MDF-Specification

Nico Schmoigl

MDF-Specification

by Nico Schmoigl

Published $Id: MDF-Specification.sgml,v 1.1 2001/04/20 17:28:48 nico Exp $

A binary orientated network-wide protocol for submitting information about bad memory modules.

Table of Contents

List of Tables

Chapter 1. Introduction

The MDF (*M*emory *D*ata *F*ormat) procotol is a block-orientated, abstract protocol, normally (but not necessarily) restricted to files and to memory areas. The block size depends on the amount of data specified with the given command. A sequence of MDF commands (see later) describes one (or more) memory modules (and its bad areas) uniquely.

1.1. A MDF command

A MDF command consists at least of the following bytes:

Table 1-1. mdf_cmd structure

offset	type	name	description
0x00	unsigned 8-bit	command	Describes the type of action which should be taken when reading this command; see reference below for more details which actions can be triggered.
0x01	unsigned 16-bit network order	sub-command	A (not-yet-used) sub command for specifing the exact action which should be taken
0x03	unsigned 8-bit	length	Stores how many bytes are in the data area which will follow this MDF command

Summarized, you can see that this structure has a total length of four bytes. Please note that after these bytes, there might be additional data which length is stated in the *length* field. Note that therefore the maximal amount of bytes in the data area is always restricted to 255 bytes; a *length* of zero means that no additional data is available.

1.2. A sequence of MDF commands

A sequence of MDF commands is one (or normally more) MDF commands concatinated in a specific, user-defined order. Please note that between the given commands there may be additional data as already stated during the definition of the command.

Therefore a valid sequence may look like this:

```
| cmd1 | subcmd1 | len1 | ***** data1 ***** | cmd2 | subcmd2 | len2 | cmd3 | sub-
cmd3 | len3 | ** data3 ** |
```

But please note that the length of the data1-field is always the same as len1 (the same on data3 and len3, too, for sure!)

Chapter 2. The MDF command reference

Every bit combination of the *command* field in a MDF command symbolizes a certain action to be performed. Here are their meanings:

Table 2-1. valid constants for the command field of an MDF command

content in the *command* field	**abbrevation**	**description**
0x01	MDFCMD_VERSION	Defines the current version of this protocol
0xff	MDFCMD_END	The MDF protocol ends here; all data after this mark should not be interpretated as MDF commands
0x10	MDFCMD_MODULENAME	Definition of module identification name
0x11	MDFCMD_SIZE	Definition of module size
0x20	MDFCMD_PATTERN	Submission of a pattern line for bad spots in a memory module

All other bytes not specified above are still unused; their meaning is not defined here. Therefore they should not be used in practice.

The commands above will now be discussed:

2.1. MDFCMD_VERSION - Version definition

command: MDFCMD_VERSION

sub-command: see description

length: always 0

data: nothing

2.1.1. Description

Defines the protocol version. The sub-command holds the version number. As this is the first release, sub-command should be always 1.

Further releases will always be compatible to earlier versions. Therefore, unknown commands should be skipped without taking any action.

*Important:*Any MDF data should begin with this command to make sure that the interpreter may use the correct command set

2.2. MDFCMD_END - End of protocol

command: MDFCMD_END

sub-command: does not matter

length: always 0

data: nothing

2.2.1. Description

This tells the interpreter that beyond this mark, no new command should be looked at.

This features enables the MDF specification to be included into other (network) protocols.

2.3. MDFCMD_MODULENAME - Start of a new module

command: MDFCMD_MODULENAME

sub-command: always 0

length: see description

data: see description

2.3.1. Description

MDFCMD_MODULENAME starts a new module definition. There is no "MODULE END" command; the next MDFCMD_MODULENAME starts a new module, closing the old one.

In the field *data*, you must specify an unique (globally, if possible) qualifier. The length of this qualifier is stored in the *length* field. The qualifier can consist of any byte combination. Please do not use the 0x0 byte as this may be confusing. It is recommended to use only tty-characters.

2.4. MDFCMD_MODULESIZE - Setting the size of a module

requires: a MDFCMD_MODULENAME must be prior this command

command: MDFCMD_MODULESIZE

sub-command: always 0

length: see description (recommendation: 4)

data: see description

2.4.1. Description

Defines the maximal size of the modules, defined by latest MDFCMD_MODULENAME command. If specified more than once for a module, the *last* MDFCMD_MODULESIZE is the valid one.

The actual module size is store in the *data* field. The field *length* stores its length. The data is in network order. The unit of the size is in (pure) bytes.

It is recommended to use an unsigned 32-bit value in network order for submission.

2.5. MDFCMD_PATTERN - Submission of a BadMEM pattern

command: MDFCMD_PATTERN

sub-command: depends on type

length: depends on type

data: see description

2.5.1. Description

There are several types of patterns:

Table 2-2. Pattern type definition

sub-command	type	comment
0x0000	ANCIENT	Please not use this pattern type - it is too old
0x0001	unused	
0x0002	V2	Due to the lack of upper bounds, please do not use the standard V2-type
0x0003	V2-with-ubound	This is the preferred pattern type

2.5.2. The V2-with-ubound pattern standard

A V2-with-ubound structure consists of four fields:

Table 2-3. structure of the V2-with-ubound pattern

offset	type	name	description
0x00	unsigned 32-bit network order	lbound	contains the lower bound of the pattern

offset	type	name	description
0x04	unsigned 32-bit network order	mask	contains the general mask of the pattern
0x08	unsigned 32-bit network order	ubound	contains the upper bound of the pattern
0x0b	unsigned 32-bit network order	offset	contains the (logical) offset of the pattern

If you need futher information what the single fields do, please read the BadMEM Pattern Specification, which you can download from http://badmem.sourceforge.net

Please note that the normal length of a V2-with-ubound structure is 16 bytes.

Chapter 3. A recommended MDF sequence

The following sequence of MDF commands is a recommended prototype:

- MDFCMD_VERSION
- first MDFCMD_MODULENAME
- first MDFCMD_MODULESIZE
- zero, one or more MDFCMD_PATTERN
- second MDFCMD_MODULENAME
- second MDFCMD_MODULESIZE
- zero, one or more MDFCMD_PATTERN
- repeat the last three steps for any module you want to define
- MDFCMD_END