LONMARK® Device Interface File Reference Guide

Revision 4.402
May 2009

Introduction

LONMARK device interface (XIF) files are files that define the network-visible interface for one or more LONWORKS® devices. The *device interface* is the interface to a device that is exposed over a LONWORKS network. The device interface does not expose the internal algorithms of a device. Instead, it only exposes the inputs to the algorithms and the outputs from the algorithms. The device interface includes the device's self-documentation information, the number of address table entries, the number of message tags, and the number, types, and directions of network variables.

Much of the device interface can be queried over the network by a network tool. The device manufacturer determines the completeness of a queried interface. For example, a device manufacturer may choose to embed network variable names in a device to ensure that the queried network interface includes these names.

There are two benefits to using device interface files. First, a device interface file may include information that is not included in a device such as network variable names. Second, a device interface file can be used during network engineering when the device is not accessible from the network engineering tool.

The primary device interface file type is a text file with a **.xif** extension. Some platforms such as the LNS™ network operating system may convert this file to alternate formats for performance optimization. For example, LNS uses a binary device interface file (**.xfb** extension) and an optimized device interface file (**.xfo** extension). Both of these files are created from the data contained within the text device interface file. This document describes the format of the text device interface file. The XIF32BIN Device Interface File Conversion Utility is used to convert a text device interface file to a binary device interface file. The optimized device interface file is created automatically by LNS to reduce the access time to data within a device interface file. Other network operating systems may create their own optimized versions of the device interface file.

Device interface files are typically generated by LONWORKS® development tools. Many of the fields of the device interface file for a device must match the application in the device. If a device interface file is modified in such a way that it does not match the application it is documenting, installation errors may occur for the device.

Revision History

The following table lists the major changes in each format version of the device interface file.

Version	*Changes*
1.0	First version.
2.0	Allow a network variable array to be described by a single network variable description record, instead of one per element. Other transaction and size parameters added.
3.0	Add a comment indicator. String fields contain an asterisk if they are not applicable or they are default values. Integer fields contain zero when they are not applicable and asterisks when they are default values.
3.1	Add support for Neuron® Chip firmware version 6 (including revised binding constraints).
3.2	Add a network variable count that includes dynamic network variables.
4.0	Introduce additional rules to reduce the chances of backward compatibility problems in future revisions. New records introduced in 4.x or later XIF files must be followed by a blank line and 4.x interpreters should discard unknown records and their contents up to the next blank line. Also, starting with format 4.0, the maximum line length has been fixed at 160 characters. Any XIF interpreters should be able to handle up to 160 characters in a XIF input line. Any XIF interpreter that claims to accept version X.Y should also accept the known parts of any file of version X.Z, where Z > Y, ignoring any data fields on any line beyond the expected end of the data line for version X.Y.
4.1	Same content as 3.2 but in the backward compatible format.
4.2	Add fields for devices that support the extended network management command set (ECS). ECS is defined by the ANSI/EIA/CEA-709.1-B protocol and allows devices to have more address table entries, and to be a member of more groups.
4.3	Add fields that identify the version number and capabilities of the Neuron firmware used by the device.
4.400	Add a field that identifies the base clock rate factor to be used by the device. Changed the minor format version number to 3 digits.
4.401	Add fields to support dynamic functional blocks. Clarified requirements for duplicate programmatic NV names.
4.402	Modified to support increased network variable and alias limits. Add a field that identifies whether LNS FX is to deduct a credit when the device is commissioned.

Text Device Interface File Format

A text device interface file consists of the following sections:

- Header
- Network variable and message tag definitions
- File definitions (added in version 4.0)
- Network variable value definitions (added in version 4.0)

All sections are optional, except for the header section. These sections must be in the specified order, and are described in the following sections. Following are a few general rules that apply to all sections:

- If the first non-blank character on a line is '#', the entire line is ignored. This means that comment lines may be inserted anywhere, since they do not count as blank lines.
- Multiple blank lines are allowed anywhere a single blank line is required, blank lines may appear between individual network variable or message tag records and at the end of the file, and blank characters are allowed at the beginning of any line.
- In general, string fields contain an asterisk if they are not applicable or they are default values. Integer fields contain zero when they are not applicable, and asterisks when they are default values.
- The maximum line length for any line is 160 characters.

Header Section

The header section is the first section of the device interface file, and is the only required section. The header describes some basic information about the capabilities of the device, such as the transceiver type and buffer configuration.

Installation tools may use the transceiver type information to determine if a device is compatible with its intended channel. This usage is optional. An installation tool may use the device interface file solely for program definition and may ignore the transceiver type information.

Following is an example of a header section. The lines are numbered for reference in this document; these line numbers are not included in the device interface file.

```
1:  File: Main.XIF generated by LONNCC32E Version 5.00.22, XIF Version 4.400
2:  Copyright (c) Echelon Corporation 1989-2009
3:  All Rights Reserved.  Run on Fri Apr 10 13:58:29 2009
4:
5:  9F:FF:FF:05:01:84:04:30
6:  2 15 1 22 0 3 3 3 3 3 3 11 11 11 11 7 0 13 16 1 1 128 22 0 0 0 0 0 0 0 0 2 15 0 0 0 0 0 2 49 0
7:  32 6 18 13 28 1118 0 15 5 3 236 4 10000000
8:  1 7 1 1 4 4 4 15 200 0
9:  78125 0 0 0 0 0 252 0 0 0 0 0
10: 90 0 240 0 0 0 40 40 0 5 22 9 26 43 44
11: *
12: "&3.4@0NodeObject,4[2Lamp,2[2Switch,1010LightSensor,1040TempS
13: "ensor,1Joystick
```

The header section consists of the following lines (the Version column identifies the minimum XIF format version required to support the entry):

Line	*Version*	*Contents*
Line 1	All	File name, source of the file, and format version number. This document describes format version 4.400. The format of the string must be as follows: **File:** *fileName* **generated by** *toolName***, XIF Version** *majorNumber***.***minorNumber* If the file was manually generated, specify the *toolName* as **Manual 0.0.0**. For version 4.400, specify the *majorNumber.minorNumber* as **4.400**.
Line 2	All	Copyright information.
Line 3	All	Optional additional copyright information plus a required timestamp of when the file was created. The format of the string must be as follows: *optionalInfo* **Run on** *day month date hour***:***min***:***sec year*
Line 4	All	Blank line.
Line 5	All	Program ID. This consists of eight 2-digit hex values, separated by colons (no spaces). The first hex digit identifies the program ID format. If the first digit is 7 or less, the format is an ASCII string, typically with the name of the program. If the first digit is 8 or 9, the format is the following: *FM***:***MM***:***MM***:***CC***:***CC***:***UU***:***TT***:***NN* The fields of the type 8 or 9 program ID are described in the *LONMARK Application-Layer Interoperability Guidelines*.
Line 6	All	Contains the following fields:
	All	Field 1 — Number of non-ECS domains. **Must be set to 2.** For ECS devices, set line 6 field 33 below to the actual number of domains. May be set to 1 for devices that are not LONMARK certified.
	All	Field 2 — Number of non-ECS address table entries. Set to 0 to 15 for non-ECS devices; for ECS devices, set to the actual number of address table entries or 15 (whichever is less) and set line 6 field 34 below to the actual number of address table entries.
	All	Field 3 — Boolean that specifies whether the application handles incoming application messages. Set to 1 if the application handles incoming application messages, otherwise set to 0.
	All	Field 4 — Number of static network variable declarations in the application. Network variables arrays count as one declaration even though each array element counts as one network variable. Set to 0 to 4096.

All	Field 5	Number of non-ECS message tags. Set to 0 to 15 for non-ECS devices; for ECS devices, set to the actual number of message tags or 15 (whichever is less) and set line 6 field 35 below to the actual number of message tags.
All	Field 6	Number of network input buffers. Encoded as follows:

Count	***Encoded Value***
0	0
1	2
2	3
3	4
5	5
7	6
11	7
15	8
23	9
31	10
47	11
63	12
95	13
127	14
191	15

All	Field 7	Number of network output buffers. Encoded as described under network input buffers (field 6).
All	Field 8	Number of priority network output buffers. Encoded as described under network input buffers (field 6).
All	Field 9	Number of priority application output buffers. Encoded as described under network input buffers (field 6).
All	Field 10	Number of application output buffers. Encoded as described under network input buffers (field 6).
All	Field 11	Number of application input buffers. Encoded as described under network input buffers (field 6).

All	Field 12	Network input buffer size. Encoded as follows:

Size	*Encoded Value*
20	2
21	3
22	4
24	5
26	6
30	7
34	8
42	9
50	10
66	11
82	12
114	13
146	14
210	15
255	0

All	Field 13	Network output buffer size. Encoded as described under network input buffer size (field 12).
All	Field 14	Application output buffer size. Encoded as described under network input buffer size (field 12).
All	Field 15	Application input buffer size. Encoded as described under network input buffer size (field 12).
2.0	Field 16	Application type, encoded as follows.

Value	*Type*
0	Unknown
1	MIP application without a host application; no network variables or message tags
2	Neuron hosted application; 62 network variables and 62 aliases maximum

		3	Host application with host selection of network variables (both ECS and non-ECS); 4096 network variables and 8192 aliases maximum
		4	Host application with network interface selection of network variables
		5	Reserved
		6	Host application with network interface selection of network variables; 254 network variables and 127 aliases maximum.
		7	Neuron hosted application; 254 network variables and 127 aliases maximum.
2.0	Field 17	Size of the network variable configuration table for MIP applications (not host applications) with network interface selection enabled. Set to 0 for all other applications, including host applications, ECS applications, and Neuron hosted applications.	
2.0	Field 18	Number of receive transaction buffers.	
3.1	Field 19	Number of network variable alias table entries provided by the device. Set to 0 to 8192.	
3.1	Field 20	Boolean that specifies whether relaxed binding constraints are allowed. If 0, each output network variables must use a unique network variable selector. If 1, multiple output network variables can share the same selector, as long as they are not polled by an input network variable. For non-ECS devices, set to 1 if two output network variables on the same device that are not polled by an input network variable can use the same network variable selector, otherwise set to 0. For non-ECS host-based applications using host selection, this should in general be set to 0. You can set an application to use host selection by writing the value 3 to field 16, described above. For ECS devices, set this field to 1 and then set the binding constraint number field below (field 26) to match the constraints of the device. For Neuron hosted applications and host-based applications using network interface selection, this should be set to match the capabilities of the Neuron firmware.	
3.1	Field 21	Specifies whether the statistics-relative address references are allowed. Set to 1.	
3.1	Field 22	Maximum size memory block that may be written at a time. For devices with flash memory, this is the flash sector size. For other devices, this value is 11 bytes.	
3.2/4.1	Field 23	Maximum number of network variables this device supports, which is equal to the number of static network variables	

		defined in field 4 plus the maximum number of dynamic network variables supported by the application. This can be no greater than 4096 and must be greater than or equal to the number of static network variables given in field 4.
4.2	Field 24	Minimum network management protocol version number. Set to 0.
4.2	Field 25	Maximum network management protocol version number. Set to 1 for devices that support ECS commands. Otherwise, set to 0.
4.2	Field 26	Binding constraint level. Set to 2 if two output network variables on the same device that are not polled by an input network variable may use the same network variable selector (in this case, field 20 should be set to 1). Otherwise, set to 1 (in this case, field 20 is set to 0).
4.2	Field 27	ECS flag 0. Set to 0 for non-ECS devices. Set to the encoded decimal value of the following bits for ECS devices:

Bit	***Flag Description***
0 (0x01)	Fixed static NV flag. Do not set this bit if the name, self-documentation string, and rate estimates of static NVs are configurable via the **UPDATE_NV_INFO** ECS command. Set this bit if the name, self-documentation string, and rate estimates of static NVs are not configurable.
1 (0x02)	Incoming group restricted flag. Set this bit if incoming groups are restricted to the non-ECS address table entries.
2 – 6	Bits 2 through 6 are reserved. **Set to 0.**

		7 (0x08)	Non-unique dynamic NV names flag. Set to 0 for a device that does not support dynamic network variables and on a device that supports dynamic network variables but requires their names to be unique on the device; set to 1 for a device that supports dynamic network variables and also supports dynamic network variables with duplicate names. When creating a dynamic network variable on a device that supports duplicate dynamic network variable names, a network management tool must ensure that the name is unique within the functional block containing the network variable, including all the static and dynamic network variables within the functional block. When creating a dynamic network variable that is not a member of a functional block, the network management tool must ensure that the name is unique for all the static and dynamic network variables that are not members of functional blocks. Network management tools may restrict or prevent the generation of duplicate dynamic NV names.
4.401	Field 28	ECS flag 1. Set to 0 for non-ECS devices. Set to the encoded decimal value of the following bits for ECS devices:	
		Bit	***Flag Description***
		0 (0x01)	Suppress dynamic NV definition flag. Set to 0 for a device that does not support dynamic network variables and on a device that supports dynamic network variables and also supports the ANSI/CEA-709.1-B network management commands to define dynamic network variables; set to 1 for a device that supports dynamic network variables and does not support the ANSI/CEA-709.1-B network management commands to define dynamic network variables. A device that supports dynamic network variables must also specify a value in field 23. **Set to 0 for LONMARK certified devices.**
		1 (0x02)	Suppress dynamic functional block definition flag. Set to 0 for a device that does not support dynamic functional blocks; set to 1 for a device that supports dynamic functional blocks. A device that supports dynamic functional blocks must also specify a value in field 43.

		2 (0x04) Suppress dynamic functional block member definition flag. Set to 0 for a device that does not support dynamic functional blocks; set to 1 for a device that supports dynamic functional blocks. A device that supports dynamic functional blocks must also specify a value in field 43. 3 (0x08) Dynamic NVs supported on static functional blocks flag. Set to 0 for a device that does not support dynamic functional blocks and on a device that supports dynamic functional blocks but does not support adding dynamic network variables to static functional blocks; set to 1 for a device that supports dynamic functional blocks and also supports adding dynamic network variables to static functional blocks. A device that supports dynamic functional blocks must also specify a value in field 43. 4 – 7 Bits 4 through 7 are reserved. **Set to 0.**
4.2	Fields 29–32	Reserved. **Set to 0**.
4.2	Field 33	Number of domains. Set this to the value in field 1.
4.2	Field 34	Number of address table entries. For non-ECS devices, set this to the value in field 2.
4.2	Field 35	Number of message tags. For non-ECS devices, set this to the value in field 5.
4.2	Field 36	Reserved. **Set to 0**.
4.2	Field 37	Reserved. **Set to 0**.
4.2	Field 38	Reserved. **Set to 0**.
4.2	Field 39	Reserved. **Set to 0**.
4.3	Field 40	The network management version number of the device. Set to 1 if the version number of the device's Neuron firmware is 13 or lower. Set to 2 if the version number of the device's Neuron firmware is 14 or higher.
4.3	Field 41	The network management capabilities of the device. Set to 0 if the version number of the device's Neuron firmware is 13 or lower. Set to 1 if the version number of the device's Neuron firmware is 14 or higher.
4.400	Field 42	Reserved. **Set to 0.**
4.401	Field 43	The number of dynamic functional blocks supported by the device. Set to 0 if dynamic functional blocks are not

supported. Devices that support dynamic functional blocks must also specify a value in field 28.

Line 7 — Describes the Neuron processor configuration. Line 7 contains the fields described below. Set fields 1 – 12 to 0, and set field 13 to 10000000, for host-based devices where the network image is not downloadable.

All — Field 1 — Protocol processor model. Encoded as follows:

Value	*Model*
0	Neuron 3150 Chip or FT 3150 Smart Transceiver
1	PL 3150 Smart Transceiver
8	Neuron 3120 Chip
9	Neuron 3120E1 Chip
10	Neuron 3120E2 Chip
11	Neuron 3120E3 Chip
12	Neuron 3120A20 Chip
13	Neuron 3120E5 Chip
14	Neuron CY3120E4 Chip or FT 3120 Smart Transceiver
15	PL 3120-E4 Smart Transceiver
16	Neuron CY7C53120L8 Chip
17	PL 3170 Smart Transceiver
32	FT 5000 Smart Transceiver
33	Neuron 5000 Chip
128	Not a Neuron Chip or Smart Transceiver

All — Field 2 — Protocol processor clock rate. The value of this field will be used in conjunction with the base clock rate factor (field 13) and a 0.5 multiplier to determine the base clock rate of the MAC-layer processor for the device. Encoded as follows:

Value	*Rate*
1	625 kHz
2	1.25 MHz

		3 2.5 MHz 4 5 MHz 5 10 MHz 6 20 MHz 7 40 MHz
3.0	Field 3	System firmware major revision number encoded as a decimal integer value.
3.0	Field 4	Receive transaction block size in bytes.
3.0	Field 5	Transaction control block size in bytes.
3.0	Field 6	Number of bytes of on-chip RAM from the end of the system area that precedes the receive transaction blocks to the first user variable or the end of on-chip RAM, whichever comes first.
3.0	Field 7	Number of bytes of off-chip RAM from the end of the available RAM that may be used by the Neuron Chip firmware to the first user variable or the end of off-chip RAM, whichever comes first.
3.0	Field 8	Domain table entry size in bytes.
3.0	Field 9	Address table entry size in bytes.
3.0	Field 10	Network variable configuration table entry size in bytes.
3.0	Field 11	Number of bytes from the beginning of the domain area up to the first byte of user code in EEPROM.
3.1	Field 12	Network variable alias table entry size in bytes. Set to 0 if aliases are not supported in the device.
4.4	Field 13	The base clock rate factor. Must be set to either 10000000 or 13107200. This value combined with the protocol processor clock rate (field 2) determines the base clock rate of the device. If the base clock rate factor is 10000000, the device clock rate is equivalent to the protocol processor clock rate defined in field 2. If the base clock rate factor is 13107200, the device clock rate is equivalent to the protocol processor clock rate defined in field 2, multiplied by a factor of 1.31072. For example, if field 4 is set to use value 4 (5 MHz), and the base clock rate factor is set to 13107200, the actual device clock rate is 6.5536 MHz.
4.4	Field 14	LNS Credit Indicator. Set to 0 or an empty field if LNS FX must deduct a credit when commissioning this device. Set to 1 if the device does not require LNS credits, and LNS

FX should not charge when commissioning the device. LNS device credits and LonMaker credits are not deducted when commissioning 5000 Series devices and devices based on PL Smart Transceivers.

This field is only used for estimating commissioning costs in the engineering phase of a system design. The actual charging of LNS device credits and LonMaker credits is determined by the LNS Server.

Line 8 3.0 Describes the channel parameters. Contains the following fields:

Field 1 Boolean that specifies whether a standard transceiver type is used. Set to 1 if a standard transceiver type is used, otherwise set to 0.

Field 2 Standard transceiver type ID. ID values are listed in the **std_id** field of the StdXcvr.xml file available on the LONMARK Web site at www.lonmark.org.

Field 3 Reserved. **Set to 1**.

Field 4 Transceiver interface type. Encoded as follows:

Value	***Type***
0	Not specified
1	Single ended
2	Special purpose
5	Differential

Field 5 Transceiver interface rate. Encoded as follows:

Value	***Rate***
0	1.25 Mbps
1	625 kbps
2	312.5 kbps
3	156.3 kbps
4	78.1 kbps
5	39.1 kbps
6	19.5 kbps
7	9.8 kbps

8	4.9 kbps
9	2.4 kbps
10	1.2 kbps
11	0.6 kbps

Field 6 Number of priority slots on the channel (0 – 127).

Field 7 Minimum clock rate for the channel. Encoded with the same values as the clock rate in line 7 field 2.

Field 8 Average packet size in bytes.

Field 9 Protocol processor oscillator accuracy in parts per million.

Field 10 Protocol processor oscillator wakeup time in microseconds.

Line 9 3.0 Describes the transceiver parameters. Contains the following fields:

Field 1 Channel bit rate in bits per second.

Field 2 Special purpose mode alternate channel bit rate in bits per second. Set to 0 for devices that do not use special purpose mode transceivers.

Field 3 Boolean that specifies whether a special purpose mode transceiver controls the preamble. Set to 1 if the transceiver controls the preamble, otherwise set to 0. Set to 0 for devices that do not use special purpose mode transceivers.

Field 4 Special purpose mode wakeup pin direction. Set to 0 for input, 1 for output. Set to 0 for devices that do not use special purpose mode transceivers.

Field 5 Boolean that specifies whether the device can override the general purpose data used for special purpose mode. Set to 1 if the device can override, otherwise set to 0. Set to 0 for devices that do not use special purpose mode transceivers.

Fields 6 – 12 General purpose data used for special purpose mode. Set to 0 for devices that do not use special purpose mode transceivers.

Line 10 3.0 Describes the channel timing parameters. Contains the following fields. All field values are in tenths of a bit time, except as noted.

Field 1 Receive start delay.

Field 2 Receive end delay.

Field 3 Indeterminate time.

		Field 4	Minimum interpacket time.
		Field 5	Preamble length.
		Field 6	Turnaround time (microseconds).
		Field 7	Missed preamble time.
		Field 8	Packet qualification time.
		Field 9	Boolean that specifies whether raw data overrides the timing values. Set to 1 if raw data overrides, 0 otherwise.
		Field 10	Raw data clock rate. Encoded with the same values as the clock rate in line 7 field 2.
		Fields 11 – 15	Raw data bytes for the communications parameters.
Line 11	3.0	Contains a single asterisk indicating the end of the transceiver parameters.	
Lines 12 – N	All	Device self-documentation string. If the documentation string is not supplied, there is a single line containing a single asterisk. If supplied, the documentation lines each begin with a double-quote character (not part of the documentation string). Multiple lines must be concatenated without any intervening characters. There is no end double-quote, instead the line is terminated by a newline. The characters of the string must all be printable ASCII characters (this includes spaces, but not tabs). Trailing spaces are included. The line may be up to 60 characters long, not including the starting double-quote character or the newline. Any non-printable characters must be encoded using an ANSI C hex character escape sequence of "**\x***HH*" where *H* represents a single hexadecimal digit. The values A – F within a hex character escape sequence must be specified with upper case letters exclusively. If the static interface contains functional blocks, the device self-documentation string must be formatted as described in *The LONMARK Interoperability Guidelines*.	
Line N+1	All	Blank line.	

Network Variable and Message Tag Definition Section

This section consists of zero or more network variable or message tag definitions. The number of network variable definitions that follow must be the same as the number of static network variable declarations specified in field 4 of line 7 of the header.

Network Variable Definition

Following is an example of a network variable definition. The lines are numbered for reference in this document; these line numbers are not included in the device interface file.

```
1: VAR nvo01Value 2 0 0 0
2: 0 1 63 1 0 1 0 1 0 1 0 0 0
3: "@1|2
4: 95 *   2
5: 1 0 0 0 0
6: 1 0 0 1 0
```

A network variable definition consists of the following lines:

Line	*Version*	*Contents*	
Line 1		A line with the following syntax:	
		VAR *name index avgRate maxRate arraySize*	
		The fields are defined as follows:	
	All	*name*	The network variable name (maximum of 16 characters). This name is also called the *programmatic name*. The name must be unique within the functional block containing the network variable. If the network variable is not a member of a functional block, the name must be unique for all the network variables that are not members of functional blocks. Development tools may restrict or prevent the generation of duplicate programmatic names.
	All	*index*	The network variable index specified as a decimal string (0 – 4095).
	All	*avgRate*	The average rate estimate specified as an encoded decimal string (0 – 250). Encoded as an unsigned decimal n, where the rate estimate = $2^{(n/8)-5}$. Set to 0 if the estimate is not specified.
	All	*maxRate*	The maximum rate estimate specified as an encoded decimal string (0 – 250). Encoded as an unsigned decimal n, where the rate estimate = $2^{(n/8)-5}$. Set to 0 if the estimate is not specified.
	2.0	*arraySize*	The number of network variables in a network variable array, or 0 if this network variable is not an array. Each element of a network variable array is assigned a unique network variable index number. The network variable index number for the entry following that for an array must be equal to the index number of the first element of the array plus the number of elements in the array.
Line 2	All	Contains the following fields:	
		Field 1	Specifies whether the device should be taken offline before updating the variable. Set to 0 if the variable can be

		updated when online or offline, or 1 if it should be updated only when offline.
	Field 2	Must be set to 1.
	Field 3	Must be set to 63.
	Field 4	Network variable direction. Set to 0 for an input, 1 for an output.
	Field 5	Default service type to use for connections containing this variable. Set to 0 for acknowledged, 1 for repeated, or 2 for unacknowledged.
	Field 6	Specifies whether the service type can be changed in the field. Set to1 if the type can be changed, 0 if it cannot.
	Field 7	Specifies the authentication default for the network variable. Set to 1 to use authentication for the network variable by default, 0 to not use authentication by default.
	Field 8	Specifies whether the use of authentication can be changed in the field. Set to 1 if the use of authentication can be changed, 0 if it cannot.
	Field 9	Specifies the default use of priority for the network variable. Set to 1 to use priority for the variable by default, 0 to not use priority by default.
	Field 10	Specifies whether the use of priority can be changed in the field. Set to 1 if the use of priority can be changed, 0 if it cannot.
	Field 11	Specifies the polled attribute of the network variable. For an input, set to 0 if the application program does not poll using this variable, 1 if it does. For an output, set to 0 if the network variable sends unsolicited updates, 1 if the network variable must be polled for updates.
	Field 12	Specifies the synchronized attribute of the network variable. Set to 0 if the network variable is not synchronized, 1 if the network variable is synchronized (i.e. all outputs are transmitted and their order is preserved).
	Field 13	Specifies the configuration attribute of the network variable. Set to 0 for a non-configuration class network variable; 1 for a configuration class network variable.
Lines 3 – N	All	This line and the following lines define the network variable's self-documentation. If the variable has no self-documentation, the line contains a single asterisk. If supplied, one or more lines of text appear here; each line begins with a double-quote character and ends with a newline. When the lines are concatenated together without the double-quote or newline characters, this forms the self-documentation text. Each line may be up to 60 characters long not including the double-quote or newline. Any non-printable

characters must be encoded using an ANSI C hex character escape sequence of "**\x***HH***"** where *H* represents a single hexadecimal digit. The values A – F within a hex character escape sequence must be specified with upper case letters exclusively.

If the variable is part of a functional block, the variable's self-documentation string must be formatted as described in *The LONMARK Interoperability Guidelines*.

Line N+1 All

The first line after the self-documentation provides network variable type information. The line has the following syntax:

snvtIndex ***** *elementCount*

The fields are defined as follows:

snvtIndex Specifies the SNVT index (1 to 255) or 0 if this variable is a user-defined network variable type. See the *SNVT and SCPT Master List* for a list of SNVT indexes.

elementCount Number of elements (1 to 256) in a network variable structure or union. Set to 1 if the network variable is not a structure or union.

Lines N+2 – M All

Network variable characteristics. If the network variable is not a structure or union, there is just one line. If the network variable is a structure or union, there is one line for each data element of the structure or union.

Each line has the following syntax:

type offset size signedFlag arraySize

The fields are defined as follows:

type Network variable data type. One of the following values:

Value	Data Type
0	Character
1	8-bit Integer (Neuron C short)
2	16-bit Integer (Neuron C long)
3	Bitfield
4	Union
5	Typeless. None of the remaining fields are applicable.

offset Network variable bitfield offset (0 to 7). Set to 0 if the network variable is not a bitfield.

size	Number of bits in a network variable bitfield (1 to 7), or the number of bytes in a union (1 to 31). Set to 0 if the network variable is not a bitfield or union.
signedFlag	Set to 0 if the network variable type is unsigned, 1 if signed. Set to 0 if not applicable.
arraySize	Set to 0 if the network variable type is not an array or, if the type is an array, the size of the array (1 to 31 bytes).

Following are several example network variable declarations and the corresponding device interface file definitions. See the *Neuron C Programmer's Guide* for a description of network variable declarations.

Example 1:

```
network output polled long
   bind_info(offline ackd(nonconfig) authenticated(nonconfig)
   priority(nonconfig) rate_est(123) max_rate_est(234)) outvar;


VAR outvar 0 69 76 0
1 1 63 1 0 0 1 0 1 0 1 0 0
*
0 *   1
2 0 0 1 0
```

Example 2:

```
network input sync config int invar;


VAR invar 1 0 0 0
0 1 63 0 0 1 0 1 0 1 0 1 1
*
0 *   1
1 0 0 1 0
```

Example 3:

```
typedef struct {
   int x;
   long y;
   int array[5];
   unsigned z : 3;
   unsigned zz : 5;
   union {
      int a;
      int b;
   } u;
} group;

network input group ingroup;


VAR ingroup 2 0 0 0
0 1 63 0 0 1 0 1 0 1 0 0 0
*
0 *   6
1 0 0 1 0
```

```
2 0 0 1 0
1 0 0 1 5
3 0 3 0 0
3 3 5 0 0
4 0 1 0 0
```

Message Tag Definition

Following is an example of a message tag definition. The lines are numbered for reference in this document; these line numbers are not included in the external interface file.

```
1: TAG user_tag 0 69 76 0
2: 0 1 63 1 0 1 0 1 0 1 0 0 0
```

A message tag definition consists of the following lines:

Line	*Version*	*Contents*	
Line 1		A line with the following syntax: **TAG** *name index avgRate maxRate zero* The fields are defined as follows:	
	All	*name*	The tag name (maximum of 16 characters).
	All	*index*	The message tag index specified as a decimal string (0 – 14).
	All	*avgRate*	The average rate estimate specified as an encoded decimal string (0 – 250). Encoded as an unsigned decimal n, where the rate estimate = $2^{(n/8)-5}$. Set to 0 if the estimate is not specified.
	All	*maxRate*	The maximum rate estimate specified as an encoded decimal string (0 – 250). Encoded as an unsigned decimal n, where the rate estimate = $2^{(n/8)-5}$. Set to 0 if the estimate is not specified.
	2.0	*zero*	Set to 0.
Line 2	All	A line with the following syntax: **0** *bindFlag* **63 1 0 1 0 1 0 1 0 0 0** The bindFlag field specifies whether the tag is bindable. Set to 1 if it is, 0 if it is not. **In general, this should be set to 1.**	

Following is an example message tag declaration and the corresponding device interface file definition. See the *Neuron C Programmer's Guide* for a description of message tag declarations.

```
msg_tag bind_info(rate_est(123) max_rate_est(234)) user_tag;
```

```
TAG user_tag 0 69 76 0
0 1 63 1 0 1 0 1 0 1 0 0 0
```

File Definition Section

This section defines the configuration files used for defining configuration properties implemented within configuration files. These files consist of zero or one template file definitions followed by zero, one, or two value file definitions. If a template file is defined, one or two value files must be defined; however, the contents of value files may be empty. This section was added for version 4.0 device interface files and is not present in version 3.1 and earlier files.

A file definition consists of the following lines:

Line	***Version***	***Contents***
Line 1	4.0	A line with the following syntax: **FILE** *name index type* [*length*] The fields are defined as follows: *name* — The filename. May be up to 16 characters without spaces. *index* — The file index as defined in the LONWORKS file transfer protocol. Set to 0 for the template file, or 1 or 2 for the value files. *type* — The file type as defined in the LONWORKS file transfer protocol. Set to 2 for the template file, or 1 for the value file. *length* — The number of bytes in the file. This value is optional and is calculated from the contents of the file, but must be specified if the contents of the file are not specified. When not required, the value may be omitted or set to 0. If both the length and file contents are specified, the length value must equal the number of bytes in the file contents.
Lines 2 – N	4.0	File contents. A line can be interpreted as characters or as binary data. Character format is indicated by a double quote (") as the first non-white space character. The quote is not included in the file. In this format a subsequent double quote is considered to terminate the string and it and all subsequent characters are not included. Non printable characters can be included by using a C-style hex escape sequence. The values A – F within a hex character escape sequence may be specified with upper or lower case letters. For example, to include a 0x8A character, enter `\x8a` or `\x8A` in the string. Binary format is assumed for any line not starting with a double quote (excluding white space). In binary format, numbers are entered using C-style hex values. Each value may optionally start with a “0x” or “\x” prefix. Values may optionally be separated with commas or spaces. If separators are not used, every pair of values represents one hex byte. Non-hex value characters are ignored. For example, the following generates a four-byte value of 0x0789abcd: `0x07, 0x89, 0xAB, 0xCD`

```
0x0789abcd
0789abcd
7,89,ab,cd
\x07\x89\xab\xcd
```

N+1	4.0	Blank line.

Network Variable Values Definition Section

This section defines default values for configuration properties implemented as configuration network variables. This section was added for version 4.0 device interface files and is not present in version 3.1 and earlier files.

The network variable values definition section consists of the following lines:

Line	***Version***	***Contents***
Line 1	4.0	The **NVVAL** keyword.

Lines 2 – N	4.0	A definition line for each configuration network variable defined in the device interface file. The order of the definitions must match the order of declaration of the configuration network variables in the device interface file, and there can be no more values than there are configuration network variables in the device interface file. Each line contains the default values, in hex. Each value may optionally start with a "0x" or "\x" prefix. Values may optionally be separated with commas or spaces. If separators are not used, every pair of values represents one hex byte. Non-hex value characters are ignored. For example, the following generates a two-byte value of 0x0789:
		`0x07, 0x89`
		`0x0789`
		`0789`
		`7,89`
		`\x07\x89`

Line N	4.0	Blank line.

Following is an example network variable values definition. Comments are used to identify each of the values.

```
NVVAL
# config network input long configNv1 = {5000};
0x13, 0x88
# config network input int configNv2 = {100};
0x64
# config network input long configNv3 = {2252};
0x08, 0xCC
```

Revision 4.402, May 2009, 078-0255-01B