USCRN/USRCRN Data Ingest

Functional Specification

Diana Kantor, Software Developer, ERT Inc.

Updated: November 18, 2013

Table of Contents

Overview

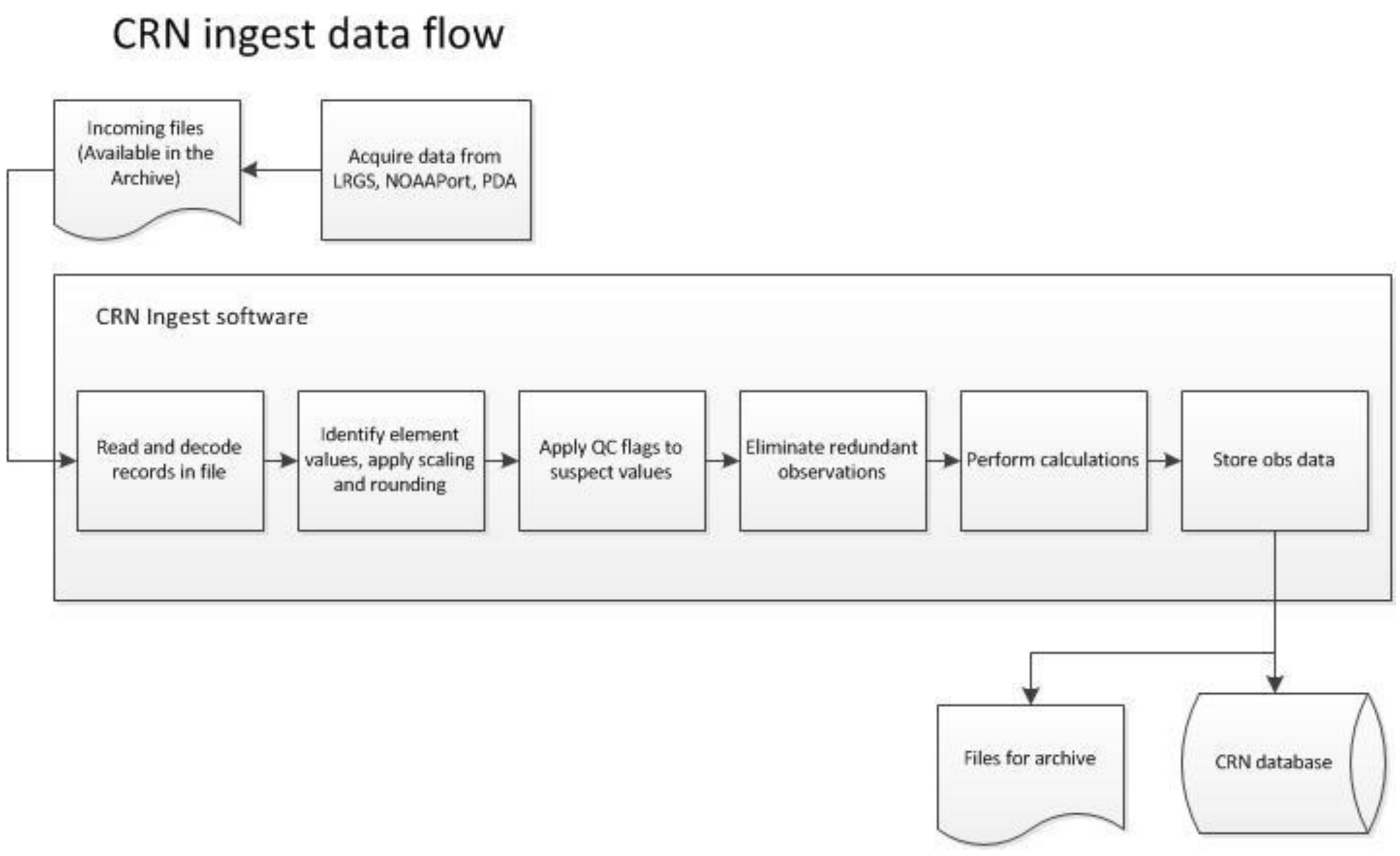
The purpose of this document is to provide details of the steps and algorithms required to fully process incoming US Climate Reference Network (USCRN) and US Regional Climate Reference Network (USRCRN) data records. This document should contain the information necessary to reproduce the data that are available in NCDC's data archive and on the USCRN website. USCRN and USRCRN data are processed identically, except where otherwise noted. The acronym "CRN" is used in this document to refer to the program as a whole, including both networks.

The basic steps for processing incoming records are:

1. Read and decode (if applicable) the records from an incoming file
2. Identify element values and apply scaling and rounding to values using stream definitions
3. Apply flags to suspect values using QC algorithms
4. Eliminate redundant observations by choosing the highest quality observation for a given station and datetime
5. Perform calculations for temperature, precipitation, and soil
6. Store observation data, including measured values, flags and calculated values

See also the **CRN ingest data flow** diagram below.

This document does not include all of the functionality present in our software, but only what is required to reproduce our data values. For instance, our software contains many monitoring features to ensure that systems are functioning properly. Such features are beyond the scope of this document.


CRN ingest data flow
Incoming files (Available in the Archive)
Acquire data from LRGS, NOAAPort, PDA
CRN Ingest software
Read and decode records in file
Identify element values, apply scaling and rounding
Apply QC flags to suspect values
Eliminate redundant observations
Perform calculations
Store obs data
Files for archive
CRN database

Incoming Records

Data come to us in one of three formats: LRGS, NOAAPort, or PDA. LRGS and NOAAPort both contain records as they were transmitted via satellite while PDA records are downloaded directly from a station's datalogger.

LRGS

LRGS (Local Readout Ground Station) records are transmitted via satellite and have a specific header prepended to each transmitted record. The header is in a readable ASCII format, while the rest of the record is pseudo-binary format. The header contains several useful pieces of information.

Example LRGS records (abbreviated):

```
CD14247C12075234141G44-0NN188WFF01226"@_\@AK@c|IsF}TlA_UE...
CD09A69012075234143G50+0NN185EFF01226 @_\@AK@c|H_T}wDAOmF...
CD1C802212123173922G38+0NN184WFF01478`@_\@A{@ZdOTP|cX@gH~...
```

Each LRGS record has a header that is 37 characters long, followed by a "separating" character or space, and then the message body. The header is comprised of the following values[1]. The 2 bolded fields are required for data processing.

- **8 hex digit DCP Address (GOES ID)** - Required for determining station. See the **Determining Station** section below.
- YYDDDHHMMSS – Time the message arrived at the receive station. The day is represented as a three digit day of the year (julian day).
- **1 character failure code** - Either G (Good) or ? (Parity error). Other codes may be present when a message was transmitted rather than data.
- 2 decimal digit signal strength
- 2 decimal digit frequency offset
- 1 character modulation index
- 1 character data quality indicator
- 3 decimal digit GOES receive channel
- 1 character GOES spacecraft indicator ('E' or 'W')
- 2 hex digit uplink carrier status
- 5 decimal digit message data length

The "separating" character is an arbitrary character between the header and the beginning of the data values. This character is usually a space or one of the following characters: ", ', `,b, B.

After the station is identified and it is established that the record does not contain a parity error, the record's data values must be decoded. See the **Decoding** section and the RecordDecoder.java code.

NOAAPort

NOAAPort records also originate from a satellite transmission but have their own specific header prepended to each transmitted record. NOAAPort records are trickier because they also have data appended to the end of the record. Additionally, there is an odd line break dividing the prepended header information into 2 separate lines. In most cases, this has the effect of the first part of that header

[1] From the Tempest LRGS User Guide, downloadable from http://www.ilexengineering.com/download

being included at the end of the previous line. NOAAPort files may contain either single line breaks or double line breaks.

The CRN Ingest software has a helper method to reconstruct a NOAAPort record into a single-line record with all of its prepended and appended information combined into a single header:
`RecordDecoder.sanitizeNoaaport(String record, String previousRecord)`

Additionally, you can pass an entire file to the following method to automatically reformat any NOAAPort records that are included in the file, as well as break the file contents into a `Collection` of `String` objects, each representing a record:

```
RecordDecoder.getRawRecords(File file)
```

Example NOAAPort records as transmitted (shortened to preserve document space):

```
SXXX90 KWAL 152100
CD0246CA 075210041`@_\@AK@_PHjf}.....@@@ 45+0NN 185ESXXX90 KWAL 152100

CD009556?075210041b@_\@AK@_PKrt}[.....E@@@ 44+1NN 180WSXXX90 KWAL 152101
```

Example (shortened) NOAAPort records after reformatting by `sanitizeNoaaport`:

```
SXXX90 KWAL 152100 CD0246CA45+0NN185E 075210041`2012,75,2000,...,2515,0
SXXX90 KWAL 152100 CD00955644+1NN180W?075210041b2012,75,2000,48308,...1669,0
```

Each *reformatted* NOAAPort record has a header that is 47 characters long, followed by a "separating" character or space, and then the message body. The header is comprised of the following values. The 2 bolded fields are required for data processing.

- 6-character code with the format `SXXX[80|90]`. In some cases the 2nd and 3rd Xs are replaced by the two-character US state code, such as `SXNC90`. These designators are not used by our system.
- 1-character space
- 4-character string `KWAL` to indicate the source was Wallops Island
- 1-character space
- 6-character numeric string representing the day of month followed by UTC time
- **8 hex digit DCP Address** - Required for determining station. See the **Determining Station** section below.
- 10-character location information
- **1-character failure code** - Either a single space (Good) or ? (Parity error).

The "separating" character between the header and the data values is an arbitrary character. This character is usually a space or one of the following characters: ", ', `,b, B.

Just as with the LRGS records, the header is in a readable ASCII format, while the rest of the record is pseudo-binary format. After the station is identified and it is established that the record does not contain a parity error, the record's data values must be decoded. See the **Decoding** section and the RecordDecoder.java code.

PDA (Datalogger output)

PDA records are downloaded directly from the datalogger located at a station during maintenance visits and are transferred to us via FTP. They are already in ASCII format and do not need to be decoded. Each line is one record, representing only one observation (one hour of instrument readings). Each record is a comma-separated values string. Rather than a header string, each record is prefixed only with the ATDD number for the station.

Here is a typical PDA record (shortened to preserve document space):

```
165,2010,258,1500,44.06028,-90.17389,2.402,13.19,13.17,13.18,13.19,...18.6,4.527,-6666
```

The first value is the ATDD number for that station, followed by all the recorded values for that observation in order based on the stream being used (see **Streams** section below).

165 is the ATDD number for WI Necedah 5 WNW. See the **Determining Station** section below.

Decoding Pseudo-Binary to ASCII

LRGS and NOAAPort records are transmitted in pseudo-binary and must be decoded into ASCII in order to be readable. PDA records are already in ASCII format and so do not need to be decoded. Very old transmitted records may already be in ASCII format and not need decoding.

The below Java code snippet can be used for decoding transmitted records to ASCII.

```
// Get the data portion of the record without the header
String binaryStr = incomingRecord.substring(headerLength);

// Convert to a byte array
byte[] bytes = binaryStr.getBytes();

// Initialize an object for storing the decoded record
String decodedRecord = "";

// Loop through all bytes in the binaryStr byte array
// Get groups of 3 bytes at a time to pass to decodeBinaryValue(b)
int len = 0;
while (len < bytes.length-1)
{
    // Add a comma between values
    if(len != 0)
        decodedRecord += ",";

    // Get groups of 3 bytes
    byte[] b = new byte[3];
    int i = 0;
    while (i < 3 && len < bytes.length-1)
    {
        b[i] = bytes[len];
        len = len + 1;
```

```
            i++;
        }

        // Append the decoded value to the decodedRecord object
        decodedRecord += decodeBinaryValue(b);
    }

    /**
    * Decodes a single value.
    *
    * @param b A byte array of 3 bytes from the incoming record. Each
    * 3 bytes represents 1 ASCII value.
    * @return An ASCII value.
    */
    private static int decodeBinaryValue(byte[] b)
    {
        int b1 = (int) (b[0] & 0x3F);
        int b2 = (int) (b[1] & 0x3F);
        int b3 = (int) (b[2] & 0x3F);
        boolean neg = (b1 & 0x20) == 32;
        b1 = b1 << 12;
        b2 = b2 << 6;
        int value = b1 | b2 | b3;

        if (neg)
        {
            value = value-1;
            value = -(~value & 0x3FFFF);
        }

        return value;
    }
```

Alternatively, you can use the full RecordDecoder.java class to read in a file of any record type(s) and output properly decoded records. This is advisable, particularly when decoding NOAAPort records due to the complex handling required for the NOAAPort header information.

Determining Station

CRN_Stations.csv contains the CRN station metadata used for determining the station from which a record came. The Integrated Station Information System (ISIS), located at https://ncdc.noaa.gov/isis, can also be used and will always have our most current station data.

ATDD Number

For PDA records, the station can be determined by the very first value in the record, which is the ATDD number. A station's ATDD number is available in ISIS and in the CRN_Stations.csv in the `ATDDNO` field.

GOES ID

For LRGS and NOAAPort record, use the GOES ID to determine which station the record belongs to. Each GOES ID maps to a station and can be found in ISIS and in the CRN_Stations.csv file in the `GOES_ID` field. All GOES IDs are 8 characters and most CRN GOES IDs begin with "CD".

In some cases, more than one station will have the same GOES ID. This happens when one station is closed and its GOES ID is reassigned to a new station. It is *never* the case that more than one station is using the same GOES ID at the same time. If a GOES ID has two stations assigned to it, use the date of the observation being processed in conjunction with the closed dates of the stations (`Closed Date` in CRN_Stations.csv) to determine which station is being processed.

As of 2011, Alaska has stations with 2 transmitters and therefore 2 GOES IDs, sending two transmissions each hour. These stations are currently treated as two separate stations in our system. Both transmitted records from such a station are processed as though they came from two different stations.

Discarding bad records

Records can become corrupted during transmission. Such records are discarded. There are several indicators used to determine that a record is corrupted.

Parity Errors

The header of LRGS records and NOAAPort records contains a “parity” indicator. If this indicator is “?” the record was corrupted and should be discarded.

Non-consecutive Datetimes

Transmitted records contain either 2 or 3 observations that are all in consecutive datetime order. Modern records order the obs in descending datetime order, while some very old records order the obs in ascending datetime order. If the observations in a single record are not in consecutive datetime order, whether ascending or descending, the entire record should be discarded.

Datetime in the Future

If a record contains an observation that has a datetime in the future, the record should be discarded. A future datetime could either mean that the record was corrupted or that the datalogger at the station had its current datetime set incorrectly.

Invalid GOES ID and datetime combination

For LRGS or NOAAPort records, the GOES ID at the beginning of the header must be a valid GOES ID for the datetime of the observations in the record. If a station has been closed, it is possible that its GOES ID is being reused, possibly by an outside agency. Records for such a reused GOES ID could be accidentally picked up and included in our incoming records files. A record with a datetime that is outside the datetime range for any stations in our system with that GOES ID, should be discarded.

Unusually Short Records

Corrupted records will often be truncated. Any record that contains fewer values than the minimum number of values expected for at least one observation for its stream (see the **Streams** section for more information on stream lengths) should be discarded. Partial observations are never processed.

Other Invalid Observation Values

Several valid values are required for proper processing of observations in a record. If any of these values are not reasonable, the entire record is assumed to be corrupted and it should be discarded.

These values include:

- `YEAR` - value must be a reasonable year, less than or equal to the present year
- `JULIAN_DAY` (day of year) - value must be a number between 1 and 366
- `ZTIME` (hour in UTC) - value must be 0, 100, 200,....2300
- `CRX_VERSION` - value must match a valid stream in our system
- `LATITUDE` - value must be between -90 and 90 or between -900 and 900 (if has not yet been scaled). It is not required that it be correct for proper processing.
- `LONGITUDE` - value must be between -180 and 180 or between -1800 and 1800 (if has not yet been scaled). It is not required that it be correct for proper processing.

Streams

By “stream” we mean a comma-separated list of element values in a particular order, representing an observation (one hour of data for the station) or multiple consecutive observations. The datalogger at a station has a particular datalogger program version (or CRX Version) which complies with a particular stream definition. You must know the stream definition of an incoming record in order to be able to understand what each value represents and how to scale and round the values.

Determining Which Stream to Use

Each incoming record contains a value for the element CRX_VN (CRX Version). This value maps to a stream definition file.

To determine which stream a record is using, get the 6th comma separated value. If it is a transmitted record, multiply that value by .001. Except for very old streams (see below), this will be the `CRX_VN` value. Use `CRX_VN` to find the correct stream definition file by using CRN_CRX_Stream_Mapping.csv. The first column is the `CRN_VN` number, the second is the `STREAM_ID`, and the third column is the number of expected elements for the stream definition, which is useful for **handling old streams** (see below). The fourth column indicates whether the stream includes 15-minute measurements or 5-minute measurements.

Using a Stream Definition file

After a record has been properly decoded (if necessary) into a string of comma-separated values, refer to the available stream definition files to properly read each value in the string. Stream definition files are archived as CSV files. If opened as a spreadsheet, a stream definition file will look similar to the following:

Position	**Multiplier**	**Stored Decimals**	**Element Name**	**Element Description**
1	1	0	YEAR	utc year of observation

2	1	0	JULIAN_DAY	utc day of year of observation (1-366)
3	1	0	ZTIME	utc hour of observation (end of hour, range: 100-2400)
4	0.001	3	LATITUDE	station latitude
5	0.01	2	LONGITUDE	station longitude
6	0.001	3	CRX_VN	datalogger version number
7	0.001	2	T160	temp sensor 1 average temp for 5 minutes ending at :60
8	0.001	2	T155	temp sensor 1 average temp for 5 minutes ending at :55
9	0.001	2	T150	temp sensor 1 average temp for 5 minutes ending at :50
10	0.001	2	T145	temp sensor 1 average temp for 5 minutes ending at :45
….				

The **Position** column is the position of that value in the incoming record (after the header). According to the above stream definition table, the first element value in the comma-separated values string, should be the year of the observation. The second value is the julian day.. The third is the hour in UTC, etc. This continues all the way to the last element of the stream definition file. If more than one observation is present in the record, start over with the first element value to represent the second observation. Then repeat if there is a third observation in the record. There may also be 36 “rolling 12” values appended to the end of the last observation, which can be discarded (See **Rolling 12** section below).

The **Multiplier** column is the multiplier to be applied to the value for proper scaling. Values transmitted via satellite (LRGS and NOAAPort records) are all transmitted as integers and must be scaled. For instance, a CRX_VN value might be transmitted as 2404, which should then be multiplied by .001 to get a value of 2.404. *This column is for transmitted values only. PDA records are already scaled.*

The **Stored Decimals** column is the number of decimal places with which the value is stored in our system. The `CRX_VN` value is stored with 3 decimal places, so its value would remain 2.404. For values transmitted with greater precision than what is stored, consistent rounding must always be used. See the **Rounding** section for details. If this column is empty, do not round; store the value with its current number of decimal places.

The **Element Name** column is the internal element name used in our systems. These names are arbitrary, but may appear elsewhere in our documentation.

The **Element Description** column is a description of the element that is intended to help users know what the value represents.

All possible elements for all streams can be found in CRN_Elements.csv.

Rolling 12

Transmitted records may contain 36 values appended to the end of the last observation. They represent temperature and precip values for the 12 most recent 5-minute periods at the time of transmission, as opposed to the 5-minute periods of the observation which start from the top of the hour. They are appended for use in weather forecasts only. They are ignored by the CRN software.

Differences between Transmitted and PDA

The values in a PDA record are already scaled, and do not require the multiplier column. Transmitted records (LRGS, NOAAPort) are transmitted as integers and need to be scaled.

The exception to the above rule is the `BV_UFL` element. For some streams, the `BV_UFL` value in PDA records is not scaled or is scaled incorrectly. A special table is used to determine what multiplier to use to scale `BV_UFL` values in PDA records. Any stream not included in the table can be assumed to have a properly scaled `BV_UFL` value.

Stream ID	Multiplier	Stored Decimals
1	0.1	2
3	0.1	2
6	0.1	2
7	0.01	2
51	0.1	2
52	0.1	2
53	0.1	2
54	0.1	2

Another difference is that LRGS and NOAAPort records typically contain either 2 or 3 observations followed by 36 “rolling 12 values” that are ignored. The first value of a new observation follows immediately after the last value of the previous observation. PDA records always contain one and only one observation with no "rolling 12" values at the end.

Handling Old Streams 1, 3, and 6

CRX Version Position

Some very old streams store `CRX_VN` as the 52nd value in the record. This is true for stream IDs 1, 3, and 6. If the 6th value of a record does not contain a valid `CRX_VN` value, then check the 52nd value in case the record is using a very old stream. If that value is also invalid, the record is corrupted and should be discarded.

CRX Version Uniqueness

Current practice is to change the CRX Version number any time there is a change to the datalogger program (and stream definition). Unfortunately, this best practice was not always followed in the early years of the CRN program. There are 4 CRX Version numbers that may describe 2 different stream definitions. The way to determine which stream definition file to use for these 4 CRX Version numbers is to determine the number of values per observation.

CRX Version	Stream ID	Number of values per ob
1.001	3	80
1.001	6	52
1.002	3	80
1.002	6	52
1.003	3	80
1.003	6	52
1.004	3	80
1.004	6	52

Using the table above, determine the stream ID by finding the CRX Version and the number of values per observation. The end of the first observation can be found by looking for the occurrence of the expected `YEAR`, `JULIAN_DAY`, `ZTIME` of the second observation in the record, or the end of the line if the record contains only 1 observation.

Record Format

Other oddities of very old streams include the fact that some are pre-scaled (all or most Multipliers are equal to 1), and values may have a + or - sign before each value.

Example of a very old record:

```
CD00134206001021422G53+0NN180W0001788"+2006.,+1.000,+200.0,+36.62,-116.0,...+0.000
```

Quality Control Flags

Flags are associated with element values from a station's datalogger (not calculated values) to indicate that a value is suspect. This also prevents a value from being used for calculations. A value may be flagged with any combination of the following flag types:

```
RANGE
DELTA
DOOR
FROZEN
SENSOR
```

Datalogger Door

If the datalogger door has been opened any time during the hour, that indicates that some type of maintenance was being done at the station and any of the measured values might have been compromised. We therefore flag all values (except for those few mentioned in the flag logic below) for that hour's observation with the `DOOR` flag.

To determine whether or not the datalogger door was open during the hour, we read the `ETDO` value, which is defined as *time in minutes datalogger door was open during hour*.

Flag logic: If `ETDO` > 0, then apply `DOOR` flag to all values in the observation except for `YEAR`, `JULIAN_DAY`, `ZTIME`, `LATITUDE`, `LONGITUDE`, and `ETDO`.

Range

Some measured values have an expected range that they should fall within to be considered legitimate measurements. If they fall outside of this range, they are flagged with the `RANGE` flag. All range flags are given the same RANGE flag, even for special cases of Dual-Fan Range and Rain Gauge Range (see below).

All measured values that have a range specified in the CRN_QC_Range_Params.csv file have the potential to have a range flag applied.

Flag logic: If (value < lower_limit OR value > upper_limit) then apply RANGE flag to the value.

Dual-Fan Range

Applies to `HCNFAN1`, `HCNFAN2`, `SEC_HCNFAN1`, `SEC_HCNFAN2`

Some stations have a single aspirated shield covering all three temperature sensors rather than one shield per sensor. These stations use a dual-fan system that uses a slightly different range check algorithm. This applies to any stations that use streams containing the elements named `HCNFAN1` and

HCNFAN2. Rather than a simple range check per fan value, these dual-fan systems use the following logic:

- If one fan is out of range, then do not apply the range flag to either fan value.
- If both fans are out of range, then apply the range flag to both fan values.

For dual-transmitter stations that have two sets of fans, it is slightly more complicated. This applies to streams containing elements SEC_HCNFAN1 and SEC_HCNFAN2. The following logic is used to flag the fan values:

- If one primary fan is out of range, then do not apply the range flag to either primary fan value.
- If both primary fans are out of range, then apply the range flag to both primary fan values.
- If both primary fans are flagged as out of range, then perform the following check on the secondary fans:
  - If one secondary fan is out of range, then do not apply the range flag to either secondary fan value.
  - If both secondary fans are out of range, then apply the range flag to both secondary fan values.

Rain Gauge Range

Applies to all Geonor wire depth elements: D105, D110,...D160, D205,...D260, D305,...D360

The upper range of a rain gauge depth measurement depends upon the size of the rain gauge, which can change for a given station over time as equipment is upgraded. As of 2012, we have 2 standard sizes: 600mm and 1000mm. The rain gauge size for a given datetime can be found in ISIS. If available, the size of the gauge is used as the upper range limit for the measured value. If it cannot be found, 600 is used as the default upper range. The lower range is taken from the CRN_QC_Range_Params.csv file for the specified element name.

Because the station metadata in ISIS is not typically updated immediately after a larger gauge is installed, a rain gauge depth measurement might be compared against the wrong upper range limit for several weeks, potentially resulting in erroneous range flags. These flags will be corrected the next time those observations are reprocessed.

Wetness Sensor Range

Applies to all wetness sensor elements: WET105,WET205,WET110,WET210,…WET160,WET260

A wet ness sensor instrument measures two separate wetness values, represented as WET1XX and WET2XX elements in our system (where the XX represents minutes during the hour). A wetness element is flagged as out of range if either it is itself outside of its valid range or if its corresponding wetness element during that same time period is outside of its valid range.

Examples:

- If WET105 is out of range, then WET105 and WET205 are both RANGE flagged.

- If `WET210` is out of range, then `WET110` and WET210 are both `RANGE` flagged.
- If `WET115` and `WET215` are out of range, then both are also `RANGE` flagged.

Delta

Applies to "temp sensor" and "secondary temp sensor" element values.

For a group of 3 measured temperature values, check that the difference between each value and the other 2 values in the group is no more than the allowed maximum delta of 0.3.

Flag logic: If a value is more than the allowed difference between BOTH corresponding values for the same time period, then flag it with the `DELTA` flag.

Example 1:
```
T105: 2.4 not flagged
T205: 2.5 not flagged
T305: 3.1 DELTA flag
```

Example 2:
```
T125: 2.7 not flagged
T225: 2.9 not flagged
T325: 3.1 not flagged
```

Example 3:
```
T110: 2.2 DELTA flag
T210: 3.5 DELTA flag
T310: 2.9 DELTA flag
```

The groups of elements used for the delta comparisons are the 3 temperature elements from the same time period. See CRN_QC_Delta_Params.csv for a full listing of element groups used for Delta flagging.

Note: There is a "delta" concept used for calculating precipitation (see below) but it is unrelated to the `DELTA` flags described in this section.

Frozen

Applies to soil moisture element values.

If it is determined that the soil was frozen at the site of a soil moisture probe, then the measurement from that probe is flagged with the `FROZEN` flag. If the soil temperature is determined to be less than 0.5 degrees Celcius, then the soil is considered frozen.

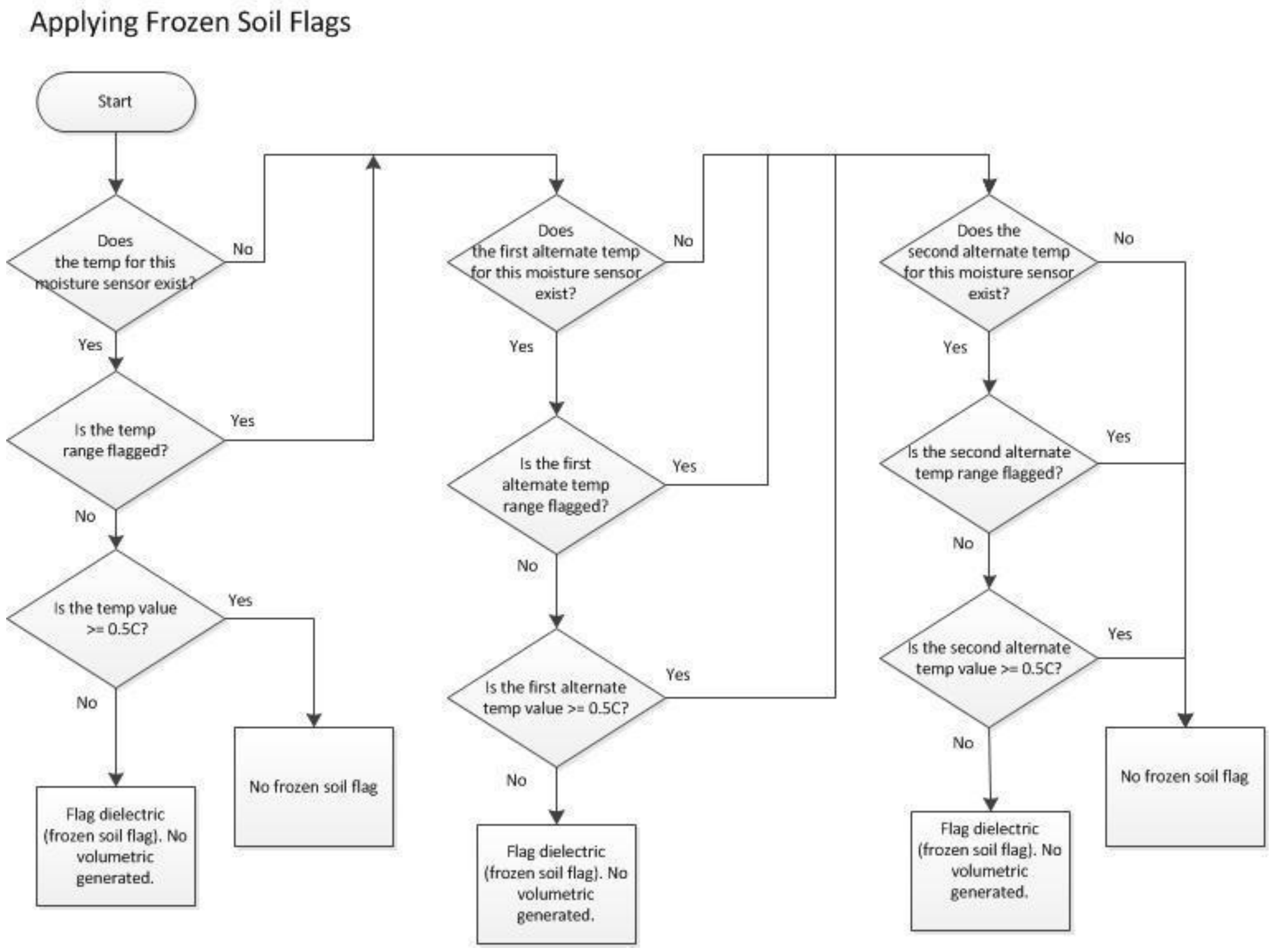
If the soil temperature measurement, at the same hole and time period, is present and not range flagged, it is used to determine if the ground was frozen. If the soil temperature measurement value is less than 0.5 degrees Celcius, then the soil is frozen and the `FROZEN` flag is applied.

If the soil temperature measurement is `RANGE` flagged, then the first alternative soil temperature value is used.

If the first alternative soil temperature value is RANGE flagged, then the second alternative temperature value is used.

If no usable soil temperature measurements exist for that soil moisture element value, then the soil moisture element value will not be flagged.

The flowchart below illustrates this logic. See the CRN_QC_Frozen_Soil_Params.csv file to determine the soil temperature, first alternative soil temperature, and second alternative temperature elements for a given soil moisture element.


Applying Frozen Soil Flags
Start
Does the temp for this moisture sensor exist?
No
Yes
Is the temp range flagged?
Yes
No
Is the temp value >= 0.5C?
Yes
No
Flag dielectric (frozen soil flag). No volumetric generated.
No frozen soil flag
Does the first alternate temp for this moisture sensor exist?
No
Yes
Is the first alternate temp range flagged?
Yes
No
Is the first alternate temp value >= 0.5C?
Yes
No
Flag dielectric (frozen soil flag). No volumetric generated.
Does the second alternate temp for this moisture sensor exist?
No
Yes
Is the second alternate temp range flagged?
Yes
No
Is the second alternate temp value >= 0.5C?
Yes
No
Flag dielectric (frozen soil flag). No volumetric generated.
No frozen soil flag

Sensor

Applies to any element specifically named in the QC-bad-sensor-list.csv file and any 5-minute elements related to an element in QC-bad-sensor-list.csv.

If a sensor at a station is determined by scientists to be faulty or “bad”, we flag all values measured by that sensor, even if they are within normal range. A sensor may be bad for the entire period of record for the station, or it may have gone bad and then been replaced by a good sensor. To accommodate this latter case, a datetime range is associated with a “bad sensor” and only measurements taken during that time period will be flagged.

Because a sensor is not known to be bad right away, and because we may not be aware immediately when a sensor is replaced, the QC-bad-sensor-list.csv is not updated in real-time, meaning that values that are from a bad sensor may not yet be flagged when the observation data is initially processed, but will be flagged (or unflagged in the case of sensors that have been removed from the list) retroactively after reprocessing.

Below are a few example entries in the QC-bad-sensor-list.csv. All lines beginning with the # sign are comments. The first value of each row is the internal stationId which can be looked up in the CRN_Stations.csv file. The second value is the name of the sensor, which can apply directly to an element name as well as apply to its related 5-minute elements. The third value is a begin datetime or “BOR” for “beginning of record”. The 4th value is an optional end datetime. An empty 4th value is treated as present time. Any extra values in a line are treated as comments.

```
#station,sensor,begin date,end date,comments
# LA Monroe 26,,,,,100cm and 50cm is probably in water table.,
1012,SM1100,BOR,,NO RECORD from 5/25/10 to 3/7/11,,
1012,SM1020,20120101,,,noisy,
1012,SM3020,20120101,,,noisy,
# NE Lincoln 8,,,,,,
1004,SM2005,BOR,20110618,,"continue to monitor",
```

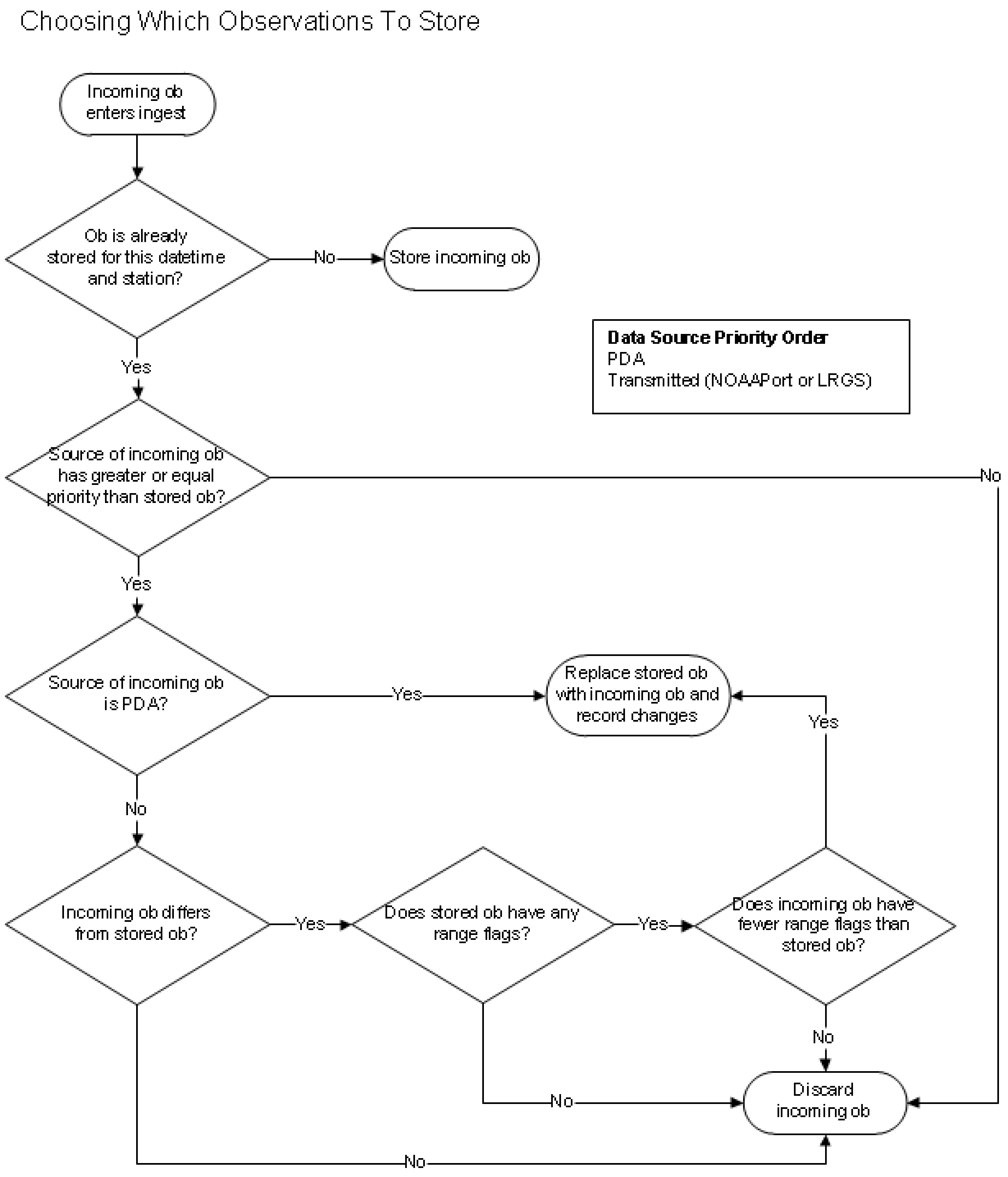
For the last Bad Sensor line, the sensor is `SM2005`, meaning soil moisture probe in hole 2 for depth of 5cm. This element name in a stream will represent an hourly value, but 5cm soil elements have hourly as well as 5-minute values. The 5-minute elements for this same sensor will have the names `SM2005_05`, `SM2005_10`, `SM2005_15`, `SM2005_20`, `SM2005_25`, `SM2005_30`, `SM2005_35`, `SM2005_40`, `SM2005_45`, `SM2005_50`, `SM2005_55`, `SM2005_60`. In this example, all 5-minute elements will be flagged with the `SENSOR` flag as well, even though only the hourly element name is listed in QC-bad-sensor-list.csv.

See QC-bad-sensor-list.csv for the full list, which can be viewed as a comma-separate values file or an Excel spreadsheet.

Eliminating Redundant Observations

Many layers of redundancy are built into our systems to ensure that data are not lost. As a result, incoming records files will contain several records representing the same observation (same station and datetime). Only one copy of an observation will be stored at any given time. If observation data are received that are determined to be of higher quality than previously received data for that same station and datetime, the earlier data are discarded and replaced with the new higher quality data.

The following logical flowchart is used to determine whether an incoming observation should replace a previously stored observation.


Choosing Which Observations To Store
Incoming ob enters ingest
Ob is already stored for this datetime and station?
No
Store incoming ob
Yes
Data Source Priority Order
PDA
Transmitted (NOAAPort or LRGS)
Source of incoming ob has greater or equal priority than stored ob?
No
Yes
Source of incoming ob is PDA?
Yes
Replace stored ob with incoming ob and record changes
Yes
No
Incoming ob differs from stored ob?
Yes
Does stored ob have any range flags?
Yes
Does incoming ob have fewer range flags than stored ob?
No
No
Discard incoming ob
No

Calculated Elements

When ingesting observation data, CRN performs several types of calculations for each observation and stores those values with the observation data. Each type of calculation is described below.

Precipitation

Calculated precipitation elements are:

- `P_OFFICIAL: calculated Geonor precip total for hour`
- `P5_1, P5_2,...P5_12: calculated Geonor precip for 5 minutes ending at :05, :10, …:60 (5-minute streams)`
- `P15_1, P15_2, P15_3, P15_4: calculated Geonor precip for 15 minutes ending at :15, :30, :45, :60 (15-minute streams from early years of CRN)`

Geonor wire depth element values and wetness sensor element values are used in determining precip (in millimeters) for 5-minute subhourly periods (or 15-minute periods for streams that contain only 15-minute Geonor wire depth values) and for the hour. A complex set of flowcharts follow, which illustrate the algorithm used for calculating these precipitation values.

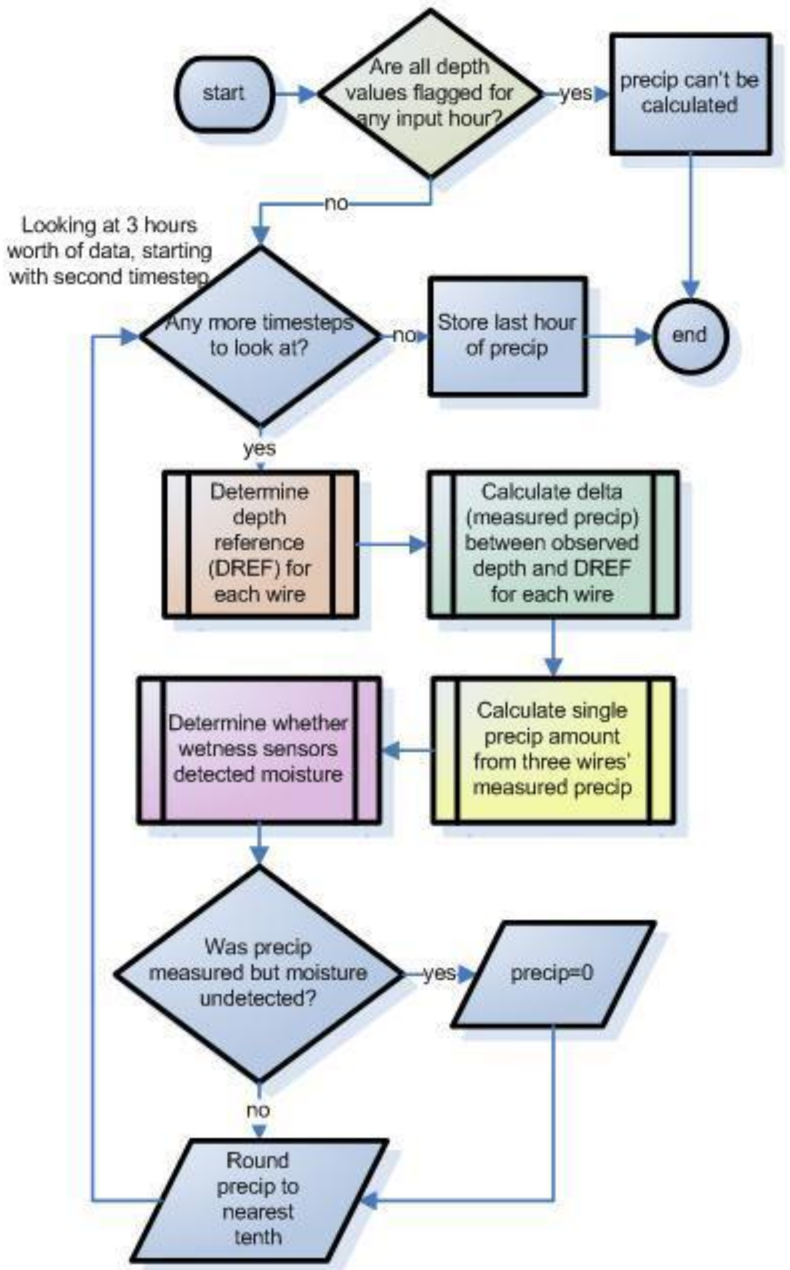
Notice that the first flowchart contains 4 blocks which are color-coded to a corresponding detailed flowchart.The flowcharts entitled **Calculate deltas**, **Determine whether moisture detected**, **Determine DREF**, and **Calculate precip from wires**, are all detailed views into portions of the first multi-colored flowchart, entitled **P5_i and P15_i**.

The section of the diagram entitled **Calculate precip from wires** contains a special case to handle heavy rainfall events in which the water in the bucket may be unevenly distributed. In the event that all three diffs are > MAX_DIFF, we take each diff and divide it by the sum of the deltas of its two wires. We then test the three diffs just as we did before, but against .1 rather than .2 (because we sum the deltas).

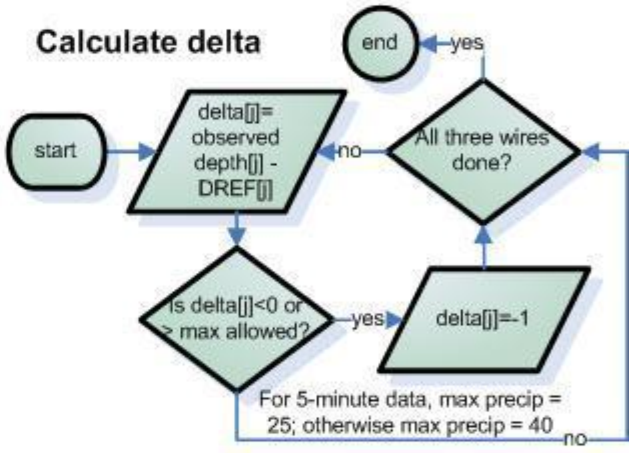
The very last flowchart is a simple diagram for calculating total precip from the calculated 5-minute or 15-minute precip calculations.

All calculated precipitation values are rounded to 3 decimal places but displayed publicly with only 1 decimal place.

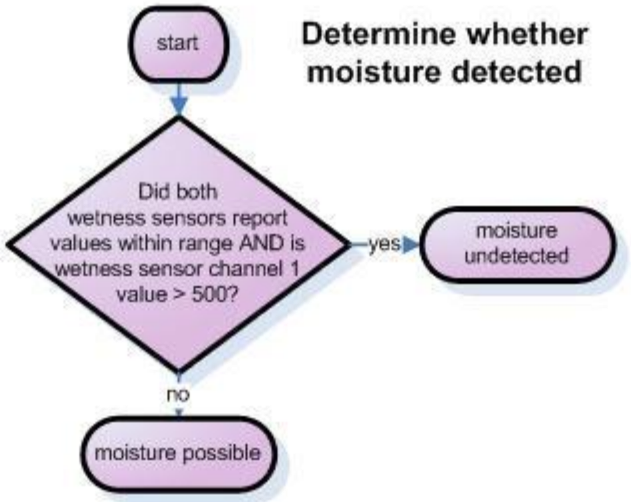
P5_i and P15_i
(5 minute and 15 minute precip)


start
Are all depth values flagged for any input hour?
yes
precip can't be calculated
no
Looking at 3 hours worth of data, starting with second timestep
Any more timesteps to look at?
no
Store last hour of precip
end
yes
Determine depth reference (DREF) for each wire
Calculate delta (measured precip) between observed depth and DREF for each wire
Calculate single precip amount from three wires' measured precip
Determine whether wetness sensors detected moisture
Was precip measured but moisture undetected?
yes
precip=0
no
Round precip to nearest tenth

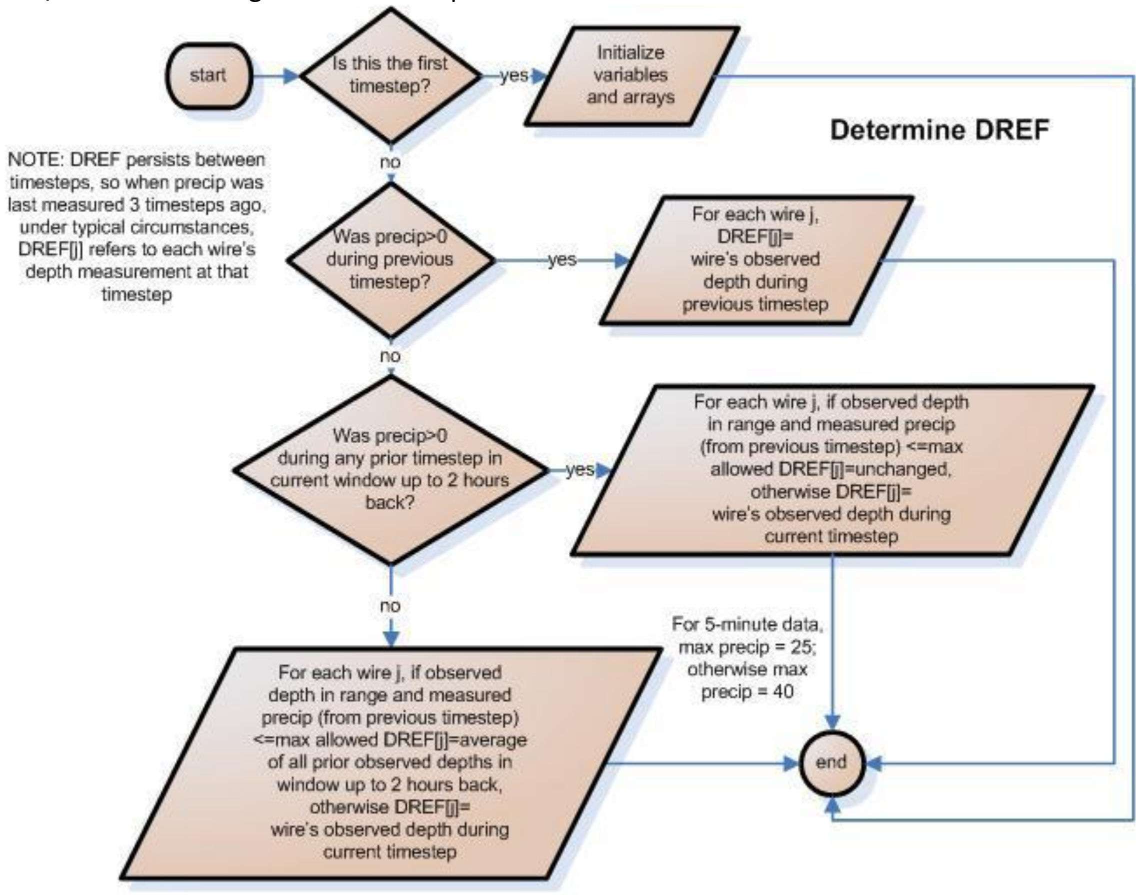
Calculate delta

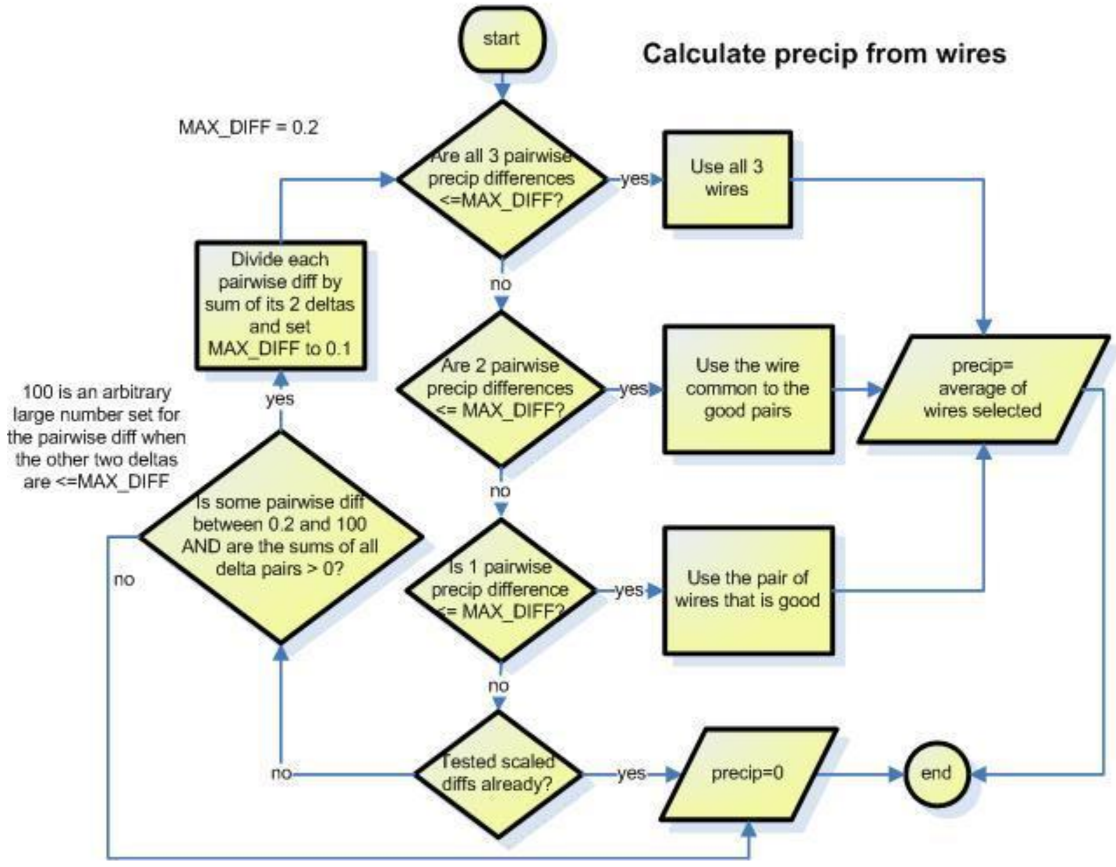

end
yes
start
delta[j]= observed depth[j] - DREF[j]
no
All three wires done?
Is delta[j]<0 or > max allowed?
yes
delta[j]=-1
For 5-minute data, max precip = 25; otherwise max precip = 40
no

Determine whether moisture detected

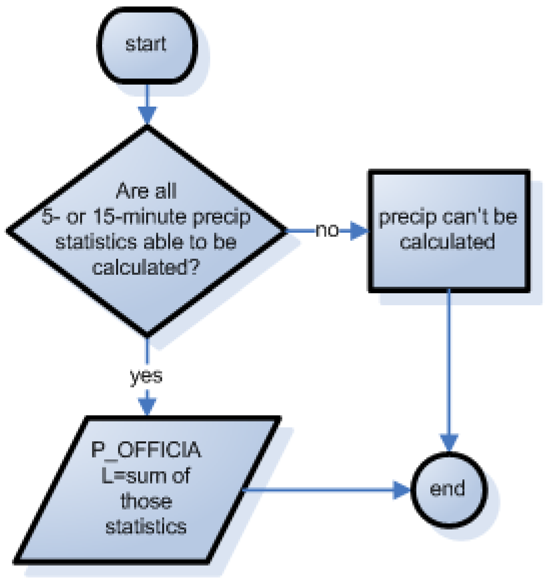

start
Did both wetness sensors report values within range AND is wetness sensor channel 1 value > 500?
yes
moisture undetected
no
moisture possible

The general idea behind this algorithm is that we're calculating 3 hours worth of depth observations but storing only the last hour. This is done so that between precipitation events instrument noise is smoothed during dry periods up to 2 hours. The calculation of the first 2 hours of precip is done entirely for the benefit of calculating the final hour.


Determine DREF
start
Is this the first timestep?
yes
Initialize variables and arrays
no
NOTE: DREF persists between timesteps, so when precip was last measured 3 timesteps ago, under typical circumstances, DREF[j] refers to each wire's depth measurement at that timestep
Was precip>0 during previous timestep?
yes
For each wire j, DREF[j]= wire's observed depth during previous timestep
no
Was precip>0 during any prior timestep in current window up to 2 hours back?
yes
For each wire j, if observed depth in range and measured precip (from previous timestep) <=max allowed DREF[j]=unchanged, otherwise DREF[j]= wire's observed depth during current timestep
no
For 5-minute data, max precip = 25; otherwise max precip = 40
For each wire j, if observed depth in range and measured precip (from previous timestep) <=max allowed DREF[j]=average of all prior observed depths in window up to 2 hours back, otherwise DREF[j]= wire's observed depth during current timestep
end


start
Calculate precip from wires
MAX_DIFF = 0.2
Are all 3 pairwise precip differences <=MAX_DIFF?
yes
Use all 3 wires
no
Divide each pairwise diff by sum of its 2 deltas and set MAX_DIFF to 0.1
Are 2 pairwise precip differences <= MAX_DIFF?
yes
Use the wire common to the good pairs
precip= average of wires selected
yes
100 is an arbitrary large number set for the pairwise diff when the other two deltas are <=MAX_DIFF
no
Is some pairwise diff between 0.2 and 100 AND are the sums of all delta pairs > 0?
no
Is 1 pairwise precip difference <= MAX_DIFF?
yes
Use the pair of wires that is good
no
no
Tested scaled diffs already?
yes
precip=0
end

Total Precip for Hour


start
Are all 5- or 15-minute precip statistics able to be calculated?
no
precip can't be calculated
yes
P_OFFICIAL=sum of those statistics
end

Note:
The CRN ingest code does check whether all subhourly components are present. However, the algorithm always produces all subhourly values or none.

Temperature

Calculated temperature elements are:

- `T_MAX:   calculated maximum temp for hour`
- `T_MIN:   calculated minimum temp for hour`
- `T_OFFICIAL:    calculated average temp for hour`
- `T5_1, T5_2, ...T5_12:  calculated average temp for 5 minutes ending at :05, :10, …:60`

All calculated temperature values are rounded to 3 decimal places but displayed publicly with only 1 decimal place. A detailed flowchart is included below, showing how hourly temperature calculations (`T_MAX, T_MIN, T_OFFICIAL`) are performed. 5-minute subhourly average values (`T5_1, T5_2,....T5_12`) are also calculated for each 5-minute period in the hour and included in the flowchart below.

CRN Calc Temp Inputs

For all temperature calculations, there are 3 specific temperature element values used to determine a particular calculated value, one for each of the 3 temperature sensors. CRN_Calc_Temp_Inputs.csv lists the temperature value inputs for each calculated value. For convenience, a chart containing the same information is also included below.

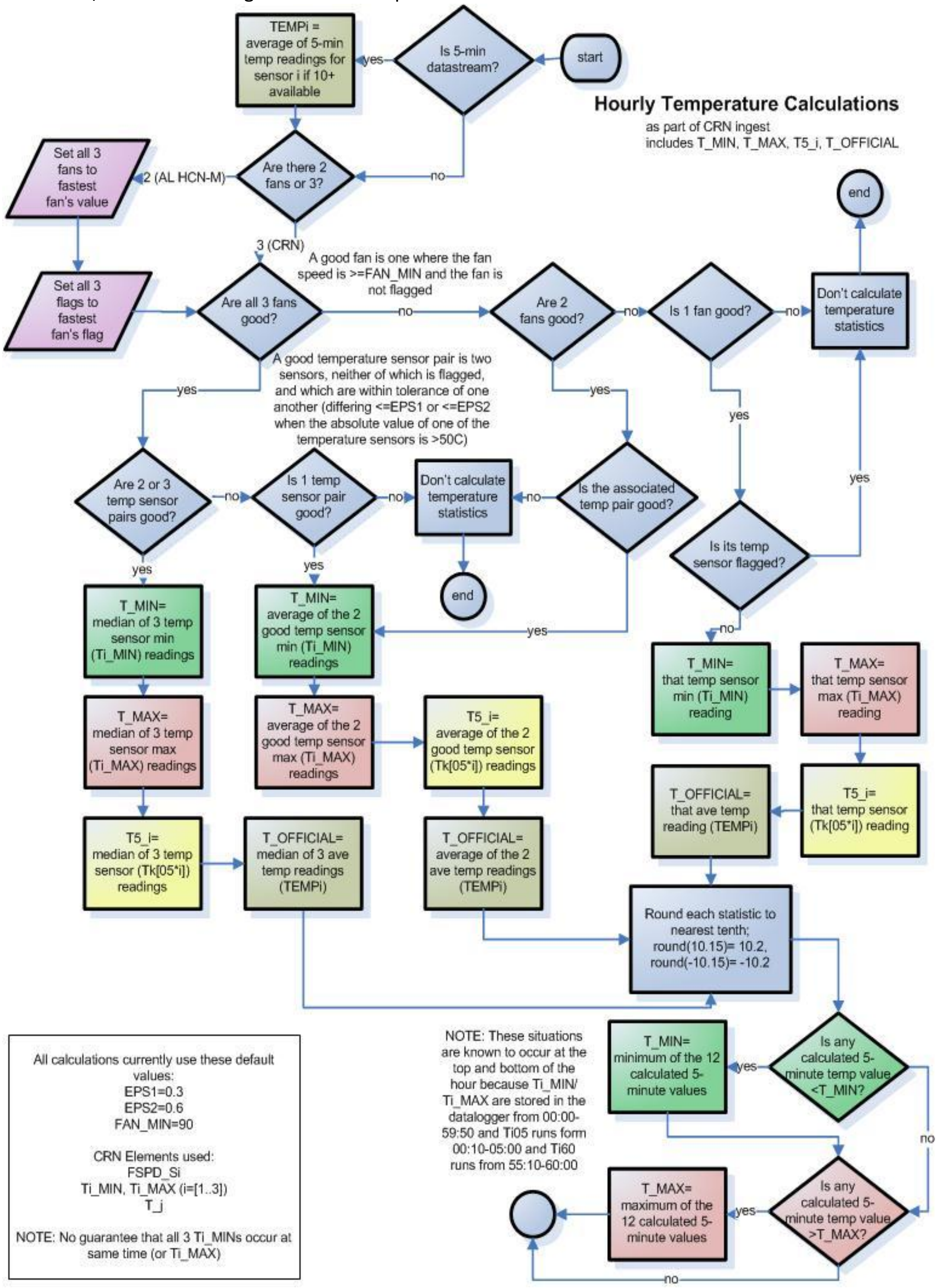
Calculated Element	Input1	Input2	Input3
T_OFFICIAL	TEMP1	TEMP2	TEMP3
T_MIN	T1_MIN	T2_MIN	T3_MIN
T_MAX	T1_MAX	T2_MAX	T3_MAX
T5_1	T105	T205	T305
T5_2	T110	T210	T310
T5_3	T115	T215	T315
T5_4	T120	T220	T320
T5_5	T125	T225	T325
T5_6	T130	T230	T330
T5_7	T135	T235	T335
T5_8	T140	T240	T340
T5_9	T145	T245	T345
T5_10	T150	T250	T350
T5_11	T155	T255	T355
T5_12	T160	T260	T360

Temperature inputs are only used to calculate values if a temperature value is unflagged *and* if its corresponding fan value is also unflagged. The remaining valid temperature inputs are used according to the flowcharts below.

Fan Value Inputs

USCRN stations generally have 3 temperature fans, one corresponding to each temperature sensor. Elements FSPD_S1, FSPD_S2, and FSPD_S3 represent these 3 hourly fan values for hourly and 5-minute temperature calculations.

USRCRN and some USCRN stations have only 2 fans, only one of which is expected to be running at any given time. Elements HCNFAN1 and HCNFAN2 represent these 2 hourly fan values. For these stations, take the higher fan speed value of the 2 fans, and use that fan value for all 3 temperatures.


start
Is 5-min datastream?
yes
no
TEMPi = average of 5-min temp readings for sensor i if 10+ available
Hourly Temperature Calculations
as part of CRN ingest
includes T_MIN, T_MAX, T5_i, T_OFFICIAL
Are there 2 fans or 3?
2 (AL HCN-M)
3 (CRN)
Set all 3 fans to fastest fan's value
Set all 3 flags to fastest fan's flag
Are all 3 fans good?
Are 2 fans good?
Is 1 fan good?
Don't calculate temperature statistics
end
Are 2 or 3 temp sensor pairs good?
Is 1 temp sensor pair good?
Don't calculate temperature statistics
Is the associated temp pair good?
Is its temp sensor flagged?
end
T_MIN= median of 3 temp sensor min (Ti_MIN) readings
T_MAX= median of 3 temp sensor max (Ti_MAX) readings
T5_i= median of 3 temp sensor (Tk[05*i]) readings
T_OFFICIAL= median of 3 ave temp readings (TEMPi)
T_MIN= average of the 2 good temp sensor min (Ti_MIN) readings
T_MAX= average of the 2 good temp sensor max (Ti_MAX) readings
T5_i= average of the 2 good temp sensor (Tk[05*i]) readings
T_OFFICIAL= average of the 2 ave temp readings (TEMPi)
T_MIN= that temp sensor min (Ti_MIN) reading
T_MAX= that temp sensor max (Ti_MAX) reading
T5_i= that temp sensor (Tk[05*i]) reading
T_OFFICIAL= that ave temp reading (TEMPi)
Round each statistic to nearest tenth; round(10.15)= 10.2, round(-10.15)= -10.2
Is any calculated 5-minute temp value <T_MIN?
T_MIN= minimum of the 12 calculated 5-minute values
Is any calculated 5-minute temp value >T_MAX?
T_MAX= maximum of the 12 calculated 5-minute values

A good fan is one where the fan speed is >=FAN_MIN and the fan is not flagged

A good temperature sensor pair is two sensors, neither of which is flagged, and which are within tolerance of one another (differing <=EPS1 or <=EPS2 when the absolute value of one of the temperature sensors is >50C)

NOTE: These situations are known to occur at the top and bottom of the hour because Ti_MIN/ Ti_MAX are stored in the datalogger from 00:00-59:50 and Ti05 runs form 00:10-05:00 and Ti60 runs from 55:10-60:00

All calculations currently use these default values:
EPS1=0.3
EPS2=0.6
FAN_MIN=90

CRN Elements used:
FSPD_Si
Ti_MIN, Ti_MAX (i=[1..3])
T_j

NOTE: No guarantee that all 3 Ti_MINs occur at same time (or Ti_MAX)

Soil

Soil moisture and temperature values may be hourly or 5-minute subhourly values. Each soil probe has a layer depth (5cm, 10cm, 20cm, 50cm, 100cm) and a hole number (1, 2, or 3). There are 3 probes per layer, 1 probe per hole, for a total of *up to* 15 probes per station. Each probe measures soil temperature and soil moisture. Some stations do not have probes installed at every depth.

`SoilCalc.java` contains static methods for performing soil calculations.

Soil Moisture

Calculated soil moisture elements are:

- `SMV005, SMV010, SMV020, SMV050, SMV100:` hourly soil moisture volumetric layer average at each depth
- `SMV1005, SMV2005,SMV3005,....SMV1100, SMV2100, SMV3100:` hourly soil moisture volumetric average for each hole at each layer depth for the hour
- `SMV005_05, SMV005_10,...SMV005_60:` soil moisture volumetric layer average *at 5cm depth only*, for each subhourly 5-minute period.
- `SMV1005_05, SMV2005_05, SMV300_05,...SMV3100_60:` Soil moisture volumetric for each hole, for each depth, for each subhourly 5-minute period

Each soil moisture dielectric value is used to calculate a volumetric soil moisture value. The details of this calculation can be found in
`SoilCalc.calculateVolumetric(BigDecimal dielectricValue)`

The calculation uses the following logic:

- If dielectric is present and unflagged
  - If dielectric is less than or equal to 2.7, then volumetric = 0
  - Else, volumetric = (sqrt(dielectricValue) * 0.109) - 0.179

From the calculated volumetric moisture values, layer averages are calculated. For each group of 5-minute subhourly volumetric values per layer depth (all 3 holes), an average is calculated and then rounded to 3 decimal places. For each group of hourly volumetric values per layer depth, an average is calculated and then rounded to 3 decimal places. Subhourly moisture values are not used by CRN software to calculate hourly values.

Soil Temperature

Calculated soil temperature elements are:

- `SMT005, SMT010, SMT020, SMT050, SMT100 (`hourly soil temperature layer average at each depth)
- `SMT005_05, SMT005_10,...SMT005_60 (`soil moisture volumetric layer average *at 5cm depth only*, for each subhourly 5-minute period)

From the measured soil temperature values, layer averages are calculated. For each group of hourly temperature values per layer depth (3 holes), an average is calculated and then rounded to 3 decimal places. For each group of 5-minute subhourly temperature values (3 holes) *for 5cm layer depth only*, an average is calculated and then rounded to 3 decimal places. Subhourly soil temperature values are not used by CRN software to calculate hourly values.

Only soil temperature values that are unflagged are used to calculated soil temperature layer averages.

Rounding

As of September 12, 2012, following the advice of the National Data Stewardship Team, all rounding in CRN is asymmetric half-up rounding.

Examples:

- 1.234 rounded to 2 decimals is 1.23
- 1.235 rounded to 2 decimals is 1.24
- -1.234 rounded to 2 decimals is -1.23
- -1.235 rounded to 2 decimals is -1.23

Prior to this date, all rounding in CRN was symmetric half-up rounding.

Examples:

- 1.234 rounded to 2 decimals is 1.23
- 1.235 rounded to 2 decimals is 1.24
- -1.234 rounded to 2 decimals is -1.23
- -1.235 rounded to 2 decimals is -1.24

After the next reprocessing of the period of record, all values after December 31, 2003 will be updated to reflect the new asymmetric half-up rounding.

BigDecimal vs. double or float

For all calculations performed, no matter how trivial, the Java object type `BigDecimal` is used. Primitive types `double` and `float` are never used because they do not accurately and precisely represent floating point numbers. `BigDecimal` is an exact way of representing numbers and must always be used. If not using Java for development, the language equivalent of Java's `BigDecimal` must be used.

One exception to this rule exists in the soil moisture volumetric calculation (`SoilCalc.calculateVolumetric`) where a double is used when taking its square root. BigDecimal does not support square root because the loss in precision from taking the square root is larger than the loss in precision from using double.

Definitions

ASCII: Human-readable character set composed largely of the characters found on a standard English keyboard.

ATDD Number: a 3-digit or 4-digit integer used by ATDD and CRN processing software to associate a PDA record with a specific station.

Calculated Element: An element derived during processing from element values sent from a station's datalogger.

CRX Version: The identifying version number of the software installed at a station's datalogger.

Datalogger: The device used at a station to capture all the instrument readings at a station for transmission.

Datetime: A date and hour combination. Each observation represents 1 datetime of data for 1 station.

Element: A specific measured or calculated value. All elements have a name and description and can be found in the CRN_Elements.csv file.

GOES ID: An 8-character code used by the GOES satellite and CRN processing software to associate a transmitted record with a specific station.

ISIS: The database which stores CRN station metadata.

LRGS: A transmitted file or record, typically received in pseudo-binary format. LRGS files typically contain 10 minutes of transmissions from any stations which have transmitted data during that 10-minute period.

NOAAPort: A transmitted file or record, typically received in pseudo-binary format. NOAAPort files typically contain 1 hour of transmissions from all stations.

Observation: Data values recorded at a station during 1 hour.

PDA: A type of file or record, in ASCII format, that is downloaded directly from a station's datalogger.

Pseudo-binary: A compressed format used for transmitting data via satellite that requires decoding software in order to be able to read.

Quality Control Flag: An indicator associated with an element value in an observation, indicating the value is untrustworthy in some way.

Record: In the case of transmitted raw files, a single transmission from a station containing observation data for one or more consecutive hours. In the case of PDA files downloaded from the datalogger, one line of text representing a single hour of observation data.

Stream: The expected output format from a station datalogger. The output is a comma-separated list of element values in a particular order, representing an observation or multiple consecutive (usually in descending datetime order) observations.

Change Log

Significant changes to ingest code or processes that are likely to impact observed values:

2012/09/12 Following the advice of the National Data Stewardship Team, all rounding in CRN is asymmetric half-up rounding.

2013/04/18 Following approval from the Configuration Control Board for Change Request #44, paired wetness sensor values will both be flagged as out of range if either one is out of range.

2013/07/10 Calculated temperature and precipitation will be stored with 3 decimal places in order to prevent rounding bias when using these values in daily, monthly, and other calculations. Because calculated temperature and precipitation are being derived from values that are rounded to the hundredths place before transfer to NCDC, the third decimal place stored for these calculated values is not traditionally a significant digit, but is necessary to preventing a positive bias in further calculation results that are rounded up. Calculated soil moisture and soil temperature will continue to be stored with 3 decimal places, as further calculations with these variables yield results to 2 decimal places.