SSSG dsLog Ini Files

Tim Thiel

2013-01-11 - Rev 003

Scope of document

The dsLog software is used to log and redistribute various, RS232, RS422, RS485, and UDP data streams. The operation of the dsLog software is controlled by specifying a user-generated initialization file. This document will provide the information needed to create an appropriate initialization file.

Random dsLog Facts

- Specified destination directories for disk logging must be created by the user. dsLog will NOT create these directories.
- Initialization file syntax is strictly interpreted. For instance, if spaces exist before or after the '=' in a line of the file it will be ignored.

dsLog Status Messages

The dsLog process creates a file with information about the status of the data logging process. This file has a suffix of LGS and is placed in the root logging directory.

LGS messages are added to the file regularly to give basic statistics about each of the input threads. The syntax for these messages is:

LGS timestamp LGR INPUT_1_INFO INPUT_2_INFO … INPUT_N_INFO

For each defined data Input there is a section of data defined as:

Sn TimeSinceLastData NumberOfMessagesReceived NumberOfBytesReceived NumberOfReadErrors

Here is an example:

LGS 2013/01/11 10:46:28.190 LGR S1 1.97 52 2460 0 S2 4.32 3 69 0

In this example, for the [INPUT_1] stream it has been 1.97 seconds since data has been received; 52 messages have been received; 2460 bytes of data have been received; and there have been zero read errors. For the [INPUT_2] stream it has been 4.32 seconds since data has been received; 3 messages have been received; 69 bytes of data have been received; and there have been zero read errors.

DSR messages are added to the file which give information on the disk space available for each disk based destination. The syntax for these messages is:

DSR timestamp LGR INPUT_1_INFO INPUT_2_INFO … INPUT_N_INFO

For each defined data Input there is a section of data defined as:

Sn numberOfDestinations DEST_1_INFO DEST_2_INFO … DEST_N_INFO

For each destination there will be three associated fields. If the destination is a disk type destination then the fields will be:

- Error_Count – at this time this is unimplemented and will always be zero.
- Percentage of free space remaining
- Remaining space in bytes

If the destination is not a disk type destination then the three associated fields will be "NOT_DISK 0.0 0".

Here is an example:

DSR 2013/01/11 10:46:48.189 LGR S1 4 0 92.6 1362468093952 NOT_DISK 0.0 0 NOT_DISK 0.0 0 NOT_DISK 0.0 0 NOT_DISK 0.0 0 S2 2 0 92.6 1362468093952 NOT_DISK 0.0 0

The 'S1' field starts the output related to the '[INPUT_1]' section. The next field indicates that there are four defined destinations for that input. The first defined INPUT_1 destination is a disk so there are fields indicating the error count, percentage of space remaining, and amount of available space. The remaining three destinations associated with INPUT_1 are UDP type, so they each show the NOT_DISK field and zeroes. Finally there are similar outputs for the two destinations associated with '[INPUT_2]' in the fields following the 'S2' string.

Initialization File Sections

The initialization file consists of several sections. Each of these sections is delimited at the top of the section by a header line which contains the name of the section surrounded by square brackets (e.g. ***[INPUT_1]***).

The sections are enumerated below with brief descriptions:

- [GENERAL]
- [INPUT_n]
- [GUI_THREAD]
- [TCP_SERVER_THREAD]

[GENERAL] section

- **ROOT_DIR**
  - Type – string
  - Stored as – startup.rootDirectory

- Valid values – A string representing a valid path on the host machine.
- Description – All logged data files will be placed in this directory.

- **READ_FROM_INI**
  - Type – int
  - Stored as – Not applicable
  - Valid values – Not applicable
  - Description – Apparently an unimplemented function. Appears in template ini files but not in source. Will check with Jon Howland about intentions.
- **INI_SERVER_IP_ADDRESS**
  - Type – string
  - Stored as – Not applicable
  - Valid values – Not applicable
  - Description – Apparently an unimplemented function. Appears in template ini files but not in source. Will check with Jon Howland about intentions.
- **INI_SERVER_SOCKET**
  - Type – int
  - Stored as – Not applicable
  - Valid values – Not applicable
  - Description – Apparently an unimplemented function. Appears in template ini files but not in source. Will check with Jon Howland about intentions. [INPUT_x] section

[INPUT_n] section

- **DATA_TYPE**
  - Type – string
  - Stored as – loggingList[x].dataType
  - Valid values – Any unique three character string.
  - Description – Typically this is a three character string. It is appended to the ASCII log file names after a dot (‘.’) character. For each received string the corresponding log file entry will contain the dataType string as the leftmost field.
- **DATA_SOURCE**
  - Type – string
  - Stored as – loggingList[x].sourceName
  - Valid values – An ASCII string.
  - Description – For each received string the corresponding log file entry will contain the sourceName string after the timestamp and before the received data.
- **SOURCE_TYPE**
  - Type – int
  - Stored as – loggingList[x].sourceType
  - Valid values – 0 (UDP_ASCII), 1 (UDP_BINARY), 2 (SERIAL_ASCII), 3 (SERIAL_BINARY)
  - Description – Used to specify whether data is received via a serial port or a UDP connection and whether the data is ASCII or binary.

- **TIMESTAMP**
    - Type – int
    - Stored as – loggingList[x].timeStampFlag
    - Valid values – 0, 1
    - Description – Specifies whether logged data will have a timestamp included in the records.
- **IN_SOCKET**
    - Type – int
    - Stored as – loggingList[x].inSocketNumber
    - Valid values – Any valid UDP socket number.
    - Description – Specifies the UDP socket upon which the UDP data will be received.
- **SERIAL_PORT_NAME**
    - Type – string
    - Stored as – loggingList[x].theSerialPort.sio_com_port_name
    - Valid values – A valid serial port on the host machine. (e.g. "/dev/ttyS0", "/dev/ttyS1").
    - Description – Specifies the serial port to use to recive logged data and (if specified) send queries.
- **PARITY**
    - Type – int
    - Stored as – loggingList[x]. theSerialPort.parity
    - Valid values – 0 (NONE); 1 (ODD); 2 (EVEN).
    - Description – Specifies the parity setting for the associated serial port.
- **BAUD**
    - Type – int
    - Stored as – loggingList[x]. theSerialPort.baud
    - Valid values – Any baud rate supported by the hardware serial port and system software.
    - Description – Specifies the baud rate for the associated serial port.
- **DATA_BITS**
    - Type – int
    - Stored as – loggingList[x]. theSerialPort.data_bits
    - Valid values – Any number of data bits supported by the hardware serial port and system software. Typically 8 which is the default value.
    - Description – Specifies the number of data bits for the associated serial port.
- **STOP_BITS**
    - Type – int
    - Stored as – loggingList[x]. theSerialPort.stop_bits
    - Valid values – Any number of stop bits supported by the hardware serial port and system software.
    - Description – Specifies the number of stop bits for the associated serial port.
- **ECHO**

- Type – int (Boolean)
- Stored as – loggingList[x]. theSerialPort.echo
- Valid values – 0 and 1.
- Description – If set to a non-zero value then received characters will be echoed out on the associated serial port.

- **FLOWCONTROL**
  - Type – int
  - Stored as – loggingList[x]. theSerialPort.flow_control
  - Valid values – 0 (NONE); 1 (XONXOFF); 2 (RTSCTS); 3 (XONXOFF_RTSCTS)
  - Description – Specifies the flow_control setting for the associated serial port.
- **TERM_CHAR**
  - Type – string
  - Stored as – loggingList[x]. theSerialPort.auto_mux_enabled
  - Stored as – loggingList[x]. theSerialPort.auto_mux_term_char
  - Valid values – “none”; “\r”; “\n”; the hexadecimal representation of a single character (“0xnn”); a single character enclosed in double-quotes.
  - Description – the auto_mux_enabled value is set to true unless “none” is specified. The auto_mux_term_char is set the specified ASCII character. The default value for auto_mux_term_char is ‘\r’.
- **PROMPT_SEQUENCE_INTERVAL**
  - Type – double
  - Stored as – loggingList[x].promptSequence.secondsBetweenSequences
  - Valid values –
  - Description – This value specifies the number of seconds that will elapse between sequences of specified prompt transmissions.
- **PROMPT_SEQUENCE_INTERVAL_BETWEEN_PROMPTS**
  - Type – double
  - Stored as – loggingList[x]. promptSequence.secondsBetweenPrompts
  - Valid values –
  - Description – This value specifies the number of seconds that will elapse between individual prompts in a prompt sequence, unless ‘USE_PROMPT_SPECIFIC_REPLY_WINDOWS’ is defined and set to 1.
- **PROMPT_SEQUENCE_NUMBER_OF_PROMPTS**
  - Type – int
  - Stored as – loggingList[x]. promptSequence.numberOfPrompts
  - Valid values –
  - Description – This value specifies the number of user specified prompts contained in the prompt sequence for this input.
- **PROMPT_n_CHAR_xx**
  - Type – Hexadecimal
  - Stored as – loggingList[x].promptSequence.prompts[n].promptChars[xx]

- Valid values –
- Description – Using hexadecimal, this value specifies the xx-th character of the n-th prompt in the prompt sequence. For example if the fourth character of the second prompt in the sequence is the digit 6 then the following line would be used:
  - PROMPT_2_CHAR_4=36

- **USE_PROMPT_SPECIFIC_TERM_CHAR**
  - Type – int
  - Stored as – loggingList[x].usePromptSpecificTermChar
  - Valid values – 0 or 1.
  - Description – If set to 0 then incoming characters will be written to the logger destinations every time the character specified by the "TERM_CHAR" is received. If set to one then the user can specify a specific term_char to be associated with each prompt string. This allows the use of devices with different response syntaxes to be used on a single input channel.
- **PROMPT_n_TERM_CHAR**
  - Type – Hexadecimal
  - Stored as – loggingList[x].promptTermChar[n]
  - Valid values – Any ASCII character.
  - Description – Ignored unless USE_PROMPT_SPECIFIC_TERM_CHAR is set to 1. After the nth prompt is sent on the interface the logger will buffer characters until the specified term_char is received. The reception of this term_char will trigger the logging of a data record.
- **USE_PROMPT_SPECIFIC_DATA_TYPE**
  - Type – int
  - Stored as – loggingList[x]. usePromptSpecificDataType
  - Valid values – 0 or 1.
  - Description – If set to 0 then incoming records will be written to the logger destinations with the INPUT_x 'DATA_TYPE'. If set to one then the user can specify a specific DATA_TYPE to be associated with each prompt string.
- **PROMPT_n_DATA_TYPE**
  - Type – string
  - Stored as – loggingList[x].promptDataType[n]
  - Valid values – Any unique three character string.
  - Description – Typically this is a three character string. If USE_PROMPT_SPECIFIC_DATA_TYPE is unspecified or set to zero each log file entry for data on this input will contain the 'DATA_TYPE' string as the leftmost field. If set to one then the data records received in reply to prompt n will contain the 'PROMPT_n_DATA_TYPE' string as the leftmost field.
- **USE_PROMPT_SPECIFIC_SOURCE_NAME**
  - Type – int
  - Stored as – loggingList[x]. usePromptSpecificSourceName

- Valid values – 0 or 1.
- Description – If set to 0 then incoming records will be written to the logger destinations with the INPUT_x 'SOURCE_NAME'. If set to one then the user can specify a specific SOURCE_NAME to be associated with each prompt string.

- **PROMPT_n_SOURCE_NAME (change to DATA_SOURCE???)**
  - Type – string
  - Stored as – loggingList[x].promptSourceName[n]
  - Valid values – Any unique string.
  - Description –If USE_PROMPT_SPECIFIC_SOURCE_NAME is unspecified or set to zero each log file entry for data on this input will contain the 'DATA_SOURCE' string as the field after the timestamp and before the received data. If set to one then the data records received in reply to prompt n will contain the 'PROMPT_n_SOURCE_NAME' string as the field after the timestamp and before the received data.
- **USE_PROMPT_SPECIFIC_SYNCOPATION**
  - Type – int
  - Stored as – loggingList[x]. usePromptSpecificSyncopation
  - Valid values – 0 or 1.
  - Description – If unspecified or set to 0 then all of the prompts in the sequence will be sent each time the prompt sequence is triggered. If set to one then the user must specify the associated syncopation for each prompt in the sequence.
- **PROMPT_n_SYNCOPATION**
  - Type – int
  - Stored as – loggingList[x].promptSyncopation[n]
  - Valid values – Any integer.
  - Description – Ignored unless USE_PROMPT_SPECIFIC_SYNCOPATION is set to 1. This setting allows the user to specify that certain prompts in the sequence will not be transmitted every time the prompt sequence is triggered. The transmission of the associated prompt will be skipped PROMPT_N_SYNCOPATION times after each time the prompt is actually sent. So to send the prompt every cycle set PROMPT_N_SYNCOPATION to zero; to send it every other cycle set it to one; to send it every tenth cycle set PROMPT_N_SYNCOPATION to 9.
- **USE_PROMPT_SPECIFIC_REPLY_WINDOWS**
  - Type – int
  - Stored as – loggingList[x]. usePromptSpecificReplyWindows
  - Valid values – 0 or 1.
  - Description – If unspecified or set to 0 then the prompts in the sequence will be sent at a fixed cadence specified by 'PROMPT_SEQUENCE_INTERVAL_BETWEEN_PROMPTS'. If set to one then the user must specify the associated reply_window for each prompt in the sequence.
- **PROMPT_n_REPLY_WINDOW**
  - Type – int

- Stored as – loggingList[x].promptReplyWindows[n]
- Valid values – Any integer.
- Description – If USE_PROMPT_SPECIFIC_REPLY_WINDOWS is unspecified or set to zero then the individual prompts in the prompt sequence will be sent at a fixed cadence specified by 'PROMPT_SEQUENCE_INTERVAL_BETWEEN_PROMPTS'. If set to one then the n+1th prompt will be sent PROMPT_n_REPLY_WINDOW seconds after the nth prompt.

- **USE_PROMPT_SPECIFIC_DESTINATIONS_FLAG**
  - Type – int
  - Stored as – loggingList[x]. usePromptSpecificDestinationsFlag
  - Valid values – 0 or 1.
  - Description – Specifies whether the logged data on this channel will be sent to all of the associated destinations or if the user can specify destinations to be associated with each prompt.
- **PROMPT_n_DESTINATION_ENABLE_nn**
  - Type – int
  - Stored as – loggingList[x].promptDestinationEnable[n][nn]
  - Valid values – 0 or 1.
  - Description – If USE_PROMPT_SPECIFIC_DESTINATIONS_FLAG is unspecified or set to zero then all data records on this input will be sent to all of the defined destinations for this input. If set to one then the data records received in reply to prompt n will be sent only to the destinations with the PROMPT_n_DESTINATION_EANBLE_nn set to one.
- **DESTINATION_n**
  - Type – int
  - Stored as – loggingList[x].destinations[n].destinationType
  - Valid values – 0 (UDP_SOCKET); 1 (DISK)
  - Description – Specifies the type of the logging destination (disk vs. network).
- **DESTINATION_n_IPADDRESS**
  - Type – string
  - Stored as – loggingList[x]. destinations[n].networkDestination.ipAddress
  - Valid values – A double quote delimited string specifying the IP_ADDRESS of the UDP destination (e.g. "192.168.1.44").
  - Description – For network destination types this variable specifies the IP_ADDRESS of the destination.
- **DESTINATION_n_SOCKET**
  - Type – int
  - Stored as – loggingList[x]. destinations[n].networkDestination.toSocketNumber
  - Valid values – A valid socket number.
  - Description – For network destination types this variable specifies the SOCKET_NUMBER of the destination.
- **DESTINATION_n_SOCKET_STARTS_WITH**

- Type – string
- Stored as – loggingList[x]. destinations[n].destinationFilter.startsWith
- Valid values –
- Description -

- **DESTINATION_n_SOCKET_STARTS_CONTAINS**
  - Type – string
  - Stored as – loggingList[x]. destinations[n]. destinationFilter.greps
  - Valid values –
  - Description -
- **DESTINATION_n_SOCKET_STARTS_DOESNT_CONTAIN**
  - Type – string
  - Stored as – loggingList[x]. destinations[n]. destinationFilter.grepMinusVs
  - Valid values –
  - Description -
- **DESTINATION_n _PATHNAME**
  - Type – string
  - Stored as – loggingList[x]. destinations[n].loggingDestination.loggingDirectory
  - Valid values –
  - Description – For disk type destinations, this value will specify the base directory for logged data. It will default to the directory specified in [GENERAL]->ROOT_DIR.

[GUI_THREAD] section (TBD)

- **PORT**
  - Type – int
  - Stored as –
  - Valid values –
  - Description -
- **IPADDRESS**
- **TO_IP_ADDRESS**
- **TO_PORT**
- **DESTINATION_N**
  - Description -

[TCP_SERVER_THREAD] section (TBD)

- **TCP_SERVER_THREAD_SOCKET**
  - Type – int
  - Stored as –
  - Valid values –
  - Description –

Document History

- R001 – 2012-12-15
  - This version documented the code as it was delivered by Jon Howland
- R002 – 2012-12-26
  - Added this Document History section.
  - Modified to reflect changes in code made by T. Thiel.
- R003 – 2013-01-11
  - Added section describing dsLog status output messages.
-