VISTA HEALTH LEVEL SEVEN (HL7) MESSAGE PARSING UTILITIES

SUPPLEMENT TO PATCH DESCRIPTION

PATCH HL*1.6*118

JULY 2004

Department of Veterans Affairs (VA)

VHA OI Health Systems Design & Development (HSD&D)
Messaging & Interface Services (M&IS)

REVISION HISTORY

The following table displays the revision history for this document. Revisions to the documentation are based on patches and new versions released to the field.

Date	Revision	Description	Author
07/2004	V.1.0	Initial Draft. HL7 Message Parsing Utilities. Supplement to HL7 Site Manager & Developer Manual Version 1.6*56	REDACTED
01/2004		$$GET moved to HLOPRS PROCESSING MODE parsed from header	REDACTED
2/12/2004		Changed HL2PRS* routines to HLOPRS* - due to decision to move the namespace	REDACTED

Patch Revisions

For a complete list of patches related to this software, please refer to the Patch Module on FORUM.

TABLE OF CONTENTS

REVISION HISTORY ..... **I**

TABLE OF CONTENTS ..... **III**

1. OVERVIEW OF THE HL7 PARSING UTILITIES **1**

2. FUNCTION SPECIFICATIONS **2**

2.1. $$STARTMSG^HLPRS(.HLMSG,IEN,.HEADER) 2
2.1.1. Message Header, MSH Segment *2*
2.1.2. Message Header, BHS Segment *3*
2.2. $$NEXTSEG^HLPRS(.HLMSG,.SEGMENT) 4
2.3. $$GET^HLOPRS(.SEGMENT,SEQ,COMP,SUBCOMP,REP) 4
2.4. $$NEXTMSG^HLPRS(.HLMSG,.MSH) 4
2.4.1. Message Header, MSH Segment *5*

3. EXAMPLES OF MESSAGE PARSING **6**

3.1. PARSING A SINGLE MESSAGING 6
3.2. PARSING A BATCH MESSAGE 6

Table of Contents

1. OVERVIEW OF THE HL7 PARSING UTILITIES

This supplement is meant to assist application developers who are using the HL7 package. A new set of utilities for parsing HL7 messages was introduced with patch HL*1.6*118. Prior to this patch, each application was responsible for parsing received messages with little assistance from the HL7 package. The new parsing utilities relieve the individual applications from most of the burden of parsing messages.

The HL7 parsing utilities provide applications with the ability to easily parse their messages. There are two additional benefits:

1. For values within the message where instances of the HL7 encoding characters were replaced with an escape sequence, the parsed values are automatically restored to their original value.
2. MSH and BHS segments when parsed are represented as arrays with each individual value is assigned a meaningful subscript, for example:
    a. MSH("MESSAGE TYPE")="ADT"
    b. MSH("EVENT")="A01"

The utilities consist of these four functions:

1. $$STARTMSG: Starts the parsing process, returning the parsed header values.
2. $$NEXTSEG: Advances the parsing process to the next segment within the message.
3. $$GET: Obtains a particular value from the current segment.
4. $$NEXTMSG: Advances the parsing process to the next message within the batch, returning the parsed header values (for batch messages only).

2. FUNCTION SPECIFICATIONS

2.1. $$STARTMSG^HLPRS(.HLMSG,IEN,.HEADER)

This function begins the parsing of the message, parsing the header and returning the individual values in the array HEADER().

Input / Output	Description
Input	Internal Entry Number (IEN) of the message in the HL7 MESSAGE ADMINISTRATION file (#773)
Output	Function returns 1 on success, 0 on failure. Failure would indicate that the message was not found. HLMSG() (pass by reference, required): This array is used by the HL7 package to track the progress of parsing the message. The application MUST NOT touch it! HEADER (pass by reference, optional): This array contains the results of parsing the message header. The format is provided in section 2.1.1 below:

2.1.1. Message Header, MSH Segment

Sequence	Header	Description/Comment
	HEADER("SEGMENT TYPE")= "MSH"	
SEQ1	HEADER("FIELD SEPARATOR")	The 1st character field separator.
SEQ2	HEADER("COMPONENT SEPARATOR") HEADER("SUBCOMPONENT SEPARATOR") HEADER("REPETITION SEPARATOR") HEADER("ESCAPE CHARACTER")	The four encoding characters.
SEQ3	HEADER("SENDING APPLICATION")	
SEQ4	HEADER("SENDING FACILITY",1) HEADER("SENDING FACILITY",2) HEADER("SENDING FACILITY",3)	First Component Second Component Third Component
SEQ5	HEADER("RECEIVING APPLICATION")	
SEQ6	HEADER("RECEIVING FACILITY",1) HEADER("RECEIVING FACILITY",2) HEADER("RECEIVING FACILITY",3)	First Component Second Component Third Component
SEQ7	HEADER("DT/TM OF MESSAGE")	Converted to FileMan format.
SEQ8	HEADER("SECURITY")	
SEQ9	HEADER ("MESSAGE TYPE") HEADER("EVENT") HEADER("MESSAGE STRUCTURE")	First Component Second Component Third Component
SEQ10	HEADER("MESSAGE CONTROL ID")	Message control ID.
SEQ11	HEADER("PROCESSING ID") HEADER("PROCESSING MODE")	First Component Second Component
SEQ12	HEADER("VERSION")	
SEQ14	HEADER("CONTINUATION POINTER")	MESSAGE CONTROL ID of the message that this one continues.

SEQ15	HEADER("ACCEPT ACK TYPE")	ACCEPT ACKNOWLEDGMENT TYPE, <AL or NE>
SEQ16	HEADER("APP ACK TYPE")	APPLICATION ACKNOWLEDGMENT TYPE, <AL or NE>
SEQ17	HEADER("COUNTRY")	COUNTRY CODE

2.1.2. Message Header, BHS Segment

Sequence	Header	Description/Comment
	HEADER("SEGMENT TYPE")= "BHS"	
SEQ1	HEADER("FIELD SEPARATOR")	1st character field separator
SEQ2	HEADER("COMPONENT SEPARATOR") HEADER("SUBCOMPONENT SEPARATOR") HEADER("REPETITION SEPARATOR") HEADER("ESCAPE CHARACTER")	The four encoding characters,
SEQ3	HEADER("SENDING APPLICATION")	
SEQ4 (See Note)	HEADER("SENDING FACILITY",1) HEADER("SENDING FACILITY",2) HEADER("SENDING FACILITY",3)	First Component Second Component Third Component
SEQ5	HEADER("RECEIVING APPLICATION")	
SEQ6 (See Note)	HEADER("RECEIVING FACILITY",1) HEADER("RECEIVING FACILITY",2) HEADER("RECEIVING FACILITY",3)	First Component Second Component Third Component
SEQ7	HEADER("DT/TM OF MESSAGE")	Converted to FileMan Format.
SEQ8	HEADER("SECURITY")	
SEQ9	HEADER("BATCH NAME/ID/TYPE") The below fields are not defined by the standard within the BHS segment, but they are needed and are encoded in SEQ 9 by the VistA HL7 package: 1. HEADER("PROCESSING ID") 2. HEADER("ACCEPT ACK TYPE") 3. HEADER("APP ACK TYPE")	
SEQ10	HEADER("BATCH COMMENT") The VistA HL7 package, Version 1.6, currently requires that an application designate a batch message as being of a particular message type, event type, and version, and this information is encoded as components of SEQ10 1. HEADER("MESSAGE TYPE") 2. HEADER("EVENT") 3. HEADER("VERSION")	
SEQ11	HEADER("BATCH CONTROL ID")	
SEQ12	HEADER("REFERENCE BATCH CONTROL ID")	
Note: *The HL7 Version 2.4 Standards Manual does not document this, but components 2 & 3 were added in an errata to be compatible with the MSH segment. It is documented in version 2.5 of the standard.*		

2.2. $$NEXTSEG^HLPRS(.HLMSG,.SEGMENT)

This function advances parsing to the next segment.

Input / Output	Description
Input	HLMSG() (pass by reference, required): This array is used by the HL7 package to track the current position in the message. The application MUST NOT touch it!
Output	Function returns 1 on success, 0 if there are no more segments in this message. For batch messages, a return value of 0 does not preclude the possibility that there are additional individual messages within the batch. HLMSG (pass by reference, required): This array is used by the HL7 package to track the current position in the message. The application MUST NOT touch it! SEGMENT (pass by reference, required): The segment is returned in this array. SEGMENT("SEGMENT TYPE") is returned with the 3 letter HL7 segment type that always begins a segment. The structure is not further described here because it should not be accessed directly by the application developer. Use $$GET to obtain individual segment values. As a shortcut, $$GET^HLOPRS(0) may be used to return the segment type.

2.3. $$GET^HLOPRS(.SEGMENT,SEQ,COMP,SUBCOMP,REP)

This function gets a value from a segment that was obtained by calling $$NEXTSEG. The SEQ, COMP, SUBCOMP, and REP parameters are optional. If not specified, they default to the value1. For example, $$GET^HLPRS(.SEGMENT,1) will return the value of the first field, first component, first subcomponent, in the first occurrence. Since many fields consist of a single simple value, this is a useful feature. $$GET^HLOPRS(0) may be used to return the segment type.

Input / Output	Description
Input	❑ SEGMENT (required, pass by reference): The segment was placed in this array by $$NEXTSEG. ❑ SEQ: The sequence number of the field, defaults to 1 ❑ COMP: The number of the component, defaults to 1 ❑ SUBCOMP: The number of the subcomponent, defaults to 1 ❑ REP: The occurrence number. For a non-repeating field, the occurrence number need not be provided, because it is always 1.
Output	Function returns the value on success, "" on failure or if the specified part is not valued.

2.4. $$NEXTMSG^HLPRS(.HLMSG,.MSH)

Advances to the next message within the batch, with the MSH segment returned.

Input / Output	Description
Input	HLMSG (pass by reference, required) This array is used by the HL7 package to track the current position in the message. The application MUST NOT touch it!
Output	Function returns 1 on success, 0 if there are no more messages MSH (pass by reference, required): Returns the message header in the format provided in Section 3.4.1 below.

2.4.1. Message Header, MSH Segment

Sequence	Header	Description/Comment
	HEADER("SEGMENT TYPE")	"MSH"
SEQ1	HEADER("FIELD SEPARATOR")	The 1st character field separator.
SEQ2	HEADER("COMPONENT SEPARATOR") HEADER("SUBCOMPONENT SEPARATOR") HEADER("REPETITION SEPARATOR") HEADER("ESCAPE CHARACTER")	The four encoding characters.
SEQ3	HEADER("SENDING APPLICATION")	
SEQ4	HEADER("SENDING FACILITY",1) HEADER("SENDING FACILITY",2) HEADER("SENDING FACILITY",3)	First Component Second Component Third Component
SEQ5	HEADER("RECEIVING APPLICATION")	
SEQ6	HEADER("RECEIVING FACILITY",1) HEADER("RECEIVING FACILITY",2) HEADER("RECEIVING FACILITY",3)	First Component Second Component Third Component
SEQ7	HEADER("DT/TM OF MESSAGE")	Converted to FileMan format.
SEQ8	HEADER("SECURITY")	
SEQ9	HEADER ("MESSAGE TYPE") HEADER("EVENT") HEADER("MESSAGE STRUCTURE")	First Component Second Component Third Component
SEQ10	HEADER("MESSAGE CONTROL ID")	Message control ID.
SEQ11	HEADER("PROCESSING ID")	
SEQ12	HEADER("VERSION")	
SEQ14	HEADER("CONTINUATION POINTER")	MESSAGE CONTROL ID of the message that this one continues.
SEQ15	HEADER("ACCEPT ACK TYPE")	ACCEPT ACKNOWLEDGMENT TYPE, <AL or NE>
SEQ16	HEADER("APP ACK TYPE")	APPLICATION ACKNOWLEDGMENT TYPE, <AL or NE>
SEQ17	HEADER("COUNTRY")	COUNTRY CODE

3. EXAMPLES OF MESSAGE PARSING

3.1. Parsing a Single Messaging

```
N HLMSG,HEADER,SEG
;** start the parsing **
I $$STARTMSG^HLPRS(.HLMSG,66077,.HEADER) W !,"STARTING THE MESSAGE!",! ZW
HEADER

;** the output **
STARTING THE MESSAGE!
HEADER("ACCEPT ACK TYPE")=AL
HEADER("APP ACK TYPE")=NE
HEADER("CONTINUATION POINTER")=
HEADER("COUNTRY")=US
HEADER("DT/TM OF MESSAGE")=3040217.133516
HEADER("ENCODING CHARACTERS")=~|\&
HEADER("EVENT")=A31
HEADER("FIELD SEPARATOR")=^
HEADER("MESSAGE CONTROL ID")=55366077
HEADER("MESSAGE TYPE")=ADT
HEADER("PROCESSING ID")=T
HEADER("RECEIVING APPLICATION")=RG CIRN
HEADER("RECEIVING FACILITY",1)=553
HEADER("RECEIVING FACILITY",2)=
HEADER("RECEIVING FACILITY",3)=
HEADER("SECURITY")=
HEADER("SEGMENT TYPE")=MSH
HEADER("SENDING APPLICATION")=RG CIRN
HEADER("SENDING FACILITY",1)=553
HEADER("SENDING FACILITY",2)=
HEADER("SENDING FACILTY",3)=
HEADER("VERSION")=2.3

;** loop through the segments **
F  Q:'$$NEXTSEG^HLPRS(.HLMSG,.SEG)  D
. .W !,"SEGMENT TYPE=",SEG("SEGMENT TY
PE")
. .I SEG("SEGMENT TYPE")="PID" W !,"PATIENT NAME IS ",$$GET^HLOPRS(.SEG,5,3)_"
"_$$GET^HLOPRS(.SEG,5)

;** the output **
SEGMENT TYPE=PID
PATIENT NAME IS TWO CHESNEY
SEGMENT TYPE=NTE
SEGMENT TYPE=EVN
```

3.2. Parsing a Batch Message

```
N HLMSG,MSH,HEADER,SEG
```

```
I $$STARTMSG^HLPRS(.HLMSG,662,.HEADER) W !,"STARTING THE MESSAGE! MESSAGE
HEADER IS ",HEADER("SEGMENT TYPE")

;** output **
STARTING THE MESSAGE! MESSAGE HEADER IS BHS

;loop through the messages
F  Q:'$$NEXTMSG^HLPRS(.HLMSG,.MSH)  D
 .W !,"MESSAGE TYPE=",MSH("MESSAGE TYPE"),!," EVENT=",MSH("EVENT")
.;loop through the segments
 .F  Q:'$$NEXTSEG^HLPRS(.HLMSG,.SEG)  D
 . . W !,"SEGMENT TYPE=",SEG("SEGMENT TYPE")
 . . I SEG("SEGMENT TYPE")="PID" W !,"PATIENT NAME IS ", $$GET^HLOPRS(.SEG,5,3)_"
"_$$GET^HLOPRS(.SEG,5)

;**output**
MESSAGE TYPE=ORU
 EVENT=Z07
SEGMENT TYPE=PID
PATIENT NAME IS NITSCHE
SEGMENT TYPE=PD1
SEGMENT TYPE=ZPD
SEGMENT TYPE=ZTA
SEGMENT TYPE=ZIE
SEGMENT TYPE=ZEL
SEGMENT TYPE=ZEL
SEGMENT TYPE=ZEL
SEGMENT TYPE=ZEN
SEGMENT TYPE=ZCD
SEGMENT TYPE=ZRD
SEGMENT TYPE=ZCT
SEGMENT TYPE=ZEM
SEGMENT TYPE=ZEM
SEGMENT TYPE=ZGD
SEGMENT TYPE=ZGD
SEGMENT TYPE=ZIC
SEGMENT TYPE=ZIR
SEGMENT TYPE=ZDP
SEGMENT TYPE=ZIC
SEGMENT TYPE=ZIR
SEGMENT TYPE=ZIO
SEGMENT TYPE=NTE
SEGMENT TYPE=IN1
SEGMENT TYPE=ZMT
SEGMENT TYPE=ZMT
SEGMENT TYPE=ZMT
SEGMENT TYPE=ZBT
SEGMENT TYPE=ZSP
SEGMENT TYPE=RF1
```