# VISTA HEALTH LEVEL SEVEN (HL7)
# MSH SEGMENT CONTROL
# (DYNAMIC ROUTING)

## SENDING APPLICATION, SENDING FACILITY
## RECEIVING APPLICATION, RECEIVING FACILITY

## SUPPLEMENT TO PATCH DESCRIPTION

## PATCH HL*1.6*93

May 2003

Department of Veterans Affairs (VA)
VHA OI Health Systems Design & Development (HSD&D)
Messaging & Interface Services (M&IS)

# Revision History

## Documentation Revisions

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 |
|---|---|---|---|
| May 2003 | HL*1.6*93 | VistA Health Level Seven (HL7) MSH Segment Control (Dynamic Routing) documentation. | Larry Andreassen, Oakland OIFO, Susan Strack, Oakland OIFO. |

**Table 1: Documentation revision history**

## Patch Revisions

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

# Contents

# Tables and Figures

# Orientation

This is the documentation for the Veterans Health Information Systems and Technology Architecture (**V***IST***A**) Health Level Seven (HL7) MSH Segment Control (Dynamic Routing) software, exported in Patch HL*1.6*93. This manual uses several methods to highlight different aspects of the material:

- Various symbols are used throughout the documentation to alert the reader to special information. The following table gives a description of each of these symbols:

| Symbol | Description |
|---|---|
|  | Used to inform the reader of general information including references to additional reading material. |
|  | Used to caution the reader to take special notice of critical information. |

**Table 2: Documentation symbol descriptions**

- Descriptive text is presented in a proportional font (as represented by this font). "Snapshots" of computer online displays (i.e., character-based screen captures/dialogs) and computer source code are shown in a *non*-proportional font.

## Conventions for Technical References

The DIRECT^HLMA and GENERATE^HLMA APIs have been enhanced by patch HL*1.6*93. The parameters used in calling these APIs are identical. Therefore, in most cases, only the GENERATE^HLMA API is mentioned with the understanding that both DIRECT^HLMA and GENERATE^HLMA are represented by the one reference.

HLP("SUBSCRIBER"[,n]) is sometimes used in this documentation as a generic reference to either individual subscriber entries, i.e., HLP("SUBSCRIBER",n), or default subscriber entries, i.e., HLP("SUBSCRIBER").

## How to Obtain Technical Information Online

The MSH Segment Control software is fully explained and illustrated in hyper-text-based (HTML) documentation, which can be downloaded from the following address:

http://www.va.gov/vdl/Infrastructure.asp?appID=8

## Assumptions About the Reader

This manual is written with the assumption that the reader is familiar with the following:

- **V*IST*A** computing environment (e.g., Kernel Installation and Distribution System [KIDS]).
- VA FileMan data structures and terminology.
- M programming language.

## Reference Materials

Readers who wish to learn more about the **V*IST*A** Health Level Seven (HL7) software should consult the following:

- **V*IST*A** Health Level Seven documentation is made available online in Adobe Acrobat Portable Document Format (PDF) at the following web address:

  http://www.va.gov/vdl/Infrastructure.asp?appID=8 .
- Patch HL*1.6*93 Installation Instructions can be found in the description for HL*1.6*93 located in the National Patch Module (i.e., Patch User Menu [A1AE USER]) on FORUM.

Readers who wish to learn more about the Health Level Seven (HL7), Inc., Standards should consult the following:

- **V*IST*A** Health Level Seven (HL7) website:

  http://vista.med.va.gov/messaging/hl7/index.asp .

**DISCLAIMER: The appearance of external hyperlink references in this manual does not constitute endorsement by the Department of Veterans Affairs (VA) of this Web site or the information, products, or services contained therein. The VA does not exercise any editorial control over the information you may find at these locations. Such links are provided and are consistent with the stated purpose of this VA Intranet Service.**

# Chapter 1. Introduction

This documentation is intended for use in conjunction with the **V*IST*A** Health Level Seven (HL7) MSH Segment Control (Dynamic Routing) software, exported in Patch HL*1.6*93. It outlines the details of the work involved in this patch for **V*IST*A** developers and VA facility Information Resources Management (IRM) personnel.

## Dynamic Control of Routing-related Fields in the MSH Segment

This documentation describes enhancements being added to the **V*IST*A** HL7 software by which routing-related fields in the MSH segment can be more easily controlled than before. With the installation of Patch HL*1.6*93, routing-related fields can be controlled at the moment a MSH segment is created. The four routing-related fields in the **V*IST*A** HL7 MSH segment are shown as follows:

- SENDING APPLICATION
- SENDING FACILITY
- RECEIVING APPLICATION
- RECEIVING FACILITY

The four fields mentioned above are referred to throughout this document as the "routing-related" fields. The SENDING APPLICATION and SENDING FACILITY fields are referred to as "sending-related" fields. The RECEIVING APPLICATION and RECEIVING FACILITY are the "receiving-related" fields.

Throughout this documentation references to the routing-related fields are abbreviated as follows:

- SENDING APPLICATION—SAPP
- SENDING FACILITY—SFAC
- RECEIVING APPLICATION—RAPP
- RECEIVING FACILITY—RFAC

The first six pieces of the MSH segment are shown in Figure 1. The routing-related fields are highlighted in boldface.

```
MSH^~|\&^SAPP^SFAC^RAPP^RFAC ^...
```

**Figure 1: Routing-related fields in MSH segment**

The MSH segment is constructed by the **V*IST*A** HL7 package by concatenating existing local variables, separated by carets (^).

In the **V*IST*A** environment and in this documentation, the carets (^) delimiter is used in message segments. However, other delimiters can be used.

After the installation of Patch HL*1.6*93, immediately before the construction of the MSH segment by the **V*IST*A** HL7 software, applications calling **V*IST*A** HL7 to create messages have a method to examine and set the local variables used in the MSH segment's routing-related fields.

For a quick overview of the functionality of Patch HL*1.6*93, refer to the section titled "Appendix A: Quick Reference Guide to MSH Segment Control."

# Chapter 2. Dynamic Routing of HL7 Messages

The enhancement described in this document is often referred to as the "dynamic routing" enhancement. However, the software released in Patch HL*1.6*93 is not involved with the routing of a message. Rather, it allows applications to fully control the contents of the four routing-related fields described in the "Introduction" section of this document. No other actions or abilities are contained in this enhancement patch.

After the creation of the MSH segment, the message is routed by processes intrinsic to **V*IST*A** HL7, which are beyond the scope of Patch HL*1.6*93. Since the actual routing of the message frequently makes use of these four routing-related fields, it is natural to refer to this software as a "dynamic routing" enhancement.

## APIs Create Transmission-Ready HL7 Messages

The following two application program interfaces (APIs) in the **V*IST*A** HL7 software have been upgraded by patch HL*1.6*93 to include the MSH segment control functionality described in this documentation:

- DIRECT^HLMA
- GENERATE^HLMA

DIRECT^HLMA and GENERATE^HLMA are very similar. Both APIs have the same parameters and both create an HL7 message that is transmission-ready. The difference between these APIs is that DIRECT^HLMA creates and transmits messages in a *foreground* process. GENERATE^HLMA creates a message, which is transmitted by a different *background* process.

The DIRECT^HLMA and GENERATE^HLMA APIs have been enhanced by patch HL*1.6*93. The parameters used in calling these APIs are identical, including the new parameter values released with this patch, which are now passed to control routing-related fields.

In most cases, only the GENERATE^HLMA API is mentioned with the understanding that both DIRECT^HLMA and GENERATE^HLMA are represented by the one reference.

This manual does not document the HL7 DIRECT^HLMA and GENERATE^HLMA APIs. For information on this APIs, please refer to the **V*IST*A** HL7 Site Manager & Developer Manual. This documentation can be found at the following website:

http://www.va.gov/vdl/Infrastructure.asp - App8 .

# Generic Event Driver and Subscriber Protocols

The routing-related fields in the MSH segment are obtained from the event driver protocol and by the subscriber protocol in use at the time that the **V*IST*A** HL7 software creates the MSH segment. When the MSH segment is created, the sending-related fields are obtained from the event driver protocol, and the receiving-related fields are obtained from the subscriber protocol. Before installation of this patch, this required that a new event driver protocol be created for every unique set of sending-related fields. It also required that a new subscriber protocol be created for every unique set of receiving-related fields.

Patch HL*1.6*93 allows more control over the creation of these fields. After installation of this enhancement patch, calls to DIRECT^HLMA and GENERATE^HLMA APIs can directly control the values assigned to the routing-related fields in the MSH segment, overriding the values obtained from the event and subscriber protocols. The means by which this control is given to API calls is discussed in this documentation.

Because these API calls directly control the routing-related fields, it is now practical in many circumstances to create a generic event driver protocol and subscriber protocol to populate the routing-related fields at the time of message creation.

# Chapter 3. Enhanced HL7 Message Processing

Patch HL*1.6*93 is built on two already existing **V*IST*A** HL7 APIs: DIRECT^HLMA and GENERATE^HLMA. This chapter reviews how these APIs are called and explains how this patch adds MSH segment control functionality to them. Patch HL*1.6*93 upgrades these APIs to dynamically create and send HL7 Messages.

## VISTA HL7 APIs: DIRECT^HLMA and GENERATE^HLMA

The DIRECT^HLMA and GENERATE^HLMA API calls are used to create and send HL7 messages over Transmission Control Protocol (TCP) links. These APIs are constructed identically; the number of parameters and the characteristics of the parameters used in calls are the same.

### API Syntax

The call format for the GENERATE^HLMA API is shown in Figure 2.

```
D GENERATE^HLMA(HLEID,HLARYTYP,HLFORMAT,.HLRESLT,HLMTIEN,.HLP)
```

**Figure 2: Syntax for call to the GENERATE^HLMA API**

The HLEID parameter can be either the internal entry number (IEN) or the name of the event driver protocol.

The HLP array is the last parameter shown in Figure 2. The preceding period (.) before "HLP" indicates that it is passed by reference (e.g., .HLP). This array can contain various values such as HLP("SUBSCRIBER") and HLP("NAMESPACE").

### Passing MSH Segment Control Data Into API

An additional type of information can now be passed into this API in the HLP array by which the routing-related fields can be reviewed and set. This information must be passed using one of the following array structures:

- **Structure #1: "Default" Subscriber Entry**
  HLP("SUBSCRIBER")=^[SAPP]^[SFAC]^[RAPP]^[RFAC]^[XEC SUBRTN^XEC RTN]^[DEBUG]

- **Structure #2: "Individual" Subscriber Entry**
  HLP("SUBSCRIBER",n)=[SUB PROT]^[SAPP]^[SFAC]^[RAPP]^[RFAC]^[XEC SUBRTN^XEC RTN]^[DEBUG]

Structure #1 is the "default" subscriber entry. Structure #2 is the "individual" subscriber entry. Only one default subscriber entry can be passed into the APIs. Any number of individual subscriber protocol entries can be created and passed into DIRECT^HLMA or GENERATE^HLMA. The individual subscriber entries contain the name or internal entry number (IEN) of specific subscriber protocols on the first piece of data.

### Example: Data passed into GENERATE^HLMA

Figure 3 illustrates how the HLP array values can be created and passed into the GENERATE^HLMA API.

```
>S HLP("SUBSCRIBER")="^^^RALINK^512"
>S HLP("SUBSCRIBER",1)="RA TALKLINK ORM^^^TALKLINK^512"
>S HLP("SUBSCRIBER",2)="RA TALKLINK ORU^^^VOICELINK^549"
>D GENERATE^HLMA(HLEID,HLARYTYP,HLFORMAT,.HLRESLT,HLMTIEN,.HLP)
```

**Figure 3: HLP("SUBSCRIBER") array values created and passed into GENERATE^HLMA**

The receiving-related fields (i.e., receiving application and receiving facility) of the MSH segment are based on a specific subscriber protocol. At the time the MSH segment is created, a search is made for a matching "individual subscriber entry." The subscriber protocol of the individual subscriber entry must be the same as the subscriber protocol for the MSH segment being created.) If a matching individual subscriber entry (in the HLP("SUBSCRIBER",n) array) is found, it is evaluated, possibly resulting in the resetting of the routing-related fields in the MSH segment. If no match entry is found, the default subscriber entry (i.e., HLP("SUBSCRIBER") is used.

## API Processes Subscriber Protocol

Immediately before the MSH segment is created, **V*IST*A** HL7 finds the subscriber protocol being used in message creation. HL7 does this by searching the individual subscriber entries for a matching subscriber protocol. The individual subscriber entry will have the format of HLP("SUBSCRIBER",n). Based on the results of this search, the following action is taken:

- If the search through the HLP("SUBSCRIBER",n) entries turns up no match, the default subscriber entry is used if it exists.
- If a match is found, either the HLP("SUBSCRIBER",n) or the HLP("SUBSCRIBER") array entry will be used to reset the local variables used when the MSH segment is created. (This is how the routing-related fields are reset.)

The default subscriber entry never overrides individual subscriber entries. If an individual subscriber entry is found, and the general subscriber protocol entry HLP("SUBSCRIBER") also exists, the individual subscriber entry is used, and the general subscriber protocol is ignored.

## HLP("SUBSCRIBER"[,n]) Details

As mentioned previously, there are two formats for HLP("SUBSCRIBER"[,n]) data:

- **Structure #1: "Default" Subscriber Entry**
  HLP("SUBSCRIBER")=^[SAPP]^[SFAC]^[RAPP]^[RFAC]^[XEC SUBRTN^XEC RTN]^[DEBUG]

- **Structure #2: "Individual" Subscriber Entry**
  HLP("SUBSCRIBER",n) = SUB PROT^[SAPP]^[SFAC]^[RAPP]^[RFAC]^[XEC SUBRTN^XEC RTN]^[DEBUG]

The differences in the data specified are as follows:

- Data characteristics of the "default" subscriber entry are:
  - The array reference holds only one subscript: "SUBSCRIBER".
  - The first piece of data is always null.
- Data characteristics of the "individual" subscriber entry are:
  - The array reference holds two subscripts: "SUBSCRIBER", followed by a positive canonical numeral.
  - The first piece of data is never null, and holds either the IEN or name of a subscriber protocol.

HLP("SUBSCRIBER"[,n]) is sometimes used in this documentation as a generic reference to either individual subscriber entries, i.e., HLP("SUBSCRIBER",n), or default subscriber entries, i.e., HLP("SUBSCRIBER").

# Array Data—Definition and Description

Table 3 provides an explanation of the HLP("SUBSCRIBER"[,n]) array data, followed by comments about the different array structures.

| Piece # | Piece Data | Description |
|---|---|---|
| 1 | SUBSCRIBER PROTOCOL | Specifies null for general subscriber entries in the HLP("SUBSCRIBER") entry. In HLP("SUBSCRIBER",n) entries, this piece of data specifies the name or Internal Entry Number (IEN) for the subscriber protocol. |
| 2 | SENDING APPLICATION | Specifies the value to be used for SENDING APPLICATION in the MSH segment. |
| 3 | SENDING FACILITY | Specifies the value to be used for SENDING FACILITY in the MSH segment. |
| 4 | RECEIVING APPLICATION | Specifies the value to be used for RECEIVING APPLICATION in the MSH segment. |
| 5 | RECEIVING FACILITY | Specifies the value to be used for RECEIVING FACILITY in the MSH segment. |
| 6 | SUBROUTINE | (See XEC RTN below) |
| 7 | ROUTINE | Specifies M code that can be used to check and set the routing-related fields used in MSH segment creation. To do so, the routine that contains the M code to be executed must be placed in the XEC RTN piece of data, and the subroutine within the routine must be specified in the XEC SUBRTN piece of data. |
| 8 | DEBUG | Specifies whether global data should be trapped for later evaluation. |

**Table 3: HLP("SUBSCRIBER"[,n]) Array Data—Definition and Description**

Square brackets that enclose a field value in the default or subscriber entries indicate that the passing of that variable is optional. Additional explanation of these pieces of data and the rules for their use is given in the section titled "Enhancement Rules."

For more information on debugging, see the section titled "Debugging."

# Enhancement Rules

Table 4 lists the rules that the HLP("SUBSCRIBER") or HLP("SUBSCRIBER",n) data described in Table 3 must follow:

| Rule # | Name | Description |
|---|---|---|
| 1 | VALID EXECUTION REFERENCE | The "execution reference" is composed of the XEC SUBRTN and the XEC RTN. The passing of an execution reference is optional. But, if passed, both the XEC SUBRTN and the XEC RTN are required. |
| 2 | VALID ROUTING-RELATED FIELDS | There are a total of four routing-related fields. It is optional whether any of these fields are included in the passed-in data. If included, any combination of these fields may be present or null. |
| 3 | EXECUTION REFERENCE or ROUTING-RELATED FIELDS | It is required that an execution reference be passed, or one or more of the routing-related fields be included in the data. Rule #1 above states that it is not mandatory that an execution reference be passed. But, if the execution reference is not passed, one or more of the routing-related fields must be included in the passed data. Similarly, rule #2 states that it is not mandatory to pass routing-related information. But, if this is the case, an execution reference is required. |
| 4 | EXECUTION REFERENCE and ROUTING-RELATED FIELDS | It is legal to pass both routing-related field data and an execution reference. When this is done, the M code invoked by the execution reference can evaluate existing local variables and take the following actions:<br>• Take no action, (which will result in the passed-in routing-related field data being used in the MSH segment.)<br>• Change the variables used to create the routing-related fields in the MSH segment.<br>  M code actions always take precedence over routing-related field values. |

**Table 4: Rules for HLP("SUBSCRIBER") or HLP("SUBSCRIBER",n) data**

# API Examples

Example calls to the GENERATE^HLMA API are shown on the following pages, accompanied by explanations of each.

## Example 1—Individual Subscriber

In this example, when GENERATE^HLMA API is called, the following HLP("SUBSCRIBER"[,n]) local variable array is passed into this API.

```
>S HLP("SUBSCRIBER",1)="RA TALKLINK ORM^^^TALKLINK^512"
>S HLP("SUBSCRIBER",2)="RA TALKLINK ORU^^^VOICELINK^549"
>D GENERATE^HLMA(HLEID,HLARYTYP,HLFORMAT,.HLRESLT,HLMTIEN,.HLP)
```

**Figure 4: HLP("SUBSCRIBER") passes individual subscriber entries into GENERATE^HLMA**

Lets say that four subscriber protocols (see Figure 4) have been created and linked to the event driver protocol specified by the HLEID initial parameter. This will result in four messages being created, each message with its own unique subscriber protocol-based MSH segment. The four (made-up) subscriber protocols are:

- RA TALKLINK ORM—This subscriber protocol matches the subscriber protocol in HLP("SUBSCRIBER",1).)
- RA TALKLINK ORU—This subscriber protocol matches the subscriber protocol in HLP("SUBSCRIBER",2).)
- RA VIDEOLINK ORM
- RA VIDEOLINK ORU

Based on this example, the RECEIVING APPLICATION and RECEIVING FACILITY fields of data in the MSH segment will be reset as follows:

- The RA TALKLINK ORM subscriber protocol matches to the HLP("SUBSCRIBER",1) data's subscriber protocol. Since, HLP("SUBSCRIBER",1) data has a receiving application value of TALKLINK (on the fourth piece), TALKLINK will be placed in the RECEIVING APPLICATION of the MSH segment.
- Similarly, the value of "512" will be used for the RECEIVING FACILITY field of the MSH segment for the RA TALKLINK ORM subscriber protocol.
- The RA TALKLINK ORU subscriber protocol matches to the HLP("SUBSCRIBER",2) data's subscriber protocol. Since, HLP("SUBSCRIBER",2) data has a receiving application value of VOICELINK (on the fourth piece), VOICELINK will be placed in the RECEIVING APPLICATION of the MSH segment.
- Similarly, the value of "549" will be used for the RECEIVING FACILITY field of the MSH segment for the RA TALKLINK ORU subscriber protocol.

## Example 2—Individual and Default Subscriber

In "Example 1—Individual Subscriber" four subscriber protocols were associated with the subscriber protocol passed into GENERATE^HLMA. This same protocol setup that was used in Figure 4, should also be assumed for the example shown in Figure 5.

```
>S HLP("SUBSCRIBER")="^^^RALINK^512"
>S HLP("SUBSCRIBER",1)="RA TALKLINK ORM^^^TALKLINK^512"
>S HLP("SUBSCRIBER",2)="RA TALKLINK ORU^^^VOICELINK^549"
>D GENERATE^HLMA(HLEID,HLARYTYP,HLFORMAT,.HLRESLT,HLMTIEN,.HLP)
```

**Figure 5: HLP("SUBSCRIBER") passes individual and default subscriber entries into GENERATE^HLMA**

The example in Figure 5 is identical to that in Figure 4, except for the new default subscriber entry HLP("SUBSCRIBER").

As before, the RECEIVING APPLICATION and RECEIVING FACILITY fields of data in the MSH segment will be reset for RA TALKLINK ORM and RA TALKLINK ORU subscriber protocol messages. However, for the RA VIDEOLINK ORM and RA VIDEOLINK ORU subscriber protocol messages, the values of the RECEIVING APPLICATION and RECEIVING FACILITY fields in the default HLP("SUBSCRIBER") entry will be used. In this instance, the search for HLP("SUBSCRIBER",n) entries specific for the RA VIDEOLINK ORM and the RA VIDEOLINK ORU subscriber protocols fail, and so the default subscriber entry is used.

If the above events occurred, the results would be that the RECEIVING APPLICATION in the MSH segment would be set to RALINK, and the RECEIVING FACILITY would be set to 512; values taken from the HLP("SUBSCRIBER") entry.

## Example 3—Default Subscriber

The example in Figure 6 is similar to the example in Figure 5, with the exception that the individual subscriber entries have been removed.

```
>S HLP("SUBSCRIBER")="^^^RALINK^512"
>D GENERATE^HLMA(HLEID,HLARYTYP,HLFORMAT,.HLRESLT,HLMTIEN,.HLP)
```

**Figure 6: HLP("SUBSCRIBER") passes default subscriber entries into GENERATE^HLMA**

Figure 6 shows each MSH segment will be built using the values of RALINK and 512 from the default subscriber protocol entry.

## Example 4—M Code Only

Examples in Figure 4 through Figure 6 used routing-related fields in either the default subscriber entry or the individual subscriber entries. These routing-related fields contained string literal values (like "RALINK") to control the contents of the MSH segment fields. The following example uses only M code to control the creation of the routing-related fields in the message.

The example in Figure 7 includes only a default subscriber entry. And, the entry includes an execution reference to MSHREC^RAMSH.

```
>S HLP("SUBSCRIBER")="^^^^^MSHREC^RAMSH"
>D GENERATE^HLMA(HLEID,HLARYTYP,HLFORMAT,.HLRESLT,HLMTIEN,.HLP)
```

**Figure 7: HLP("SUBSCRIBER") passes an execution reference to MSHREC^RAMSH**

When the call to GENERATE^HLMA executes and immediately before the MSH segment for the message is created, the M code contained in MSHREC^RAMSH will be executed. The M code can then evaluate the environment and set the legally changeable local variables used in the creation of the MSH segment.

For more information on using M code to control the creation of the routing-related fields in an HL7 message, see the section titled "Variables Assist M Code."

# Chapter 4. Variables Assist M Code

Immediately before M code is executed (by calling MSHREC^RAMSH, in our previous example), local variables are defined to assist M code in evaluation and taking action. Table 5 lists these variables.

| Variable Name | Description |
|---|---|
| HLMSHPRE | Event driver protocol IEN^NAME |
| HLMSHPRS | Subscriber protocol IEN^NAME |
| HLMSH772 | HL7 MESSAGE TEXT file (#772) internal entry number |
| HLMSH773 | HL7 MESSAGE ADMINISTRATION file (#773) internal entry number (if available) |
| HLMSHSAO | Original SENDING APPLICATION |
| **HLMSHSAN** | **New SENDING APPLICATION to be used in the MSH segment** |
| HLMSHSFO | Original SENDING FACILITY |
| **HLMSHSFN** | **New SENDING FACILITY to be used in the MSH segment** |
| HLMSHRAO | Original RECEIVING APPLICATION |
| **HLMSHRAN** | **New RECEIVING APPLICATION to be used in the MSH segment** |
| HLMSHRFO | Original RECEIVING FACILITY |
| **HLMSHRFN** | **New RECEIVING FACILITY to be used in the MSH segment** |
| HLMSHTAG | M code subroutine |
| HLMSHRTN | M code routine |

**Table 5: Local variables defined to assist M code**

The variables that can be legally changed by M code are listed in Table 5, highlighted in boldface.

M code can evaluate the variables listed Table 5, but can only change the HLMSHSAN, HLMSHSFN, HLMSHRAN, and HLMSHRFN variables. This is the correct method for setting the values for the fields SENDING APPLICATION, SENDING FACILITY, RECEIVING APPLICATION, and RECEIVING FACILITY fields. After the M code finishes, these variables will be used when creating the MSH segment.

When M code resets the four local variables listed in Table 5, it takes precedence over string literal values for the routing-related fields that might be in the HLP("SUBSCRIBER"[,n]) array. (This is because M code is executed after all other actions that affect the MSH segment.)

Patch HL*1.6*93 introduces a new API, M^HLCSHDR4, which has been created to aid application developers using M code to reset routing-related variables. For more information, please see the section titled "M^HLCSHDR4 API—M Code to Set Routing-Related Fields."

# Chapter 5. Generic Subscriber Protocols

Prior to patch HL*1.6*93 it was possible to control the routing-related fields in the MSH segment. However, the process required to control these fields could be overwhelming.

The sending-related fields SENDING APPLICATION and SENDING FACILITY are obtained from the event driver protocol in use at the time of message creation. Prior to patch HL*1.6*93, control is possible over these sending-related fields by creating individual event driver protocols for each uniquely desired Sending Application and Sending Facility combination.

The receiving-related fields RECEIVING APPLICATION and RECEIVING FACILITY are obtained from the subscriber protocol(s) associated with an event driver protocol. (There may be one or more subscriber protocols associated with each event driver protocol.) Control over these fields is established by defining individual subscriber protocols for each uniquely desired receiving application and receiving facility combination.

To illustrate, consider the situation shown in Table 6, where three subscriber protocols are created in order to control the RECEIVING FACILITY field in the MSH segment:

| Event Protocol | | Subscriber Protocol | Subscriber Protocol's Receiving Facility |
|---|---|---|---|
| A02 SERVER | <--- | A02-72 | 512 |
| | <--- | A02-73 | 512A5 |
| | <--- | A02-74 | 512A4 |

**Table 6: Three subscriber protocols control RECEIVING FACILITY field in MSH segment**

Table 6 shows that when the A02 SERVER event "fires," that three distinct messages will be created and transmitted to remote locations (i.e., One call to GENERATE^HLMA, passing in A02 SERVER as the event driver protocol, will result in three messages being created and transmitted.).

The structure in Table 6 is created by entries in an event driver protocol's SUBSCRIBERS multiple, or by linking the three subscriber protocols to the A02 SERVER event protocol on-the-fly using dynamic addressing.

With the additional control over the RECEIVING FACILITY field in the MSH segment implemented by Patch HL*1.6*93, the structure shown in Table 7 can be created:

| Event Protocol | | Subscriber Protocol | Subscriber Protocol's Receiving Facility |
|---|---|---|---|
| A02 SERVER | <--- | A02 | 512 |

**Table 7: One subscriber protocol—dynamic control over RECEIVING FACILITY field in MSH seg.**

In other words, only one subscriber protocol is required. This is a "generic" subscriber protocol. When using this generic subscriber protocol, the RECEIVING FACILITY field value of 512 will automatically be placed in the MSH segment, or 512 can be overridden by HLP("SUBSCRIBER"[,n]) data.

To illustrate the use of MSH segment control, consider the following simple example:

```
>S HLP("SUBSCRIBER")="^^^^^MSHREC^RAMSH"
>D GENERATE^HLMA(HLEID,HLARYTYP,HLFORMAT,.HLRESLT,HLMTIEN,.HLP)
```

**Figure 8: Simple MSH segment control**

The HLP("SUBSCRIBER") array entry is a "default" subscriber entry and will be applied to all subscriber protocols linked to the HLEID event driver protocol. In this example, Figure 8, let's assume that only one subscriber protocol is associated with the event driver protocol. In that case, when the MSH segment is created, the MSHREC^RAMSH M code can create the RECEIVING FACILITY field value for use in the MSH segment.

In the first example shown in Table 6, three subscriber protocols were linked with an event driver protocol. When a call is made to GENERATE^HLMA, this results in three separate messages being created and transmitted. However, if only one generic subscriber protocol is linked to the event driver protocol, when GENERATE^HLMA is called, only one message will be created and transmitted. To send the three messages out to three unique sites, it would be necessary to call GENERATE^HLMA three times. Each call to GENERATE^HLMA would control the created MSH segment in such a way as to contain the desired receiving-related fields.

# Advantages/Disadvantages to MSH Segment Control

There are advantages and disadvantages to using the MSH segment control techniques provided by patch HL*1.6*93. Each are detailed under the two categories listed as follows:

### Protocol-Based Control of MSH Segment Fields

Advantages:

- Avoids multiple calls to GENERATE^HLMA.
- Stores the text of the message (all segments other than MSH) only once.

Disadvantage:

- Requires creation of one subscriber protocol for every receiving location.

### HLP("SUBSCRIBER"[,n])-Based Control of MSH Segment Fields

Advantage:

- Avoids creating multiple subscriber protocols.

Disadvantages:

- May require multiple calls to GENERATE^HLMA.
- Stores the text of the message one time for every call to GENERATE^HLMA.

The most advantageous situation for HLP("SUBSCRIBER"[,n])-based control of MSH segment fields will usually be when there are a large number of receiving locations, and the size of the message being transmitted is relatively small. This maximizes the advantage of using this method in that only one subscriber protocol will need to be created and maintained instead of many. And, a small sized message minimizes the disadvantage of having to store the text of the message once for every recipient of the message.

It is possible to realize the advantages of both methods of controlling the routing-related fields in the MSH segment by using them simultaneously. To do so, subscriber protocols would be created as before, and linked to an event protocol. However, HLP("SUBSCRIBER"[,n]) data could still be passed into DIRECT^HLMA or GENERATE^HLMA to evaluate and reset the routing-related fields where appropriate.

For a quick overview of the functionality of patch HL*1.6*93, refer to "Appendix A: Quick Reference Guide to MSH Segment Control."

# Chapter 6. Routing Field Change Ramifications

When *creating* protocols for HL7 applications for *messaging between* **V***IST***A** *HL7 applications or sites*, great care is required to ensure that protocol field values match the values of the protocols on receiving systems. For example, Table 8 shows event and subscriber protocol that might be created on a sending system:

| Protocol Name | Type | Sending Application | Receiving Application |
|---|---|---|---|
| ORU SERVER | Event | VOICERAD | |
| ORU CLIENT | Subscriber | | RADIOLOGY |

**Table 8: Event and subscriber protocol created on sending system**

When protocols are created on the receiving system, the sending application name (VOICERAD) and the receiving application name (RADIOLOGY) must be identical to the names on the sending system.

The same caution is required when *changing* routing-related fields in the MSH segment. It is true that the sending application name can now be easily changed. However, the value this field is changed to must be found as one of the sending applications in the receiving site's event protocols.

**When sending messages to non- V*IST*A HL7 applications or sites, the responsibility still remains with the application developer to ensure that the routing-related field changes are compatible and appropriate.**

## VISTA HL7 Messaging Considerations

When a message is sent to a **V***IST***A** HL7 system, an attempt is made on the receiving system to find an event and subscriber protocol to process the message. This topic explains the means by which these protocols are found.

When a new message is received, the following fields are extracted from the MSH segment and are used to find the protocols:

- SENDING APPLICATION
- MESSAGE TYPE
- EVENT TYPE
- VERSION
- RECEIVING APPLICATION

The first four fields listed are compared to the same fields, which are present in event protocols on the receiving system. The event protocol in which all four fields match the same fields in the MSH segment is the one used.

If the first four fields don't match the same fields in the MSH segment, a secondary search using less stringent conditions is made. In this secondary search, the EVENT TYPE field is not used; the search is for event protocols in which the SENDING APPLICATION, MESSAGE TYPE, and VERSION match.

After the event protocol is found, the subscriber protocol linked to the just-found event protocol is evaluated. The subscriber protocol with the same receiving application as the MSH segment's receiving application field is used for message processing.

The sending application field and the receiving application field can now be easily changed. However, messaging will abort, if these fields are changed to values that don't exist in the event and subscriber protocols on the receiving system.

## Protocol Messaging Fields Report Option

In order to assist developers in finding matching protocols on sending and receiving systems, Patch HL*1.6*93 introduces a new report listing the protocol information that is used when searching for matching protocols. This new report can be printed using the Protocol Messaging Fields Report [HL PROTOCOL MSG FIELDS REPORT] menu option. (This option is located in the Reports menu, under the Interface Developer Options menu.)

The dialog seen when using this option is shown in Figure 9.

```
                        HL7 Protocol Messaging Fields
================================================================================
This 'HL7 Protocol Messaging Fields' report holds information that will help you
determine the effects from changes to routing-related fields in the MSH segment when
messages are sent between or within VistA HL7 systems. Additional explanation is
included at the bottom of the report.

* Previous selection: SENDING APPLICATION not null
START WITH SENDING APPLICATION: FIRST//  <Enter>
  * Previous selection: TRANSACTION MESSAGE TYPE not null
  START WITH TRANSACTION MESSAGE TYPE: FIRST//  <Enter>
DEVICE:  <Enter>   SYSTEM    Right Margin: 80//  <Enter>


...HMMM, LET ME THINK ABOUT THAT A MOMENT...


Protocol Fields Used in Messaging             JAN 16,2003  15:27    PAGE 1
Snd/Rec App's    mTYP   eTYP   Ver        Protocol                      Link__
-------------------------------------------------------------------------------
A1BV ADT SERVER  ADT    A01    2.2    |   JC CIRN SERVER
-CIRN ROUTER     ADT    A01    2.2    |   JC CIRN CLIENT/ROUTER
-JC KRN REC      ADT           2.2    |   JC TEST RECV                 MM  MAIL
-------------------------------------------------------------------------------
A1BX CPT         MFN    M01    2.2    |   RH MFN SERVER
-CPT HL7         MFN    M01    2.2    |   RH MFN CLIENT
-------------------------------------------------------------------------------
AC-VOICERAD      ORU    R01    2.3    |   AC ORU SERVER
-AC-RADIOLOGY           R01    2.3    |   AC ORU CLIENT                NC  TCP
-------------------------------------------------------------------------------
ADT A01 KERNEL   ADT    A01    2.1    |   PSD A01 SERVER
-ADT A01 KERNEL  ACK    A01    2.1    |   PSD A01 CLIENT                   TCP
-------------------------------------------------------------------------------
ALSNXTHLLPTEST   ORU    R01    2.2    |   ALSNXT2ISC SERVER
-ALSISCHLLPTEST  ACK    R01    2.2    |   ALSNXT2ISC CLIENT            SH  HLLP
-------------------------------------------------------------------------------
^

Press RETURN for 'printout help', or '^' to exit...  <Enter>
```

**Figure 9: Dialog seen from using the Protocol Messaging Fields Report option**

The last line of the screen dialog in Figure 9 illustrates that you can press the Enter (or Return) key to output further explanation of this report, or you can enter an up-arrow to skip the printout explanation and exit the report. It is recommended that you read this "printout help," shown in Figure 10, as it provides additional information augmenting this documentation.

```
When messages are received, their SENDING APPLICATION (MSH-3), MESSAGE TYPE (MSH-9),
EVENT TYPE (MSH-9), and HL7 VERSION (MSH-12) fields are used to find the event driver
protocol to be used in processing the just-received message. After the event protocol
is found, that protocol's subscriber protocols are evaluated.  The subscriber
protocol with a RECEIVING APPLICATION value that matches the RECEIVING APPLICATION
field in the MSH segment (MSH-5) is used.

The first line for every "section" in the printout is the event driver protocol.
Lines preceded by dashes, are related subscriber protocols.  An example is shown
below.

Snd/Rec App's    mTYP   eTYP   Ver        Protocol                      Link
-------------------------------------------------------------------------------
AC-VOICERAD      ORU    R01    2.3    |   AC ORU SERVER
-AC-RADIOLOGY    ORU    R01    2.3    |   AC ORU CLIENT                 NC  TCP

In this example, the 'AC-VOICERAD' line holds information for the 'AC ORU SERVER'
event protocol.  And, the '-AC-RADIOLOGY' line holds information for the 'AC ORU
CLIENT' subscriber protocol.
```

**Figure 10: Protocol Messaging Fields Report option "printout help"**

When changes to the routing-related fields in the MSH segment on the sending system are contemplated, the results can be easily determined by referring to the printout from the receiving system. This is done in the following steps:

1. Determine the following values from the MSH segment being sent:
    - Sending application—Used to find the event protocol.
    - Message type—Used to find the event protocol.
    - Event type—Used to find the event protocol.
    - Version—Used to find the event protocol.
    - Receiving application—Used to find the subscriber protocol.
2. Refer to the printout from the receiving site to find an entry where the first (event protocol) row holds matching values for the first four fields. (Remember that if all four fields do not match, a secondary search is conducted for an event protocol in which the sending application, message type, and version match.)
3. After the event protocol is found, look at the line(s) (preceded by dashes) immediately below the event protocol row for a subscriber protocol with the same receiving application as in the MSH segment.

If the above steps fail (i.e., if no matching event and subscriber protocol rows are found), then even if the routing-related fields in the MSH segment are changed, when the message is received, the receiving site will send back a rejection acknowledgement message. The rejection acknowledgement message specifies that the sending application could not be found, or that the receiving application could not be found.

**When implementing the functionality provided by Patch HL*1.6*93, it is highly recommended that this report be printed on both the sending and receiving system and used whenever changes to the routing-related fields in the MSH segment are made.**

# Chapter 7. Debugging

Several tools and help methods have been created to assist application developers in incorporating the features released with Patch HL*1.6*93 into their software. They fall into two categories:

- Routing-related information stored in globals for application developer evaluation.
- An API created to assist developers using M code to reset routing-related fields.

These help methods are explained further in this chapter.

## Global Data

As of Patch HL*1.6*93, whenever routing-related fields are changed, global data is created that holds the values of the routing-related fields before they where changed. This data is stored in audit fields, in the HL7 MESSAGE ADMINISTRATION file (#773).

Optionally, application developers can create even more helpful global data for debugging new applications. Some debug global data is placed in the audit fields in the HL7 MESSAGE ADMINISTRATION file (#773). Some is placed in the ^XTMP global.

Both types of global data are documented on the following pages, under the topic sections titled "Audit Fields" and "Debug Global Data."

### Audit Fields

As described earlier in this documentation, HLP("SUBSCRIBER"[,n]) data is passed into the DIRECT^HLMA and GENERATE^HLMA APIs in order to control the setting of the routing-related fields.

For information on Passing MSH Segment Control Data into the DIRECT^HLMA and GENERATE^HLMA APIs, see the topic section titled "Passing MSH Segment Control Data Into API."

Whenever the MSH segment is reset in this way, information about the resetting of this data is stored in the new audit fields in the HL7 MESSAGE ADMINISTRATION file (#773). Figure 11 shows these audit fields created in global map format, highlighted in boldface.

```
^HLMA(D0,91)= (#91.01) ORIGINAL SND APP [1F] ^ (#91.02) ORIGINAL SND
            ==>APP-SOURCE [2S] ^ (#91.03) ORIGINAL SND FAC [3F] ^ (#91.04)
            ==>ORIGINAL SND FAC-SOURCE [4S] ^ (#91.05) ORIGINAL REC APP [5F] ^
            ==>(#91.06) ORIGINAL REC APP-SOURCE [6S] ^ (#91.07) ORIGINAL REC
            ==>FAC [7F] ^ (#91.08) ORIGINAL REC FAC-SOURCE [8S] ^
```

**Figure 11: Fields contain information about resetting data stored in the HL7 MESSAGE ADMINISTRATION file (#773)**

For example, using Patch HL*1.6*93's functionality, let's say that the following MSH segment:

```
MSH^~|\&^AC-VOICERAD^512^AC-RADIOLOGY^512^20030116135828-0800^^
ORU~R01^9987560^T^2.3^^^AL^AL^
```

is changed to:

```
MSH^~|\&^VOICERAD^512^AC-RADIOLOGY^512^20030116135828-0800^^
ORU~R01^9987560^T^2.3^^^AL^AL^
```

In other words, the sending application value of AC-VOICERAD was changed to VOICERAD. Let's say that this change was made by passing the following in a local variable array entry:

```
HLP("SUBSCRIBER")=^VOICERAD
```

Given these details, the first two fields in ^HLMA(D0,91) would be stored as:

```
^HLMA(D0,91)=AC-VOICERAD^A
```

The first field holds the value of the sending application before change, and the second field holds the way the change was made.

If the string literal value in pieces 2 through 5 of the local variable array entry was used to change the value, the value of the second field will be A. And, if the M reference in the local variable array entry on pieces 6 and 7 was used to change the value, the value of the second field will be M.

Table 9 lists all the fields in the ^HLMA(D0,91) node.

| Field # | Field Name | Description |
|---|---|---|
| 91.01 | ORIGINAL SND APP | Holds the pre-change value for sending application. |
| 91.02 | ORIGINAL SND APP-SOURCE | Holds A if the field was changed by the local variable array entry, or M if the field was changed by M code. |
| 91.03 | ORIGINAL SND FAC | Holds pre-change value for sending facility. |
| 91.04 | ORIGINAL SND FAC-SOURCE | Holds A or M. |
| 91.05 | ORIGINAL REC APP | Holds pre-change value for receiving application. |
| 91.06 | ORIGINAL REC APP-SOURCE | Holds A or M. |
| 91.07 | ORIGINAL REC FAC | Holds pre-change value for receiving facility. |
| 91.08 | ORIGINAL REC FAC-SOURCE | Holds A or M. |

**Table 9: Fields in the ^HLMA(D0,91) node**

The global data stored in the ^HLMA(D0,91) node, Table 9, is always created whenever routing-related fields in the MSH segment are changed.

## Debug Global Data

The eighth piece of HLP("SUBSCRIBER"[,n]) data, DEBUG, can be used to control the creation of useful debugging data. The syntax for the DEBUG data along with other details is described on the following pages.

The pieces of the HLP("SUBSCRIBER"[,n]) data were described previously in the chapter titled "Enhanced HL7 Message Processing."

In addition to the audit fields stored on the ^HLMA(D0,91) node, there are two types of global data that can optionally be recorded at the time of MSH segment creation for later evaluation:

- The HLP("SUBSCRIBER"[,n]) entry used to reset the routing-related fields can be permanently stored in the HL7 MESSAGE ADMINISTRATION file (#773).
- Select local variables, or all local variables existing at the moment of MSH segment creation can be stored in the ^XTMP global.

Patch HL*1.6*93 added new fields to the HL7 MESSAGE ADMINISTRATION file (#773) in order to store the HLP("SUBSCRIBER"[,n]) entry used when the routing-related fields are reset. The Global Map view of this data is shown in Figure 12.

```
^HLMA(D0,90)= (#90.01) HLP-SUBSCRIBER PROTOCOL [1F] ^ (#90.02) HLP-SENDING
           ==>APPLICATION [2F] ^ (#90.03) HLP-SENDING FACILITY [3F] ^
           ==>(#90.04) HLP-RECEIVING APPLICATION [4F] ^ (#90.05)
           ==>HLP-RECEIVING FACILITY [5F] ^ (#90.06) HLP-SUBROUTINE [6F] ^
           ==>(#90.07) HLP-ROUTINE [7F] ^ (#90.08) HLP-DEBUG [8F] ^
```

**Figure 12: Global Map of data in HL7 MESSAGE ADMINISTRATION file (#773)**

As mentioned earlier, the structure of the HLP("SUBSCRIBER"[,n]) data is:

```
[SUB PROT]^[SAPP]^[SFAC]^[RAPP]^[RFAC]^[XEC SUBRTN^XEC RTN]^[DEBUG]
```

**Figure 13: Structure of HLP("SUBSCRIBER"[,n]) data**

The eighth piece of HLP("SUBSCRIBER"[,n]) data (DEBUG) can be used to control whether the HLP("SUBSCRIBER"[,n]) entry is stored on ^HLMA(D0,90) and also whether additional global data is stored in ^XTMP. Figure 14 shows the syntax of the DEBUG parameter in HLP("SUBSCRIBER"[,n]):

```
#1-#2
```

**Figure 14: DEBUG parameter syntax in HLP("SUBSCRIBER"[,n])**

The #1 piece of the DEBUG data shown in Figure 14, controls whether the HLP("SUBSCRIBER"[,n]) data used in override is stored on the ^HLMA(D0,90) node. It can have the value of 1, 0, or null. (Null is considered a zero.)

- If null or set to zero, *no* ^HLMA(D0,90) data is stored.
- If set to 1, ^HLMA(D0,90) data is stored.

The #2 piece of data shown in Figure 14 controls whether debug global data is stored in ^XTMP. This data can have the value of 1 or 0 or null.

- If null or set to zero, *no* ^XTMP data is stored.
- If set to 1, ^XTMP data is stored.

The #2 piece of data also controls the amount of ^XTMP data that is stored. It can have the value of 1, 2, 0, or null. If null or zero, no ^XTMP data is stored.

- If set to 1, only the most critical local variables are stored.
- If set to 2, all existing local variables are stored.

Figure 15 illustrates the use of the DEBUG parameter. In this example, both types of data are requested. The structure of the data passed into GENERATE^HLMA was:

```
HLP("SUBSCRIBER")="^SAN^SFN^RAN^RFN^SET^ZZLJA^1-1"
```

**Figure 15: Data in the DEBUG parameter passed into GENERATE^HLMA**

The two types of data created from the DEBUG piece of data (i.e., "1-1") are highlighted in boldface in Figure 16.

```
^HLMA(6944,0) = 7075^9986944^O^D^^6944^190^322^^^32^29^1^46
^HLMA(6944,90) = ^SAN^SFN^RAN^RFN^SET^ZZLJA^1-1
^HLMA(6944,91) = VOICERAD^M^512^A^RADIOLOGY^A^512^A
^HLMA(6944,"MSH",0) = ^^1^1^3021024
^HLMA(6944,"MSH",1,0) = MSH^~|\&^MSAN^SFN^RAN^RFN^20021024115103-
0800^^ORU~R01^9986944^D^2.2^^^AL^AL^
^HLMA(6944,"P") = 1^^^^^^600


^XTMP("HLCSHDR3 6944",0) = 3021024.155104^3021024.115104^Debug data created by
DEBUG~HLCSHDR4
^XTMP("HLCSHDR3 6944",1,0) = 3021024.115104^7075^321^ZZLJA RH ORU SERVER^322^ZZLJA RH
ORU CLIENT
^XTMP("HLCSHDR3 6944",1,1) = ^SAN^SFN^RAN^RFN^SET^ZZLJA^1-1-1
^XTMP("HLCSHDR3 6944",1,"RA") = RADIOLOGY^RAN^RAN
^XTMP("HLCSHDR3 6944",1,"RF") = 512^RFN^RFN
^XTMP("HLCSHDR3 6944",1,"SA") = VOICERAD^SAN^MSAN
^XTMP("HLCSHDR3 6944",1,"SF") = 512^SFN^SFN
```

**Figure 16: Data created as output from the DEBUG parameter highlighted in boldface**

The subscript for ^XTMP data is always "HLCSHDR3 " (i.e., the HLCSHDR3 variable and one space) concatenated with the HL7 MESSAGE ADMINISTRATION file's IEN. In the example in Figure 16, the

IEN is 6944, and the full subscript is "HLCSHDR3 6944". The explanation for the ^XTMP data is documented on the following pages.

### ^XTMP ("HLCSHDR3 6944",1,0)

The following is the ^XTMP("HLCSHDR3 6944",1,0) data entry excerpt, which is part of the output from the DEBUG parameter shown in Figure 16:

```
^XTMP("HLCSHDR3 6944",1,0) = 3021024.115104^7075^321^ZZLJA RH ORU SERVER^322^ZZLJA RH
ORU CLIENT
```

Breakdown of data entry:

- Piece 1 is the date and time when ^XTMP data created.
- HLMSH772—Piece 2 is the local variable holding the HL7 MESSAGE TEXT file (#772) IEN.
- HLMSHPRE—Pieces 3 and 4 are the Event driver protocol IEN^NAME. (Piece 3 stores the IEN and piece 4 stores the NAME.)
- HLMSHPRS—Pieces 5 and 6 are the subscriber protocol IEN^NAME. (Piece 5 stores the IEN and piece 6 stores the NAME.)

### ^XTMP("HLCSHDR3 6944",1,1)

The following is the ^XTMP("HLCSHDR3 6944",1,1) data entry excerpt, which is part of the output from the DEBUG parameter shown in Figure 16:

```
^XTMP("HLCSHDR3 6944",1,1) = ^SAN^SFN^RAN^RFN^SET^ZZLJA^1-1-1
```

Breakdown of data entry:

- The HLP("SUBSCRIBER"[,n]) entry used.

### ^XTMP("HLCSHDR3 6944",1,"RA")

The following is the ^XTMP("HLCSHDR3 6944",1,"RA") data entry excerpt, which is part of the output from the DEBUG parameter shown in Figure 16:

```
^XTMP("HLCSHDR3 6944",1,"RA") = RADIOLOGY^RAN^RAN
```

Breakdown of data entry:

- Piece 1 is the original RECEIVING APPLICATION value.
- Piece 2 is the free-text value for RECEIVING APPLICATION.
- Piece 3 is the value for RECEIVING APPLICATION after M code runs.

### ^XTMP("HLCSHDR3 6944",1,"RF")

The following is the ^XTMP("HLCSHDR3 6944",1,"RF") data entry excerpt, which is part of the output from the DEBUG parameter shown in Figure 16:

```
^XTMP("HLCSHDR3 6944",1,"RF") = 512^RFN^RFN
```

Breakdown of data entry:

- Piece 1 is the original RECEIVING FACILITY value.
- Piece 2 is the free-text value for RECEIVING FACILITY.
- Piece 3 is the value for RECEIVING FACILITY after M code runs.

### ^XTMP("HLCSHDR3 6944",1,"SA")

The following is the ^XTMP("HLCSHDR3 6944",1,"SA") data entry excerpt, which is part of the output from the DEBUG parameter shown in Figure 16:

```
^XTMP("HLCSHDR3 6944",1,"SA") = VOICERAD^SAN^MSAN
```

Breakdown of data entry:

- Piece 1 is the original SENDING APPLICATION value.
- Piece 2 is the free-text value for SENDING APPLICATION.
- Piece 3 is the value for SENDING APPLICATION after M code runs.

### ^XTMP("HLCSHDR3 6944",1,"SF")

The following is the ^XTMP("HLCSHDR3 6944",1,"SF") data entry excerpt, which is part of the output from the DEBUG parameter shown in Figure 16:

```
^XTMP("HLCSHDR3 6944",1,"SF") = 512^SFN^SFN
```

Breakdown of data entry:

- Piece 1 is the original SENDING FACILITY value.
- Piece 2 is the free-text value for SENDING FACILITY.
- Piece 3 is the value for SENDING FACILITY after M code runs.

# M^HLCSHDR4 API—M Code to Set Routing-Related Fields

Immediately before the MSH segment is constructed, application developers can invoke M code to directly set the local variables used for the routing-related fields in the MSH segment. This was discussed in the section titled "Variables Assist M Code."

Table 10 lists the four variables that can be changed:

| Variable Name | Description |
|---|---|
| HLMSHSAN | New SENDING APPLICATION to be used in the MSH segment. |
| HLMSHSFN | New SENDING FACILITY to be used in the MSH segment. |
| HLMSHRAN | New RECEIVING APPLICATION to be used in the MSH segment. |
| HLMSHRFN | New RECEIVING FACILITY to be used in the MSH segment. |

**Table 10: Local variables that can be legally changed by M code**

If an application invokes M code to set the SENDING APPLICATION, the M code must set the HLMSHSAN variable to the new value.

The M^HLCSHDR4 API was release with Patch HL*1.6*93. This API was created for application developers wishing to use M code to set routing-related fields, and should be useful to illustrate the process occurring. In addition, it allows developers to interactively set the routing-related variables listed above.

**This API should be used by application developers when creating their software. But, a "real" application-specific API must be substituted before software release.**

Let's say that the following call, Figure 17, to GENERATE^HLMA is created:

```
D INIT^HLFNC2("ZZLJA RH ORU SERVER",.HL)
S HL("SUBSCRIBER")="^^^^^M^HLCSHDR4"
D GENERATE^HLMA("ZZLJA RH ORU SERVER","LM",1,.HLRST,"",.HL)
Q
```

**Figure 17: "Practice" M^HLCSHDR4 API invoked by HLP("SUBSCRIBER"[,n])**

Notice in Figure 17 that the "practice" M^HLCSHDR4 API is invoked by HLP("SUBSCRIBER"[,n]).

A sample terminal dialog seen when invoking this API is shown in Figure 18.

```
The original MSH segment is...

MSH^~|\&^VOICERAD^512^RADIOLOGY^512^20030116122809-0800^^ORU~R01
^9987540^T^2.3^^^AL^AL^

The MSH segment, after modification by passed-in data, is...

MSH^~|\&^AC-VOICERAD^512^AC-RADIOLOGY^512^20030116122809-0800^^ORU~R01
^9987540^T^2.3^^^AL^AL^

------------------------------------------------------------------------
    Protocol-derived value of SENDING APPLICATION: VOICERAD
Passed-in value of SENDING APPLICATION (HLMSHSAN): AC-VOICERAD
          Enter new value for SENDING APPLICATION: AC-VR<enter>

          The variable HLMSHSAN will be changed to 'AC-VR'.
          This value will be stored in the SENDING APPLICATION
          field in the MSH segment...

------------------------------------------------------------------------

    Protocol-derived value of SENDING FACILITY: 512
Passed-in value of SENDING FACILITY (HLMSHSFN):
          Enter new value for SENDING FACILITY: <enter>

          No changes will be made...

------------------------------------------------------------------------
    Protocol-derived value of RECEIVING APPLICATION:
Passed-in value of RECEIVING APPLICATION (HLMSHRAN):
          Enter new value for RECEIVING APPLICATION: <enter>

          No changes will be made...

------------------------------------------------------------------------

    Protocol-derived value of RECEIVING FACILITY: 512
Passed-in value of RECEIVING FACILITY (HLMSHRFN):
          Enter new value for RECEIVING FACILITY: <enter>

          No changes will be made...

------------------------------------------------------------------------

Before your changes above, the modified MSH segment was...

MSH^~|\&^AC-VOICERAD^512^AC-RADIOLOGY^512^20030116122809-0800^^ORU~R
^9987540^T^2.3^^^AL^AL^

After your changes, the MSH segment is...

MSH^~|\&^AC-VR^512^AC-RADIOLOGY^512^20030116122809-0800^^ORU~R01
^9987540^T^2.3^^^AL^AL^

------------------------------------------------------------------------
Message being sent...
```

Remember that M code controls routing-related fields by changing the four (legal to change) variables. This statement is saying that 'AC-VR' was entered by user, and that the HLMSHSAN variable was set to 'AC-VR'.

Application developers, in their own APIs must take similar action to change the sending application (and the other three fields.)

**Figure 18: M^HLCSHDR4 output shows which of the four local variables should be changed**

The dialog seen during the use of the M^HLCSHDR4 API, Figure 18, should make it clear to developers which of the four local variables should be changed, and the effect of the change on the MSH segment.

# Chapter 8. Technical Manual Information

This documentation is intended for use in conjunction with the **V*IST*A** Health Level Seven (HL7) MSH Segment Control (Dynamic Routing) software, exported in Patch HL*1.6*93. This is the Technical Manual section of this documentation. It is included in this documentation as required by the Veterans Health Administration Software Documentation Handbook. It details the implementation and maintenance of Patch HL*1.6*93, as well as routines, options, external and internal relations and software product security for the software.

## System Requirements

**For System Requirements:** Please refer to the patch description on the Patch Module (i.e., Patch User Menu [A1AE USER]) on FORUM.

**For Package Requirements**: Please refer to the "Package Requirements" section in "Implementation and Maintenance."

## Implementation and Maintenance

**V*IST*A** Health Level Seven Patch HL*1.6*93 is a Kernel Installation and Distribution System (KIDS) software release. Installation instructions can be found in the description for Patch HL*1.6*93, located on the Patch Module (i.e., Patch User Menu [A1AE USER]) on FORUM.

### Package Requirements

Patch HL*1.6*93 requires that both development Test and Production accounts exist in a standard **V*IST*A** operating environment in order to function correctly. The accounts must contain the *fully* patched versions of the following software:

- Health Level Seven (HL7) V. 1.6 (in particular, Patches HL*1.6*80 and HL*1.6*94)
- Kernel V. 8.0
- Kernel Toolkit V. 7.3
- RPC Broker V. 1.1
- VA FileMan V. 22.0

# Routines

This section lists the routines that are exported with Patch HL*1.6*93. They are categorized as updates to existing routines and new routines.

**Existing routine modified by Patch HL*1.6*93:**

HLCSHDR1

**New routines issued by Patch HL*1.6*93:**

HLCSHDR3
HLCSHDR4
HLCSHDR5

# Option

In order to assist developers in finding matching protocols on sending and receiving systems, Patch HL*1.6*93 introduces a new report created to list the protocol information that is used when searching for matching protocols. This new report can be printed using the Protocol Messaging Fields Report [HL PROTOCOL MSG FIELDS REPORT] menu option, also new with Patch HL*1.6*93. (This option is located in the Reports menu, under the Interface Developer Options menu.)

For more information on this new option exported with Patch HL*1.6*93, see the topic section titled "Protocol Messaging Fields Report."

# Archiving and Purging

There are no package-specific archiving or purging procedures or recommendations for Patch HL*1.6*93.

# Callable Routines

This section lists the one API, which is new with Patch HL*1.6*93.

This callable entry point is described in detail in the topic section titled "M^HLCSHDR4 API—M Code to Set Routing-Related Fields."

| Entry Point | Brief Description |
|---|---|
| M^HLCSHDR4 | Immediately before the MSH segment is constructed, application developers can invoke M code to directly set the local variables used for the routing-related fields in the MSH segment. This was discussed in the section titled "Variables Assist M Code."<br><br>The four variables that can be changed are listed below:<br>• HLMSHSAN New SENDING APPLICATION to be used in the MSH segment.<br>• HLMSHSFN New SENDING FACILITY to be used in the MSH segment.<br>• HLMSHRAN New RECEIVING APPLICATION to be used in the MSH segment.<br>• HLMSHRFN New RECEIVING FACILITY to be used in the MSH segment. |

**Table 11: Callable entry points exported with the Patch HL*1.6*93**

# External Interfaces (HL7 Components)

Patch HL*1.6*93 has no external interfaces. This patch only interacts with the DIRECT^HLMA and GENERATE^HLMA APIs *within* the **V*ISTA*** HL7 software.

For more information on the DIRECT^HLMA and GENERATE^HLMA APIs, please refer to the VistA HL7 Site Manager & Developer Manual.

# External Relations

Patch HL*1.6*93 has no relationships with packages or software structures outside the **V*ISTA*** HL7 software. However, Patch HL*1.6*93 introduces a new API named M^HLCSHDR4, which has been created for use by application software developers making calls to DIRECT^HLMA and GENERATE^HLMA. This API is used by application developers for training and help in use of this patch's software. This new API is a supported reference and covered under Integration Agreement #3988 on the FORUM DBA [DBA] menu.

For more information on the new M^HLCSHDR4 API, please refer to the section titled "Callable Routines."

# Internal Relations

There is a direct relationship between the DIRECT^HLMA and GENERATE^HLMA APIs in the **V*ISTA*** HL7 software and the software in Patch HL*1.6*93. When calls are made to these APIs, the calling

application may pass in the dynamic routing to be used to evaluate and reset the routing-related fields in the MSH segment for messages.

### Namespace

Patch HL*1.6*93 uses the HL namespace for the local variables created by this patch's software. All local variables newly created by HL*1.6*93 are NEWed prior to creation, and so do not exist beyond the scope of the creation of the MSH segment.

## Package-wide Variables

There are no package-wide variables exported with Patch HL*1.6*93.

# Software Product Security

## Mail Groups

There are no mail groups exported with Patch HL*1.6*93.

## Remote Systems

There are no remote systems involved with Patch HL*1.6*93.

## Archiving/Purging

There are no package-specific archiving or purging procedures or recommendations for Patch HL*1.6*93.

## Interfacing

No *non*-VA products are embedded in or required by Patch HL*1.6*93, other than those provided by the underlying operating systems.

## Electronic Signatures

There are no electronic signatures used within Patch HL*1.6*93.

## Security Keys

No security keys are exported with Patch HL*1.6*93.

# Appendix A: Quick Reference Guide to MSH Segment Control

As of Patch HL*1.6*93, users of the DIRECT^HLMA and GENERATE^HLMA APIs can now control the following MSH routing-related fields: SENDING APPLICATION, SENDING FACILITY, RECEIVING APPLICATION, and RECEIVING FACILITY. Control of these fields is obtained by passing data into these APIs used to set them at the moment the MSH segment is created. The syntax for calling these APIs is shown in Figure 19.

```
D [DIRECT/GENERATE]^HLMA(HLEID,HLARYTYP,HLFORMAT,.HLRESLT,HLMTIEN,.HLP)
```

**Figure 19: Syntax for calling the DIRECT^HLMA and GENERATE^HLMA APIs**

The passed by reference HLP array controls the routing-related fields in the MSH segment.

There are two legal syntax forms for HLP("SUBSCRIBER"[,n]) data shown in Figure 20.

```
HLP("SUBSCRIBER")=^[SAPP]^[SFAC]^[RAPP]^[RFAC]^[Subrtn^Rtn]"
HLP("SUBSCRIBER",n)=Sub101^[SAPP]^[SFAC]^[RAPP]^[RFAC]^[Subrtn^Rtn]"
```

**Figure 20: Two legal syntax forms for HLP("SUBSCRIBER"[,n]) data**

When [Subrtn^Rtn] is passed, both the "Subrtn" and the "Rtn" must be included. The "n" in the second entry represents a positive canonical number like 1 or 2. Immediately before the MSH segment is created, the data in these HLP("SUBSCRIBER"[,n]) entries is evaluated. If the subscriber protocol being used by DIRECT^HLMA or GENERATE^HLMA matches Sub101, shown in Figure 11, which is the data on that node is used to control the MSH segment. If no subscriber protocol match is found, the general entry HLP("SUBSCRIBER"), is used.

If M code exists, it evaluates the environment and can set the local variables used to create the MSH segment if appropriate. M code is always executed as the last step. Because of this, M code actions take precedence over all other changes.

Table 12 contains some of the most useful local variables that exist and can be evaluated when M code is executed.

| Variable Name | Description |
|---|---|
| HLMSHPRE | Event driver protocol IEN^NAME |
| HLMSHPRS | Subscriber protocol IEN^NAME |
| HLMSH772 | HL7 MESSAGE TEXT file (#772) internal entry number |
| HLMSH773 | HL7 MESSAGE ADMINISTRATION file (#773) internal entry number (if available) |

| Variable Name | Description |
|---|---|
| HLMSHSAO | Original SENDING APPLICATION |
| **HLMSHSAN** | **New SENDING APPLICATION to be used in the MSH segment** |
| HLMSHSFO | Original SENDING FACILITY |
| **HLMSHSFN** | **New SENDING FACILITY to be used in the MSH segment** |
| HLMSHRAO | Original RECEIVING APPLICATION |
| **HLMSHRAN** | **New RECEIVING APPLICATION to be used in the MSH segment** |
| HLMSHRFO | Original RECEIVING FACILITY |
| **HLMSHRFN** | **New RECEIVING FACILITY to be used in the MSH segment** |
| HLMSHTAG | M code subroutine |
| HLMSHRTN | M code routine |

**Table 12: Local variables that can be evaluated when M code is executed**

All these variables may be examined by M code. However, only the variables that are highlighted in boldface may be changed. To set the RECEIVING APPLICATION field in the MSH segment, M code should set the HLMSHRAN local variable. And, when M code sets the HLMSHRFN variable, that value is used in the RECEIVING FACILITY field in the MSH segment.

## Example

The following is an example of the HLP array-based control of these fields using string literals instead of M code:

```
S HLP("SUBSCRIBER")=^^^TALKLINK^512A5^^^M"
D GENERATE^HLMA(HLEID,HLARYTYP,HLFORMAT,.HLRESLT,HLMTIEN,.HLP)
```

**Figure 21: HLP("SUBSCRIBER"[,n]) array-based control of fields using string literals instead of M code**

Figure 21 shows that no M code is used. Rather, the literal value of "TALKLINK" will be used for the RECEIVING APPLICATION field in all MSH segments, and the value of "512A5" will be used for the RECEIVING FACILITY field in MSH segments.

Many more features are available for your use! Refer to this documentation for details.

# Glossary

| | |
|---|---|
| ACCEPT ACKNOWLEDGMENT | A message sent back to the sending system of a message acknowledging receipt of the message. |
| ACK | General Acknowledgment message. The ACK message is used to respond to a message where there has been an error that precludes application processing or where the application does not define a special message type for the response. |
| ADMISSION, DISCHARGE AND TRANSFER (ADT) TRANSACTION SET | Provides for transmitting new or updated demographic and visit information about patients. Generally information will be entered into an ADT system and passed to the nursing, ancillary and financial systems either in the form of an unsolicited update or in response to a record-oriented query. |
| ADT | Admission, Discharge and Transfer (ADT) message. |
| ANSI | American National Standards Institute. Founded in 1918, ANSI itself does not develop standards. ANSI's roles include serving as the coordinator for U.S. voluntary standards efforts, acting as the approval body to recognize documents developed by other national organizations as American National Standards, acting as the U.S. representative in international and regional standards efforts, and serving as a clearinghouse for national and international standards development information. |
| ANSI | **A**merican **N**ational **S**tandards **I**nstitute |
| APPLICATION | In the terminology of **V***IST***A** HL7, an application implements the interface between two systems. A **V***IST***A** HL7 application is stored in **V***IST***A** HL7 file entries. The application consists of one entry in the HL7 Application Parameter file, and *protocols* representing each transaction (i.e., HL7 message type), set up to be exchanged in the interface. |
| APPLICATION ACKNOWLEDGMENT | A message sent back to the sending system of a message acknowledging completion of processing of the message by the application. |
| APPLICATION PROGRAM INTERFACE (API) | Programmer calls provided for use by application programmers. APIs allow programmers to carry out standard computing activities without needing to duplicate utilities in their own packages. APIs also further DBA goals of system integration by channeling activities, such as adding new users, through a limited number of callable entry points. |
| ARRAY | An arrangement of elements in one or more dimensions. An M array is a set of nodes referenced by subscripts that share the same variable name. |
| CALLABLE ENTRY POINT | Authorized programmer call that may be used in any **V***IST***A** application software. The DBA maintains the list of DBIC-approved entry points. |
| CO | **C**entral **O**ffice |

| | |
|---|---|
| COMMIT ACKNOWLEDGMENT | See Accept Acknowledgment |
| CONTROLLED SUBSCRIPTION INTEGRATION AGREEMENT | This applies where the IA describes attributes/functions that must be controlled in their use. The decision to restrict the IA is based on the maturity of the custodian software. Typically, these IAs are created by the requesting software based on their independent examination of the custodian software's features. For the IA to be approved, the custodian grants permission to other **V***IST***A** software to use the attributes/functions of the IA; permission is granted on a one-by-one basis where each is based on a solicitation by the requesting package. An example is the extension of permission to allow a package (e.g., Spinal Cord Dysfunction) to define and update a component that is supported within the Health Summary package file structures. |
| CROSS REFERENCE | There are several types of cross-references available. Most generally, a VA FileMan cross-reference specifies that some action be performed when the field's value is entered, changed, or deleted. For several types of cross-references, the action consists of putting the value into a list; an index used when looking-up an entry or when sorting. The regular cross-reference is used for sorting and for lookup; you can limit it to sorting only. |
| DATA TYPE | HL7 provides a special set of HL7 data types. These are defined in Chapter 2. Page D-10 Health Level Seven, Version 2.4 © 2000. All rights reserved. November 2000. Final Standard. |
| DBIA | **D**ata**b**ase **I**ntegration **A**greement, a formal understanding between two or more **V***IST***A** software applications that describes how data is shared or how software interacts. The DBA maintains a list of DBIAs. |
| DCL | **D**igital **C**ommand **L**anguage |
| DEPARTMENT OF VETERANS AFFAIRS | The Department of **V**eterans **A**ffairs, formerly called the **V**eterans **A**dministration. |
| DHCP | **D**ecentralized **H**ospital **C**omputer **P**rogram of the Veterans Health Administration (VHA), Department of Veterans Affairs (VA) is the former name for Veterans Health Information Systems and Technology Architecture (**V***IST***A**). **V***IST***A** software, developed by VA, is used to support clinical and administrative functions at VA Medical Centers nationwide. It is written in M and, via the Kernel, runs on all major M implementations regardless of vendor. **V***IST***A** is composed of packages that undergo a verification process to ensure conformity with namespacing and other **V***IST***A** standards and conventions. |
| EVENT DRIVER PROTOCOL | For **V***IST***A** HL7, an event driver protocol represents the sending application's side of a transaction for a particular message type/event type, whether the message originates on the **V***IST***A** side of the interface, or on the other side of the interface. |
| EVENT TYPE | Trigger event code (one component of a message header's Message Type field) defined by *HL7 table 0003 - Event type*. |

| | |
|---|---|
| EXTRINSIC FUNCTION | Extrinsic function is an expression that accepts parameters as input and returns a value as output that can be directly assigned. |
| FACILITY | Geographic location at which VA business is performed. |
| FIELD | In a record, a specified area used for the value of a data attribute. The data specifications of each VA FileMan field are documented in the file's data dictionary. A field is similar to blanks on forms. It is preceded by words that tell you what information goes in that particular field. The blank, marked by the cursor on your terminal screen, is where you enter the information. |
| FILE | Set of related records treated as a unit. VA FileMan files maintain a count of the number of entries or records. |
| FILE MANAGER (VA FILEMAN) | **V***IST***A**'s Database Management System (DBMS). The central component of Kernel that defines the way standard **V***IST***A** files are structured and manipulated. |
| FILER | One of **V***IST***A** HL7's background processes. *Incoming* filers move received messages over **V***IST***A** HL7 HLLP and X3.28 links to **V***IST***A** HL7's messages files, and process the message. *Outgoing* filers deliver messages generated by **V***IST***A** programs to HLLP, X3.28 and MailMan links for delivery to other systems. |
| FORM | Please refer to the Glossary entry for "SCREENMAN FORMS." |
| GLOBAL VARIABLE | Variable that is stored on disk (M usage). |
| HEADER | The first segment in an HL7 message. Usually it is a message header segment (MSH) but it can also be the batch header segment (BHS) or file header segment (FHS). The header defines the intent, source, destination, and some specifics of the syntax of a message. |
| HEALTH LEVEL SEVEN (HL7) | National level standard for data exchange in all healthcare environments regardless of individual computer application systems. |
| HEALTH LEVEL SEVEN (HL7) **V***IST***A** | Messaging system developed as a **V***IST***A** software that follows the HL7 Standard for data exchange. |
| HISPP | Healthcare Informatics Standards Planning Panel. HISPP was formed in early 1992. HISPP is charged with coordinating the work of the standards groups for healthcare data interchange and healthcare informatics (e.g., HL7), and other relevant standards groups (e.g., ASC X12) toward achieving the evolution of a unified set of non-redundant, non-conflicting standards that are compatible with ISO and non-ISO communications environments. HISPP also interacts with and provides input to CEN/TC251 in a coordinated fashion and explores avenues of international standards development (e.g., ISO). |
| HL7 BATCH PROTOCOL | Protocol utilized to transmit a batch of HL7 messages. The protocol uses FHS, BHS, BTS and FTS segments to delineate the batch. |
| HYBRID LOWER LAYER | A *Lower Layer Protocol* for communications over a LAN that may |

| | |
|---|---|
| PROTOCOL (HLLP) | involve RS-232 connections. It is a more efficient LLP than X3.28, but still suitable for use over generally error-free RS-232 connections. |
| IIS (NOW IS ISS– **I**NFRASTRUCTURE AND **S**ECURITY **S**ERVICES.) | **I**nformation **I**nfrastructure **S**ervice |
| INPUT TEMPLATE | A pre-defined list of fields that together comprise an editing session. |
| INTEGRATION AGREEMENTS (IA)<br>(FORMERLY KNOWN AS DATABASE INTEGRATION AGREEMENTS [DBIA]) | Integration Agreements define an agreement between two or more **V***IST***A** packages to allow access to one development domain by another. Any package developed for use in the **V***IST***A** environment is required to adhere to this standard; as such it applies to vendor products developed within the boundaries of DBA assigned development domains (e.g., MUMPS AudioFax). An IA defines the attributes and functions that specify access. All IAs are recorded in the Integration Agreement database on FORUM. Content can be viewed using the DBA menu or the Technical Services' web page. |
| INTERNAL ENTRY NUMBER (IEN) | The number used to identify an entry within a file. Every record has a unique internal entry number. |
| IRM | **I**nformation **R**esource **M**anagement. A service at VA medical centers responsible for computer management and system security. |
| ISS (FORMERLY KNOWN AS IIS–**I**NFORMATION **I**NFRASTRUCTURE **S**ERVICE) | **I**nfrastructure and **S**ecurity **S**ervices. |
| ITAC | **I**nformation **T**echnology **A**pproval **C**ommittee was established as an advisory committee to the Chief Information Officer to ensure that the Information Technology (IT) program supports VHA goals and to provide guidance concerning priorities for IT initiatives. |
| IV&V | **I**ndependent **V**alidation and **V**erification team acts to ensure the functional integrity and technical correctness of software, processes and documentation. |
| KERNEL | Kernel is **V***IST***A** software that functions as an intermediary between the host operating system and other **V***IST***A** software applications (e.g., Laboratory, Pharmacy, IFCAP, etc.). Kernel provides a standard and consistent user and programmer interface between software applications and the underlying M implementation. |
| LINK | An entry in the HL LOGICAL LINK file (#870) that links **V***IST***A** HL7 to a particular destination. The destination may be a device (defined in the DEVICE file (#3.5)), a TCP/IP address and port, or a system that is reached through VA MailMan. |
| LOGICAL LINK | Please refer to the Glossary entry for "LINK." |

| | |
|---|---|
| LOWER LAYER PROTOCOL (LLP) | Defined by the *HL7 Implementation Guide*. LLPs provide for a degree of lower layer communications functionality, such as flow control and error recovery, needed between systems in a networked environment. Examples include *Hybrid Lower Layer Protocol* (HLLP) and *Minimal Lower Layer Protocol* (MLLP). |
| MAILMAN | **V***IST***A** software that provides a mechanism for handling electronic communication, whether it's user-oriented mail messages, automatic firing of bulletins, or initiation of server-handled data transmissions. |
| MENU | List of choices for computing activity. A menu is a type of option designed to identify a series of items (other options) for presentation to the user for selection. When displayed, menu-type options are preceded by the word "Select" and followed by the word "option" as in Select Menu Management option: (the menu's select prompt). |
| MENU SYSTEM | The overall Menu Manager logic as it functions within the Kernel framework. |
| MESSAGE | From the HL7 standard, "A message is the atomic unit of data transferred between systems. It is comprised of a group of segments in a defined sequence. Each message has a message type that defines its purpose. For example, the ADT Message type is used to transmit portions of a patient's ADT data from one system to another. A three character code contained within each message identifies its type."1 |
| MESSAGE DELIMITERS | In constructing a message certain characters are used. These include the Segment Terminator, the Field Separator, the Component Separator, the Sub- Component Separator, Repetition Character, and the Escape Character. |
| MESSAGE TYPE | Each message has a message type that defines its purpose. For example, the ADT Message Type is used to transmit portions of a patient's ADT data from one system to another. A 3-character code contained within each message identifies its type. Message type code (one component of a message header's Message Type field) defined by *HL7 table 0076 - Message type*. |
| MINIMAL LOWER LAYER PROTOCOL (MLLP) | A very simple *Lower Layer Protocol* that simply delimits messages. It is used in LAN-based networking environments such as TCP/IP that provide a reliable byte stream (flow control and error recovery are provided by the network), but insufficient session control to support HL7. |
| MSH SEGMENT | The **V***IST***A** HL7 MSH segment defines the intent, source, destination, and some specifics of the syntax of a message. The Message Header (MSH) segment is always the first segment in every HL7 message (except for batch HL7 messages, which begin with BHS or FHS segments). This HL7 segment contains control fields that characterize any one particular HL7 message. Of particular interest to Patch HL*1.6*93, the MSH segment also contains these four routing-related |

1 Health Level Seven, *Health Level Seven, Version 2.3.1*, copyright 1999, p. E-18.

| | |
|---|---|
| | fields:<br>SENDING APPLICATION<br>SENDING FACILITY<br>RECEIVING APPLICATION<br>RECEIVING FACILITY |
| NAMESPACING | Convention for naming **V*IST*A** package elements. The DBA assigns unique character strings for package developers to use in naming routines, options, and other package elements so that packages may coexist. The DBA also assigns a separate range of file numbers to each package. |
| OIFO | **O**ffice of **I**nformation **F**ield **O**ffice |
| OPTION | An entry in the OPTION file (#19). As an item on a menu, an option provides an opportunity for users to select it, thereby invoking the associated computing activity. Options may also be scheduled to run in the background, non-interactively, by Task Manager. |
| PERSISTENT | When a link is persistent, the connection between the two systems is kept open even when no messages are being exchanged. When a link is non-persistent, the connection is closed when there are no messages to exchange. |
| PRIVATE INTEGRATION AGREEMENT | Where only a single application is granted permission to use an attribute/function of another **V*IST*A** package. These IAs are granted for special cases, transitional problems between versions, and release coordination. A Private IA is also created by the requesting package based on their examination of the custodian package's features. An example would be where one package distributes a patch from another package to ensure smooth installation. |
| PROMPT | The computer interacts with the user by issuing questions called prompts, to which the user issues a response. |
| QUERYING APPLICATION | A querying application neither exerts control over, nor requests changes to a schedule. Rather than accepting unsolicited information about schedules, as does an auxiliary application, the querying application actively solicits this information using a query mechanism. It will be driven by a person wanting information about schedules, and may be part of an application filling the placer application role as defined in this chapter. The information that the querying application receives is valid only at the exact time that the query results are generated by the filler application. Changes made to the schedule after the query results have been returned are not communicated to the querying application until it issues another query transaction. |

| | |
|---|---|
| RECEIVING APPLICATION | This is a site-defined field located in the MSH segment that uniquely identifies the receiving application among all other applications that participate in the exchange of HL7 messages in **V*IST*A**.<br><br>A subscriber protocol represents the receiving application's side of a particular transaction for a particular message type/event type, whether the message is being received by the **V*IST*A** side of the interface, or by the other side of the interface. In both cases, a subscriber protocol must be set up in **V*IST*A** HL7, and the required fields are Sending Application, Message Type, Event Type and Version ID. |
| RECEIVING FACILITY | This is a site-defined field located in the MSH segment that identifies the receiving application among multiple identical instances of the application running on behalf of different organizations. The role of the Receiving Facility field is to ensure that when the message is received, it is received at the appropriate facility. |
| RECEIVING-RELATED FIELDS | Within the **V*IST*A** HL7 MSH segment, the RECEIVING APPLICATION and RECEIVING FACILITY are the "receiving-related" fields. They are site-defined fields that uniquely identify the receiving application and ensure that when the message is received, it is received at the appropriate facility. |
| ROUTINE | Program or a sequence of instructions called by a program that may have some general or frequent use. M (previously referred to as MUMPS) routines are groups of program lines, which are saved, loaded, and called as a single unit via a specific name. |
| ROUTING-RELATED FIELDS | The **V*IST*A** HL7 MSH segment contains these four routing-related:<br>SENDING APPLICATION<br>SENDING FACILITY<br>RECEIVING APPLICATION<br>RECEIVING FACILITY |
| SCREENMAN FORMS | Screen-oriented display of fields, for editing or simply for reading. VA FileMan's Screen Manager is used to create forms that are stored in the FORM file (#.403) and exported with a package. Forms are composed of blocks (stored in the BLOCK file [#.404]) and can be regular, full screen pages or smaller, "pop-up" pages. |
| SEGMENT | From the HL7 standard, "An HL7 segment is a logical grouping of data fields. Segments of a message may be required or optional. They may occur only once in a message or they may be allowed to repeat. Each segment is identified by a unique three character code known as the Segment ID."2 |

2 Health Level Seven, *Health Level Seven, Version 2.3.1*, copyright 1999, p. E-28.

| | |
|---|---|
| SEGMENT (RECORD) | A typed aggregate of fields (fields) describing one complete aspect of a message. For example, the information about one order is sent as type of segment (OBR), the information related to an observation is sent as another segment (OBX). The segment in a message is analogous to a record in a database, and in previous versions of the standard we used record in place of the word segment. We have changed the nomenclature to be consistent with HL7 and other standards organizations in this version. |
| SEGMENT TERMINATOR | The segment terminator is the last character of every segment. It is always the ASCII CR character (hex 0D). |
| SENDING APPLICATION | This is a site-defined field located in the MSH segment that uniquely identifies the sending application among all other applications that participate in the exchange of HL7 messages in **V*IST*A**.<br><br>An event driver protocol represents the sending application's side of a transaction for a particular message type/event type, whether the message originates on the **V*IST*A** side of the interface, or on the other side of the interface. In both cases, an event driver protocol must be set up in **V*IST*A** HL7, and the required fields are Sending Application, Message Type, Event Type and Version ID. |
| SENDING FACILITY | This is a site-defined field located in the MSH segment that further describes the sending application and identifies the facility sending the message. |
| SENDING-RELATED FIELDS | Within the **V*IST*A** HL7 MSH segment, the SENDING APPLICATION and SENDING FACILITY fields are referred to as "sending-related" fields. They are site-defined fields that uniquely identify and describe the sending application and the facility sending the message. |
| SUBSCRIBER | An application that subscribes to a particular event point, registering its interest so it can receive unsolicited updates. |
| SUBSCRIBER PROTOCOL | For **V*IST*A** HL7, a subscriber protocol represents the receiving application's side of a particular transaction for a particular message type/event type, whether the message is being received by the **V*IST*A** side of the interface, or by the other side of the interface. |
| SUPPORTED REFERENCE INTEGRATION AGREEMENT | This applies where any **V*IST*A** application may use the attributes/functions defined by the IA (these are also called "Public "). An example is an IA that describes a standard API such as DIE or VADPT. The software that creates/maintains the Supported Reference must ensure it is recorded as a Supported Reference in the IA database. There is no need for other **V*IST*A** software applications to request an IA to use these references; they are open to all by default. |
| TCP/IP | **T**ransmission **C**ontrol **P**rotocol/**I**nternet **P**rotocol |

| | |
|---|---|
| TEMPLATE | Means of storing report formats, data entry formats, and sorted entry sequences. A template is a permanent place to store selected fields for use at a later time. Edit sequences are stored in the INPUT TEMPLATE file (#.402), print specifications are stored in the PRINT TEMPLATE file (#.4), and search or sort specifications are stored in the SORT TEMPLATE file (#.401). |
| TOOLKIT | Toolkit (or Kernel Toolkit) is a robust set of tools developed to aid the **V***IST***A** development community, and Information Resources Management (IRM), in writing, testing, and analysis of code. They are a set of generic tools that are used by developers, technical writers, software quality assurance (SQA) personnel, and software applications to support distinct tasks.<br><br>Toolkit provides utilities for the management and definition of development projects. Many of these utilities have been used by the OI Field Office–Oakland for internal management and have proven valuable. Toolkit also includes tools provided by other OI Field Offices based on their proven utility. |
| TRIGGER | A type of VA FileMan cross-reference. Often used to update values in the database given certain conditions (as specified in the trigger logic). For example, whenever an entry is made in a file, a trigger could automatically enter the current date into another field holding the creation date. |
| TRIGGER EVENT | The event that initiates an exchange of messages is called a trigger event. The HL7 Standard is written from the assumption that an event in the real world of health care creates the need for data to flow among systems. The real-world event is called the trigger event. For example, the trigger event "a patient is admitted" may cause the need for data about that patient to be sent to a number of other systems. There is a one-to-many relationship between message types and trigger event codes. The same trigger event code may not be associated with more than one message type. |
| UNSOLICITED UPDATE | When the transfer of information is initiated by the application system that deals with the triggering event, the transaction is termed an unsolicited update. |
| VA FILEMAN | Set of programs used to enter, maintain, access, and manipulate a database management system consisting of files. A package of online computer routines written in the M language, which can be used as a stand-alone database system or as a set of application utilities. In either form, such routines can be used to define, enter, edit, and retrieve information from a set of computer stored files. |
| VAMC | **V**eterans **A**ffairs **M**edical **C**enter. |

| | |
|---|---|
| VARIABLE | Character, or group of characters, that refers to a value. M (previously referred to as MUMPS) recognizes 3 types of variables: local variables, global variables, and special variables. Local variables exist in a partition of main memory and disappear at sign-off. A global variable is stored on disk, potentially available to any user. Global variables usually exist as parts of global arrays. The term "global" may refer either to a global variable or a global array. A special variable is defined by systems operations (e.g., $TEST). |
| **V*IST*A** | **V**eterans Health **I**nformation **S**ystems and **T**echnology **A**rchitecture (**V*IST*A**) of the Veterans Health Administration (VHA), Department of Veterans Affairs (VA). **V*IST*A** software, developed by the VA, is used to support clinical and administrative functions at VHA sites nationwide. Server-side code is written in M, and, via Kernel, runs on all major M implementations regardless of vendor. **V*IST*A** is composed of software that undergoes a quality assurance process to ensure conformity with namespacing and other **V*IST*A** standards and conventions. |
| X3.28 | A *Lower Layer Protocol* based on ANSI X3.28, for use over simple RS-232 circuits where flow control and error recovery are major issues. |
| Z SEGMENT (HL7) | All message type and trigger event codes beginning with Z are reserved for locally defined messages. No such codes will be defined within the HL7 Standard. |

# Index

## F



## G



## H