data file. Each RMP parameter field contains a keyword: value pair. The following RMP keywords are provided:

Keyword	Values
sigdbm:	Signal Strength in dbm
freqOff:	Frequency Offset: + or – 0 through A (50hz increments)
modIndex:	Modulation Index:(N)ormal, (L)ow or (H)igh
dataQual:	Data Quality: (N)ormal, (L)ow or (H)igh
goesStat:	Uplink Carrier Status
goesChan:	GOES Channel: 001 through 266
goesSat:	GOES Satellite: (E)ast or (W)est

Data files

- **mbdata.txt** - The application automatically appends all data messages decoded from the GOES system to the file named **mbdata.txt**. All data messages are constructed in the MBD data message format, using CSI FP2 hex-coded sensor fields. Each GOES message may spawn the generation of a number of data messages in MBD format, as defined in the sitedefs.txt configuration file. The application never deletes or shrinks this data file.

- **GOES Message Archive Files** – The application automatically archives all GOES messages, in original form, to archive files in the working directory. One file will exist for each calendar date, and each file will contain all GOES messages GOES time-stamped for that date. Duplicate messages are discarded. Example GOES Message Archive file name for March 3, 2008: 20080303.goesMsgs

GOES Data Modem Types Currently Supported

Campbell Scientific TX312 Modem - Supported as of 8/9/2007

The TX312 data modem can transmit data over GOES in either the normal CSI low resolution data format, or in a higher resolution, 18 bit integer format. In low resolution mode, date and time values can be directly inserted into the GOES data stream. In high resolution mode, date and time are not directly inserted into the data stream, but may be inserted by first redirecting the date and time to input locations, then sampling those input locations in the output section of the CR10X program. **Note that as of this writing, the high resolution mode has not been tested**.

MsiDCSdecode currently expects data from TX312 modems to be presented in low resolution format only.

TX312 Rules:

- Use normal, low resolution mode in the CR10 program
- All date and time information is derived from the first date/time fields (fields 0 and 1) in each GOES data message, so there is no reason to include date and time in each data group, although it won't kill the process if you do.
- Array IDs are not included in the low resolution TX312 GOES data stream, so if you need these, you will have to redirect them to input locations, then sample those input locations in the program's output section.
- All date/times and all data fields are transmitted as a single, all inclusive message group in the GOES data stream. **msiDCSdecode assumes that:**
    - The date (julian) is the first data field of the data stream.
    - The time (hhmm) is the second field.
- Keep track of each data group's starting and ending positions within the data stream, and give that information to Rose. For example: group 1, 2..17 defines group number 1 as extending from the third field through the 18th field of the data stream (16 sensors) . The first field is the date, and the second field is the time.

MicroComm GTX Data Modem - Supported as of 3/5/2008
MicroComm's GTX data modem can transmit data received from the CR10X in various field widths within the GOES data stream. The GTX applies a multiplier and offset and field width to each field it receives from the CR10X. msiDCSdecode must be informed of the field width, multiplier and offset applied by the GTX modem for each field in the data stream. msiDCSdecode informs itself of these parameters by parsing the tpl file generated by the GTX setup program for each site. **The tpl file must be sent to Rose before the data can be decoded for you new data site**.

GTX Rules:

- The date (julian) is the first field of the data stream.
- The time (hhmm) is the second field of the data stream.
- The tpl file used to configure the site GTX radio must be provided and named with the SNOTEL ID. For example, Ashley Divide has a SNOTEL site ID 5009 so the tpl file is named 5009.tpl.

Data Retrieval Methodology

MsiDCSdecode Revision 2 now retrieves data from the NESDIS LRGS server only when data has not been retrieved within the last hour, so you may execute the process every ten minutes or so without invoking extra overhead. The ddsClient module keeps track of successful retrieval events by logging the date/time (GMT) of the latest GOES message returned by NESDIS at each successful retrieval. 50 events are retained in the retrieval log. If the retrieval log does not exist, the process will attempt to retrieve only the past hour's data from NESDIS, otherwise ddsClient will attempt to retrieve all data from the last successful retrieval through the current date/time. The ddsClient now uses block mode data retrieval, which speeds up the process considerably.

To start over, simply delete the retrieval log, or invoke msiDCSdecode with the 'freshStart' command line option.

Command Line Options

By default, msDCSdecode retrieves and processes all GOES data messages for all sites which are defined and enabled in the sitedefs configuration file. Also by default, only data messages received by NESDIS within the past hour will be retrieved. The application's behavior can be modified through command line options. **Remember to protect parameters from the command shell with quotations, where parameters contain spaces (See Command Line Examples section below)**. The following command line options are available:

- **host** – Define the IP of the LRGS from which data is collected. If this option is used, automatic switching from the primary to the backup host is disabled.
- **help** – Show command syntax help and terminate. Do not retrieve or process anything.

- **snotID:nnnn** – SNOTEL ID to retrieve and process. Applying this option tells the application to retrieve and process data message only for the named snotID. This snotID must be known and enabled in the sitedefs configuration file.
- **start:startSpec** - start retrieving data received by NESDIS per this date specification. This specification bypasses the normal, automated means of retrieval.
- **stop:stopSpec** – stop retrieving data received by NESDIS per this date specification, when stop date is reached.
- **Start and stop date specifications may be of the following forms:**
    - **now – n minutes** - start or stop some minute in the past
    - **now – n hours** - start or stop some hours in the past
    - **now – 1 day** - start or stop 1 day in the past
    - **now - n days** - start or stop some days in the past
    - **YYYY/MM/DD** - start or stop at specific date (GMT).
- **freshStart** – Force msiDCSdecode to start over with its data collection effort, flushing the retrieve times and collecting data for the current day. Retrieve times begin logging immediately.
- **showDataMsgsOnly** – Display SNOTEL messages to stdout, showing values as real numbers.
- **showDecodingTable** – Display GOES messages on stdout, decoded per site's tpl file.
- **rev** – Show all module revision dates, then exit
- **verbose** – Show verbose progress messages on the stdout. This may be helpful when troubleshooting system operation. By setting verbose mode and redirecting the stdout to a text file, progress messages can be saved and sent to MSI for evaluation: **msiDCSdecodeR1.py verbose >output.txt**

Command Line Examples

- **msiDCSdecodeR1.py**

    Collects and processes all data from NESDIS as required. This is the normal mode of operation, where the process determines what data to retrieve.

- **msiDCSdecodeR1.py freshStart**

    The process flushes its retrieve log, and starts over, collecting all data from the previous hour of the current day. No other log or archive files are affected.

- **msiSCSdecodeR1.py rev**

    Prints the revision dates of all process modules, then exits.

- **msiDCSdecodeR1.py start:"now – 2 days"**

  Collects and processes the past 2 days of data.

  Please Note:
  - **The start specification contains spaces, so must be protected from the OS command shell by quotation.**
  - **The command parameter is protected from the command shell by enclosing it in quotations**.
  - **Retrieval logging is not used when a start date is specified, so normal, automated data retrieval is not affected.**

- **msiDCSdecodeR1.py start:2008/1/1 stop:2008/1/2**

  Collects and processes from start date until stop date is reached. **In this example, data is retrieved from 2008/1/1 00:00 through 2008/1/1 23:59**.

- **msiDCSdecodeR1.py snotID:2120 start:2008/1/1 verbose**

  Collects and processes all data for snotID 2120 from the beginning of 2008, through the present time. <u>Use this method to retrieve all data for a single site from the recent past.</u>

  Please Note:
  - **snotID must be defined and enabled in the sitedefs configuration file.**
  - **The NESDIS server retains only about 30 days of data for retrieval, so using a start date of earlier than 30 days in the past is of little value.**
  - **Retrieval logging is not used when a start date is specified, so normal, automated data retrieval is not affected.**
  - **The verbose option produces a large number of progress messages to be output to the console for troubleshooting purposes.**