VistA Health Level Seven (HL7) Event Monitoring

Supplement to Patch Description

Patch HL*1.6*109

February 2004

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

REVISION HISTORY

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

Date	Revision	Description	Author
02/2004	HL*1.6*109	Supplement to patch description for Patch HL*1.6*109.	Jim Moore, Larry Andreassen, Randy Stone

Patch Revisions

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

TABLE OF CONTENTS

ORIENTATION

This documentation uses several methods to highlight different aspects of the material. “Snapshots” of computer dialogue (or other online displays) are shown in a non-proportional font and enclosed within a box. User responses to on-line prompts are highlighted in boldface. Boldface is also used to highlight a descriptive word or sentence. The Return or Enter key is illustrated by the symbol **<Enter>** when displayed in computer dialogue and is included in examples only when it may be unclear to the reader that such a keystroke must be entered. The following example indicates that you should type two question marks followed by pressing the Return or Enter key when prompted to select an option:

```
Select Primary Menu option: ??
```

Figure 99: How to access online help

M (MUMPS – Massachusetts General Hospital Utility Multi-Programming System) code, variable names, acronyms, the formal name of options, actual field names, file names, and security keys are represented with all uppercase letters.

1. INTRODUCTION

The purpose of the **V***IST***A** HL7 Event Monitor software is to monitor the **V***IST***A** HL7 environment. The Event Monitor system is built around *monitors* - background tasks running M code - and a *master job* that starts the background monitors.

Monitors are background jobs running M code. Each monitor is built around an entry in the HL7 MONITOR file (#776.1). The entry in this file contains crucial information about the monitor, such as how often the monitor should be run and the M code API to call to start the monitor.

The master job is a background job that runs repetitively on a site-configurable schedule. The master job evaluates every existing monitor to determine whether a background job should be started. If the monitor is active, and if other criteria are met (such as the requeuing frequency), the master job starts a background job for the monitor. When it does so, it records its action in the HL7 MONITOR MASTER JOB file (#776.2) its action.

Even if the master job does not start an individual monitor, it still records the reason why the monitor was not started in the HL7 MONITOR MASTER JOB file (#776.2).

The master job is started the first time after the Event Monitor is installed. The master job's first action is to queue in the future a new master job. A site parameter controls the requeue frequency (i.e., number of minutes between master jobs) of the master job. Then, the master job evaluates all the existing monitors to determine whether they should be run.

> ***Note:*** *Monitors come with the Event Monitor software, and other event monitors may be created locally. Additional monitors, in addition to the monitors released initially with patch HL*1.6*109, will be released through* **V***IST***A** *HL7 patches.*
>
> ***Note:*** *The master job never runs monitor code directly. Monitors always run in their own dedicated background process queued by the master job.*

The master job and event monitors are described in more detail below.

2. MENU SYSTEM

The Event Monitor software is accessed by the "Event Monitoring menu [HLEV MENU MAIN] menu option." The suboptions under the "Event Monitoring menu" are shown below.

```
Event monitoring menu   [HLEV MENU MAIN]
System: NXT,KDE                                              AUG 25,
2003@14:14
===========================================================================

Setup & maintenance options   [HLEV MENU SETUP]
|   Monitor setup & maintenance   [HLEV MENU SETUP-MONITOR]
|   |   Enter/edit event monitors   [HLEV EDIT MONITOR]
|   |   Monitor setup details   [HLEV PRINT MONITOR SETUP]
|   |   Turn on/off event monitor   [HLEV EDIT MONITOR ON-OFF]
|   System setup & maintenance   [HLEV MENU SETUP-MASTER]
|   |   Edit parameters   [HLEV EDIT MASTER]
|   |   Settings of monitoring parameters   [HLEV PRINT MASTER SETUP]
|   |   Turn on/off monitoring   [HLEV EDIT MASTER ON-OFF]
|   |   Monitoring master job start   [HLEV MASTER JOB START]
|   |   Stop monitoring master job   [HLEV MASTER JOB STOP]
|   |   Grant remote request license   [HLEV GRANT REMOTE LICENSE]
|   Reports   [HLEV MENU REPORTS]
|   |   Message recipients   [HLEV REPORT MONITOR RECIPIENTS]
|   |   Condensed monitor report   [HLEV REPORT CONDENSED MONITOR]
|   |   Remote requestable report   [HLEV REPORT REMOTE REQUESTABLE]
Run-time options   [HLEV MENU RUNTIME]
|   Settings of monitoring parameters   [HLEV PRINT MASTER SETUP]
|   Monitor setup details   [HLEV PRINT MONITOR SETUP]
|   One-time monitor run   [HLEV ONE-TIME MONITOR RUN]
|   Map of monitoring activity   [HLEV MONITOR MAP REPORT]
|   Results of a monitor run   [HLEV MONITOR DETAILS]
|   Run monitor master job now   [HLEV MASTER JOB NOW]
```

3. SYSTEM PARAMETERS

The Event Monitor is controlled using site parameters. These parameters are edited using the Edit parameters [HLEV EDIT MASTER] menu option. The terminal dialogue seen when editing parameters is shown next. The explanation for each parameter is included in the terminal dialogue.

```
========================= MASTER JOB STATUS ================================
Set this field to ACTIVE to enable the master job to run and monitor your
system.  (The master job is started and stopped using the 'Turn on/off
monitoring system [HLEV EDIT MASTER ON-OFF]' menu option.)  Set this field to
INACTIVE to stop the master job (if it is running), and to ensure that the
master job does not start

STATUS-MASTER: ACTIVE//

=================== MASTER JOB INTERVAL (MINUTES) ==========================
The master job is started every MASTER JOB INTERVAL minutes to evaluate your
system.  Enter the number of minutes now that should elapse between every
"run" of the master job.

REQUEUE MIN-MASTER JOB: 120//


                        --- EVENT MONITORING FIELDS ---


===================== STATUS OF EVENT MONITORING ===========================
The master job periodically "fires off" event monitors.  If you set this field
to INACTIVE, the master job will continue to start and run, but no events will
be started.  When this field is set to ACTIVE, the master job will be able to
run event monitors.

STATUS-EVENT: ACTIVE//

============================ PURGE LIMIT FOR DATA==========================
Whenever the master job runs, data is created in the HL7 Monitor Master Job
file (#776.2.)  Whenever the master job spawns off a new background job for an
event monitor, data is created in the HL7 Monitor Job file (#776.)  Purging of
this data occurs automatically.  This parameter controls how much data to
retain.  For example, if you enter '96' now, then no data less than 96 hours
old will be purged.

PURGE HOURS: 96//

Press RETURN to exit...
```

4. MONITOR CREATION

The primary menu for the event monitoring system is Event Monitoring [HLEV MENU MAIN]. Use the Enter/Edit Event Monitors menu option under this primary menu to enter the monitor that calls your M code. During the use of the Enter/Edit Event Monitors' menu option, you will see similar dialogue to that shown below. Explanation of the editable fields is included in the terminal dialogue.

```
                       Event Monitoring System Enter/Edit
===============================================================================
You may now enter new entries, and edit existing entries.  Enter a new entry
now, or select the existing entry to be edited.

Select EVENT MONITOR ENTRY: LINK (870) CHECKS

                ---------------- editing entry ----------------

NAME: LINK (870) CHECKS//

========================= SHORT DESCRIPTION (#3) =============================
Enter a short description for this event monitor; something that is more
complete and descriptive than the NAME.

SHORT DESCRIPTION: File 870 entry w/stub status search
           Replace

=============================== STATUS (#2) ==================================
Enter ACTIVE to make this event monitor "available" to the master job for
queueing.  When set to ACTIVE the master job will run this event monitor
according to the REQUEUE FREQUENCY (that you will be asked several prompts
from now).

NOTE:  If you're entering this event monitor for the first time, you should
       set this field to INACTIVE until all fields have been filled in.  Then,
       change this field back to ACTIVE.

STATUS: ACTIVE//

============================ M STARTUP LOCATION ===============================
The master job uses this field to determine how to start this event monitor.
So, enter the M location (subroutine and routine) where the event should be
queued.  Enter it in the SUBROUTINE~ROUTINE format, substituting a tilde (~)
for the up-arrow.

The M location you enter now is the location where queued jobs start.

M STARTUP: CHK870~HLEVX000//

========================== REQUEUE FREQUENCY (#4) =============================
The master job will run this event monitor as often as you specify.  And, this
field is the way you specify rerun frequency.  Enter the number of minutes
that should elapse after this event monitor runs until it is run again.

NOTE:  Enter '0' if you want this event to run every time the master job
       checks this monitor.

REQUEUE MIN-MONITOR: 720//
```

```
==================== MAIL GROUPs, USERs, REMOTE USERs ======================
Enter the mail groups and local users and remote users to which notification
messages are to be sent.  If no notification message will ever be sent, leave
these fields blank.

Select MAIL GROUPS:
Select RECIPIENTS:
Select REMOTES: HL7DevelopmentTeam@med.va.gov//

=================== M START CHECK (EXTRINSIC FUNCTION) =====================
Normally, the master job uses the monitor's requeue frequency in order to
determine whether a new monitor job should be queued.  Alternately, you may
call an extrinsic function to determine whether a new monitor job should be
started.  Entry of the M check extrinsic function is optional.

Extrinsic functions must follow these rules:

 * Syntax = $$TAG~ROUTINE (where TAG and ROUTINE do not exceed 8 characters.)
 * $$TAG~ROUTINE returns a 1 or 0.

The extrinsic function should return '0' if a new monitor job should not be
started, or a '1' to start a new monitor job.

M START CHECK:

============================= PARAMETER NOTES ================================
Enter description and documentation of the just entered parameters.

TECHNICAL DESCRIPTION:
M code logic flow for this routine is:

 - Loops through file 870 storing in ^XTMP all stub entries.

 - Loops through file 870 again, after requeue-minutes, searching for all stub
entries.  Compares to previous record of stub entries.  If a stub entry still
remains from last loop, a notification message is sent off-station to the
VistA HL7 team.

  Edit? NO//

=========================== EVENT MONITOR NOTES ==============================
Enter overall comments about this event monitor.

EVENT DESCRIPTION:
This event monitor search the IN QUEUE and the OUT QUEUE of the HL Logical
Link file (#870) for entries that are stuck in the STUB status.

  Edit? NO//

Run monitor now? No//   NO

You may queue this monitor to run "one-time" in the future.  If so, enter a
future date/time now...

Enter future run time:
```

5. FILES

The event monitor system is built around three setup-related files, and three run-time files:

Setup Files:

- ❑ HL7 MONITOR (#776.1)
- ❑ HL7 MONITOR EVENT TYPE (#776.3)
- ❑ HL7 MONITOR PARAMETERS (#776.999)

Run-time Files:

- ❑ HL7 MONITOR JOB (#776)
- ❑ HL7 MONITOR EVENT (#776.4)
- ❑ HL7 MONITOR MASTER JOB (#776.2)

6. Run-time Details

The "engine" that runs the Event Monitor system is the master job. And, the first master job is queued immediately after the Event Monitor software is installed. The installation software also makes an entry for the queued master job in the HL7 MONITOR MASTER JOB file (#776.2). When the queued background job for the master job becomes active, it uses the already created entry in the HL7 MONITOR MASTER JOB file (#776.2) as a location to record its activities and actions.

When a master job runs, as its first action, it queues a new master job sometime in the future, and creates a new entry in the HL7 MONITOR MASTER JOB file (#776.2) for the job queued to the future. The new master job is queued into the future the number minutes specified in the REQUEUE MIN-MASTER JOB field in the HL7 MONITOR PARAMETERS file (#776.999).

After the master job queues the future master job (as described above), it continues with its work. It evaluates every existing monitor in the HL EVENT MONITOR file. If an event is active and it is time for the monitor to run, the master job queues a new background job to do the monitor work.

There are two methods to control whether a monitor should be run by the master job:

- The REQUEUE MIN-MONITOR field (#4) in the HL7 MONITOR file (#776.1).
- The M START CHECK field (#7) in the HL7 MONITOR file (#776.1).

The REQUEUE MIN-MONITOR field is a numeric field holding the number of minutes between each "run" of its M code. If this field is filled in, and if the M START CHECK field is not present, the master job will start a new job every REQUEUE MIN-MONITOR number of minutes.

The M START CHECK can be used to reference an M extrinsic function to determine whether the monitor should be started. The syntax for this field's value is $$TAG^ROUTINE. An example entry might be $$RUNOW^HLEVUTIL. The value returned by the extrinsic function should be 1 if the monitor should be started, or null if not.

When the master job finds a valid extrinsic function reference in the M START CHECK, and the extrinsic function returns a positive value, the master job queues a background job for the monitor. (If the M START CHECK field is filled in the REQUEUE MIN-MONITOR field is ignored.)

> ***Note:*** *The STATUS field in the HL EVENT MONITOR file determines whether a monitor is active. If a monitor has never run, (and the status is active), the master job will always run the monitor. If the monitor has run before, The REQUEUE MIN MONITOR field is used to determine whether it is time for the monitor to run. If the number of minutes specified in the REQUEUE MIN MONITOR field has elapsed since the monitor's last run time, the master job will start a new background job for the monitor.*

As the master job evaluates every monitor, it makes a record in the HL7 MONITOR MASTER JOB file (#776.2) of its actions for every monitor. The most usual actions for a monitor are:

- ❑ No background job started for monitor because it was too early to run another background job for monitor.
- ❑ Queued a new background job for monitor.

The other possible master job actions are:

- No background job started for monitor because monitor was inactive.
- No background job started for monitor because some problem with the monitor was detected.
- No background job started for monitor because the monitor is currently running.

When the master job queues a new job for a monitor it performs two actions (as already stated):

- Creates a record of the monitor creation in the HL7 MONITOR MASTER JOB file (#776.2).
- Creates a new stub record in the HL7 MONITOR JOB file (#776) for recording use by the just queued monitor job.

When the monitor job becomes active, and when it finishes, it also updates the two files listed above.

7. Application Programmer Interfaces (APIs)

When application developers write monitors, they must do the following:

- Write the M code to do the monitoring and/or actions.
- Embed in their M code application programmer interface (API) calls to the event monitoring system.

The APIs available for use by the **V***IST***A** HL7 team when creating M code-based monitors are listed and explained below. The next section details how to use these APIs in M code to create a monitor.

Several times in the API documentation below the HLEVIENE and HLEVIENJ variables are included in parameter subscripts passed into the API. These two variables are automatically defined when the monitor starts, and can be referenced by the monitors at any time, including these examples where the IENs are passed into the APIs.

7.1. START^HLEVAPI(VAR)

Called at beginning of a "monitor run." (The master job queues a background task for every monitor. When the monitor starts, it calls out to the M code reference in the monitor event. The called M code must call this API at the top of processing.) The VAR parameter is passed by reference and defines the variables to be tracked.

M Code Examples:

```
Direct Variable Declaration.
D START^HLEVAPI("CT^IEN")

Pass-by-reference Variable Declaration
S VAR("CT")="CT-773",VAR("IEN")="IEN-773"
D START^HLEVAPI(.VAR)

No Variables Declared
D START^HLEVAPI()
```

7.2. CHECKIN^HLEVAPI

Called periodically during the "run" to update timestamp, tracking variables, etc.

M Code Example:

```
D CHECKIN^HLEVAPI
```

7.3. CHECKOUT^HLEVAPI

Called to mark a monitor "run" complete and store final values of variables being tracked.

M Code Example:

```
D CHECKOUT^HLEVAPI
```

7.4. ABORT^HLEVAPI

Called when a monitor stops prematurely. (Monitors may be stopped by TaskManager request to stop.)

In the following M code examples, reference is made to *status* and *application status*. These are separate fields in the HL7 MONITOR JOB file (#776), which are optionally set by the abort API call. The respective fields are:

- STATUS-RUNTIME (#4)
- STATUS-APPLICATION (#5)

If a "simple call" is made, not passing any of the statuses, the STATUS-RUNTIME is automatically set to ERROR. If desired, the ERROR status can be overridden as in the "Status-supplied Call" example below.

If desired, the STATUS-APPLICATION field may be filled in by the ABORT API call. If this field is not passed (as in the last example below) the field is left blank.

Syntax: D ABORT^HLEVAPI([status-runtime],[status-application])

The allowable values for status-runtime are:

- "A" - to set the status-runtime to ACTIVE.
- "E" - to set the status-runtime to ERROR.

Any free-text value for the status-application is allowable with a 10 character length limit.

M Code Examples:

Simple Call
```
D ABORT^HLEVAPI()
```

Status-supplied Call
```
D ABORT^HLEVAPI("A")
```

Status & Application Status-supplied Call
```
D ABORT^HLEVAPI("E","INACTIVE")
```

7.5. MAILIT^HLEVAPI

Called by application to send a mail message. (Recipients can be specified by a mail group pointer in the HL EVENT MONITOR file, or can be specified at the time of call to MAILIT.

M Code Example:

```
D MAILIT^HLEVAPI
```

Additional control is possible over the content of the message as explained in the "Writing Monitors" section below.

7.6. MSGTEXT^HLEVAPI1(GBL)

Called place text in the Event Text WP field in the HL7 MONITOR JOB file (#776). The text in this WP field may be included in the mail message sent to recipients.

M Code Example:

```
D MSGTEXT^HLEVAPI1($NA(^TMP($J,"TXT")))
```

The above example places the text in ^TMP($J,"TXT") in the Event Text multiple in the HL7 MONITOR JOB file (#776). It is required that the ^TMP($J,"TXT") global, or whatever global is declared in the parameter, to contain only one more numeric subscript deeper than the declared global reference.

Examples of valid global references:

- ^TMP($J,"TXT",1)="Some text"
- ^TMP($J,"TXT",25)=""

7.7. RUNDIARY^HLEVAPI1(GBL)

Call to place text in the Run Diary WP field in the HL7 MONITOR JOB file (#776). This field is displayed on-screen by Event Monitoring options, but is not included in the email message text.

M Code Example:

```
D RUNDIARY^HLEVAPI1($NA(^TMP($J,"TXT")))
```

The same global reference rules apply to this API as to the EVENTEXT^HLEVAPI1 API.

7.8. ONOFFM^HLEVAPI0(IENE)

Called to turn on or off a specific monitor. (Turning off the event monitor stops the master job from queueing further "runs" for the monitor.)

M Code Example:

```
D ONNOFFM^HLEVAPI0(HLEVIENE)
```

IENE is the internal entry number in the HL EVENT MONITOR file of the monitor to turn off.

7.9. VARIABLE^HLEVAPI

M Code Examples:

```
Direct Variable Declaration
D VARIABLE^HLEVAPI(HLEVIENJ,"CT")

Pass-by-reference Variable Declaration
S VAR("CT")="CT-773",VAR("IEN")="IEN-773"
D VARIABLE^HLEVAPI(HLEVIENJ,.VAR)
```

7.10. ONOFFM^HLEVAPI0

This API sets the status of the Event Monitor system to ACTIVE or INACTIVE. This API can also be used to set the status of a monitor to ACTIVE or INACTIVE.

When the status of the Event Monitor system is changed the specific action taken is:

- ❑ The STATUS-MASTER field (#2) in the HL EVENT PARAMETER file (#776.999) file is changed.

When the status of monitor is changed the specific action taken is:

- ❑ The STATUS field (#2) in the HL EVENT MONITOR file (#776.1) file is changed.

Syntax:

D ONOFFM^HLEVAPI0(file#,ien,status)

The allowable values for the first parameter - file# - are 776.1 if a monitor is being turned on or off, and 776.999 if the Event Monitor system is being turned on or off.

The second parameter holds the IEN of the monitor or Event Monitor system being turned on or off. If a monitor status is being changed, always pass the variable HLEVIENE. If the Event Monitor system status is being changed, always pass the numeral 1.

To turn a monitor or the Event Monitor system on, status should be passed as "A", for active. To turn a monitor or the Event Monitor system off, status should be passed as "I", for inactive.

M Code Examples:

```
D ONOFFM^HLEVAPI0(776,HLEVIENE,"I")
D ONOFFM^HLEVAPI0(776.999,1,"A")
```

7.11. APPSTAT^HLEVAPI1

The APPSTAT API call sets the STATUS-APPLICATION field (#5) in the HL7 MONITOR JOB file (#776). This field has a 10-character length limit. This field is displayed on various reports.

M Code Example:

```
D APPSTAT^HLEVAPI1("BACKUP")
```

The use of these APIs is illustrated in the next section.

8. Writing Monitors

The steps in writing monitors are:

- Write the M code interfacing with the Event Monitor system using the APIs listed above.
- Create the monitor entry in the HL EVENT MONITOR file (#776.1.)

8.1. Writing M Code

Writing a monitor involves writing M code and embedding in the M code calls to the above APIs. Every monitor must call START. When stopping, every monitor must call CHECKOUT or ABORT. Use of all other APIs is optional.

The CHECKIN API stores variables and updates the monitor's "timestamp," the time of last recorded activity. Even though calling CHECKIN is optional, it is good practice to call CHECKIN once every 30 seconds or minute to ensure other processes monitoring the "run" can tell that the job is alive and healthy.

The remainder of this section is devoted to demonstrating the interfacing of M code with the monitoring system.

The following M code does not include any calls to the monitoring system, and is included as a "starting point" for purposes of comparison to the code included later.

```
CTRL  ; Code for example "monitor event"
      N CT,IEN
      S IEN=0,CT=0
      F  S IEN=$O(^HLMA(IEN)) Q:'IEN!($$S^%ZTLOAD)  D
      .  S CT=CT+1
      Q
      ;
```

Below is code with event monitoring API calls embedded to interface the M code with the event monitoring system.

```
CTRL  ; Code for example "monitor event"
      N CT,IEN
      S VAR("CT")="773-CT",VAR("IEN")="773-IEN"
      D START^HLEVAPI(.VAR)
      S IEN=0,CT=0
      F  S IEN=$O(^HLMA(IEN)) Q:'IEN!($$S^%ZTLOAD)  D
      .  S CT=CT+1
      .  I '(CT#1000) D CHECKIN^HLEVAPI
      D CHECKOUT^HLEVAPI
      Q
      ;
```

Note: *There are two formats for declaring the variables to be tracked and recorded by the monitor:*

- *Pass-by-reference declaration.*
- *Direct Parameter declaration*

The pass-by-reference method is illustrated in the M code above. This is normally the preferred format. When using this method, the VAR subscript holds the name of the variable to track, and the value of VAR(variable) - for example, "773-CT" - is a description of the variable's significance.

The "description of a variable's significance" - in our example above, "CT-773" and "773-IEN" - is included in some reports to facilitate the process of reviewing the report.

The direct parameter declaration method, shown below, is also acceptable. Its advantage is ease of use. Its disadvantage is that the variable cannot be "explained" by any description of the variable's significance.

```
CTRL    ; Code for example "monitor event"
  N CT,IEN
  D START^HLEVAPI("CT^IEN")
  S IEN=0,CT=0
      ...
```

Notice the creation of the VAR array. This array is passed by reference into the START^HLEVAPI API. The format for the VAR array is VAR(VARIABLE-NAME)=VARIABLE-EXPLANATION.

```
CTRL  ; Code for example "monitor event"
      N CT,IEN
      S VAR("CT")="773-CT",VAR("IEN")="773-IEN"
      D START^HLEVAPI(.VAR)
      S IEN=0,CT=0
      F  S IEN=$O(^HLMA(IEN)) Q:'IEN!($$S^%ZTLOAD)  D
      .  S CT=CT+1
      .  I '(CT#1000) D CHECKIN^HLEVAPI
      D CHECKOUT^HLEVAPI
      D MAILIT^HLEVAPI ; sends the "basic message"
      Q
      ;
```

The MAILIT call causes a mailman message to be sent to the recipients specified in the mail group entered in the event monitor entry in the HL EVENT MONITOR file (#776.1.) If the application developer sets recipients into the XMY array prior to calling MAILIT, they will be added as recipients..

If the application developer doesn't create recipients in the XMY array, and no mail group exists, the DUZ existent at the time will be used as the recipient.

Here is a sample of the email notification message:

```
Subj: EVENT MONITOR Monitor  [#7363] 05/26/03@08:42  8 lines
From: POSTMASTER  In 'WASTE' basket.   Page 1
-----------------------------------------------------------------------------
The 'EVENT MONITOR' event monitor has completed.

Start time: May 26, 2003@08:42:42   End time: May 26, 2003@08:42:43
Status: FINISHED
```

Every message will include the information shown in the above Mailman message. (This message is referred to as the "basic message" below.) However, application developers can include additional information from the following four areas:

- HL7 MONITOR JOB file (#776) entry's RUN DIARY. (Created by RUNDIARY^HLEVAPI1.)
- HL7 MONITOR JOB file (#776) entry's MESSAGE TEXT. (Created by EVENTEXT^HLEVAPI1.)
- HL7 MONITOR JOB file (#776) entry's VARIABLE VALUE multiple. (Created by the CHECKIN^HLEVAPI and CHECKOUT^HLEVAPI APIs.)
- User-defined global data existent at time of call to MAILIT.

> ***Note:*** *When MAILIT is called, only the "basic message" information is included in the message unless it is specified that one or more of the above four sources of message information should be included.*

Below is the example code, but with modifications showing how to create data in these special fields. The example code also shows how to include this additional information in the Mailman message, and how to control the order the information is added to the message. Please read the M code carefully, noting the new code (in bold letters), and the comments added to explain each step.

```
CTRL   ; Code for example "monitor event"
       N CT,HLEVTXT,IEN
       S VAR("CT")="773-CT",VAR("IEN")="773-IEN"
       D START^HLEVAPI(.VAR)
       S IEN=0,CT=0
       F  S IEN=$O(^HLMA(IEN)) Q:'IEN!($$S^%ZTLOAD)  D
       .  S CT=CT+1
       .  I '(CT#1000) D CHECKIN^HLEVAPI
       ;
       S ^TMP($J,"HLRD",1)="Text here..." ;         Create global data
       D RUNDIARY^HLEVAPI1($NA(^TMP($J,"HLRD"))) ;  Store global data
       ;
       S ^TMP($J,"HLMT",25)="Text here..." ;        Create global data
       D MSGTEXT^HLEVAPI1($NA(^TMP($J,"HLMT"))) ;   Store global data
       ;
       ; Define variable to record every time CHECKIN^HLEVAPI is called.
       ; (Value of variable also stored whenever VARIABLE is called.)
       D VARIABLE^HLEVAPI(HLEVIENJ,"IOM") ; HLEVIENJ is pre-defined
       ;
       S ^TMP($J,"SV-1",1)="Save some text to be stored on-the-fly later"
       ;
       D CHECKOUT^HLEVAPI
       ;
       ; Storage of data in record can occur before or after CHECKOUT...
       S HLEVTXT(1)="RUN DIARY" ; The RUN DIARY to be include 1st in message
       S HLEVTXT(2)="MESSAGE TEXT" ; The MESSAGE TEXT included 2nd...
       S HLEVTXT(3)="VARIABLE VALUE" ; Previous declared variables included
3rd
       S HLEVTXT(4)=$NA(^TMP($J,"SV-1")) ; Data in this global next in msg...
       S HLEVTXT(4)=$NA(^TMP($J,"SV-2")) ; Data in this global next in msg...
       ;
       ; So, 5 sources of global data will be included in the notification
       ; message and they will be included in the message in the order
       ; specified by HLEVTXT(#) subscripting.
       ;
       D MAILIT^HLEVAPI
       ;
       Q
```

```
        ;
```

The notification message sent by MAILIT^HLEVAPI is shown below.

```
Subj: Site# 662 - ZZLJA NEW Monitor  [#8052] 06/27/03@13:56  18 lines
From: POSTMASTER  In 'IN' basket.   Page 1  *New*
------------------------------------------------------------------------------
-
The 'ZZLJA NEW' event monitor has completed.

Start time: Jun 27, 2003@13:55:26   End time: Jun 27, 2003@13:56:20
Status: FINISHED

Run Diary
---------
DECLARE called - ZZLJA NEW

Additional Text
---------------
Text here...

Variable List
-------------
CT[773-CT]=91
IEN[773-IEN]=AC
Save some text to be stored on-the-fly later
```

8.2. Testing M Code

Testing the logic within M code can be difficult when the code contains Event Monitor calls. At times, developers will want to test their code *without* any interaction with the Event Monitor system. This is possible through the use of a developer "backdoor."

To "cut" any interaction with the Event Monitor system, set ^TMP("HLEVFLAG",$J)="STOP".

At the beginning of every Event Monitor API a check is made of ^TMP("HLEVFLAG",$J). If this global node exists and equals "STOP", no further processing occurs.

8.3. Creating the HL EVENT MONITOR File (#776.1) Entry

The two steps in creating a monitor are writing the M code (which was discussed above), and creating an entry in the HL EVENT MONITOR file (#776.1). After creating and testing your monitor, create the monitor entry in the HL EVENT MONITOR file using the *Enter/edit Event Monitors* menu option. The terminal dialogue seen when using this option is shown in the Monitor Creation section starting on page 7. The terminal dialogue contains explanatory text that should be read thoroughly.

9. Server Requests

The Event Monitor software now includes server-based software enabling us to retrieve HL7 data from remote sites, and to remotely request the running of monitors. When a message is sent to S.HLEV-INFORMATION-SERVER@Remote-Site.va.gov, the HLEV-INFORMATION-SERVER option is activated at the remote site and the HLEVSRV routine called. The HLEVSRV routine then evaluates the "action requests" embedded in the email message and returns to the **V***IST***A** HL7 developers the requested data.

Action requests are just lines of the email message, each line corresponding to the type or location of **V***IST***A** HL7 data on the remote site that the requester desires, or the name of the monitor to be run at the remote site.

Example:

Sending a server request is as easy as creating an email, entering into the body of the email message the "action requests", and mailing to S.HLEV-INFORMATION-SERVER@Remote-Site.va.gov. An example message is shown below.

```
Subj: SERVER REQUEST  [#7828] 06/12/03@14:11  1 line
From: HL7Developer,One X  In 'IN' basket.   Page 1
-------------------------------------------------------------------------------
-
UNIT^23^IEN772 ; Data request for all messages linked to entry 23 in file 772.
772^23^0 ; Data request for the zero node of entry 23 in file 772.
QUERY^HLMA(23)^HLMA(23, ; Data request for $Q loop through entry 23 in file
773.
MONITOR^EVENT MONITOR ; Action request to run the “Event Monitor” monitor.
```

Note: *The action requests in the message text above are explained later in this document.*

If the email message above is sent to "S.HLEV-INFORMATION-SERVER@*Remote-Site.va.gov*", the action requests in the message will be extracted and acted upon.

9.1. Action request Types

There are four types of action requests, and they are listed and explained in the following table. Each action request type is discussed fully in the following sections.

Action Request Type	What is Sent	What is Returned
Unit	*"Something"* that identifies an entry in file 772. The *"something"* can be one of the following: File 772 IEN File 773 IEN Message ID	All 772 entries associated with the identified (by the sent "something") 772 entry are returned.

File	The following is sent: A specific IEN, or all IENs are requested. The node(s) desired.	The IEN's requested node.
Query	The "strings" required to perform a $QUERY search.	The data discovered by the $QUERY search.
Monitor	The name of the monitor, and optionally the email addresses of recipients of the monitor report	The monitor's report. At times, no report is returned.

9.1.1. Unit Action Request

The Unit action request is used to find functional "units" of messages. An example unit would be a message, and the commit acknowledgement message returned by the remote system. (See the patch documentation for patch HL*1.6*103 for a more detailed discussion of "message units.")

Units are identified by passing any of the 772 IENs in a message unit, any of the 773 IENs in a message unit, or one of the message Ids in one of the messages in the message unit.

The format for a UNIT action request is: UNIT^Identifier^Type

A literal value of "UNIT" is always placed in piece 1. The Identifier is the IEN or Message ID. The type is one of the following values:

- IEN772
- IEN773
- MSGID

All three pieces are mandatory. Here is a sample message unit.

```
Type              772 IEN     773 IEN     Message ID
Message           12532       14315       50212532
Commit Ack        12533       14319       50212532
Application Ack   12537       14322       43715321
Commit Ack        12539       14325       43715321
```

Here are some sample calls using values from the table above.

- UNIT^12532^IEN772 (Finds the 4 unit messages using the file 772 IEN of 12532.)
- UNIT^14322^IEN773 (Finds the 4 unit messages using the file 773 IEN of 14322.)
- UNIT^50212532^MSGID (Finds the 4 unit messages using the message ID.)

Below is a sample Unit request email message.

```
Subj: TEST  [#7830] 06/12/03@14:36  2 lines
From: HL7Developer,One X In 'IN' basket.   Page 1
---------------------------------------------------------------------------
UNIT^20947^IEN772
UNIT^99820984^MSGID
```

Below is a sample of the returned data.

```
Subj: HLEV SERVER REQUEST SAN FRANCISCO [#662]  [#7831] 06/12/03@14:37
From: POSTMASTER  In 'IN' basket.   Page 1
-------------------------------------------------------------------------------

-------------------- Msg ID-requested Message Units --------------------
20947: 20947,20948
20984: 20984,20989

[Query log stored in ^XTMP("HLEV SERVER 3030612.14371") at site.]
```

The above lines convey the following information:

- Parent 772 message# 20947 has one child, IEN 20948.
- Parent 772 message# 20984 has one child, IEN 20989.

9.1.2. File-based Action Request

The file-based action request is used to request data from specific files and entries. The format is: File#^IEN^NODE(s)

If the zero node of record 25312 in file 773 is desired, the action request would be "773^25312^0".

Multiple nodes can be requested. If the "P" node is desired in addition to the zero node, the above example would be changed to "773^25312^0^P".

> ***Note:*** Piece two may be left blank if all entries are desired. However, when this is done, there is a limit of 1000 entries that will be returned.

> ***Note:*** Multiple nodes are included in the action request starting at up-arrow delimited piece three. The nodes are separated by up-arrows. (In other words, piece three and beyond is reserved for node designation.)

9.1.3. $QUERY Action Request

The $QUERY action request makes use of the M $QUERY function to traverse a data tree and return all data between a starting and ending point. The $QUERY function returns all elements in a data structure no matter the subscripting existent.

> ***Note:** $ORDER returns the next element in a data structure at the same subscript level. $QUERY returns the next element in a data structure, irregardless of the subscript level. Here's some sample data that will be used to illustrate the difference between these functions.*

```
^TMP($J,"D",0)=""
^TMP($J,"D",1,1)=""
^TMP($J,"D",2)=""
```

> *$QUERY of ^TMP($J,"D",0) returns ^TMP($J,"D",1,0). $ORDER of the same node returns 2 (from the ^TMP($J,"D",2 node).*

> *Contact the* **V**ISTA *HL7 Development Team (HL7DevelopmentTeam@med.va.gov) for additional help with $QUERY.*

The syntax for the QUERY action request is: QUERY^Beginning^Stop-value^Number^Format

The explanation for the fields in the QUERY action request is included in the following table.

Field	Example	Mandatory	Explanation
QUERY	QUERY	Yes	The string literal value of "QUERY" must always be in piece 1.
Beginning	^HLMA(25312)	Yes	This is the beginning value used during the $QUERY loop.
Stop-value	^HLMA(25312,	Yes	After every $QUERY, the returned global reference is compared to the "Stop-value", and looping stops if this value is not contained in the $QUERY-found global reference.
Number	25	No	The maximum number of global nodes returned in report.
Format	^HLMA(#,#)	No	The "Format" field filters the global references returned in the report. After every global reference is found by $QUERY, the returned value must pattern match this string to be included.

Example QUERY Action Request in a Mailman message:

```
Subj: SERVER REQUEST  [#7978] 06/19/03@12:51  1 line
From: HL7Developer, One X  In 'IN' basket.   Page 1
---------------------------------------------------------------------------
QUERY^HLMA(20728)^HLMA(20728,
```

Example Returned Report message:

```
Subj: HLEV SERVER REQUEST SAN FRANCISCO [#662]  [#7979] 06/19/03@12:51
From: POSTMASTER  In 'IN' basket.   Page 1
---------------------------------------------------------------------------

Data Requests
--------------------------------------------------------------------------
^HLMA(20728)      ^HLMA(20728,

$QUERY Data
--------------------------------------------------------------------------
^HLMA(20728,0)=20940^5015589^I^^^20727^213^^^20727^^^1
^HLMA(20728,"MSH",0)=^^1^1^3030527^
^HLMA(20728,"MSH",1,0)=MSH^~|\&^ZZRH ORU2^^ZZRH ORU^^20030527145418-0700^^
                       ACK^5015589^T^2.3.1
^HLMA(20728,"P")=3^3030527.145429
^HLMA(20728,"S")=3030527.145429

[Query log stored in ^XTMP("HLEV SERVER 3030619.125141") at site.]
```

9.2. Security

There are two types of action requests, which are listed below with comments about security provisions for each type.

9.2.1. Monitor Requests

When a server-based *monitor* request is received by a site, a check is made of the monitor's REMOTE REQUESTABLE field. If it is not set to YES, the request is ignored. If it is set to YES, the monitor is run.

> ***Note:*** *Sites must carefully review their monitors to ensure that this field is set correctly. If the monitor returns any report or data to the remote requester, the contents of the report must be thoroughly understood. If any sensitive patient information is included, this field must never be set to YES. As a general rule of thumb, set the REMOTE REQUESTABLE field to NO.*

9.2.2. Data Requests

When server-based *data* requests are sent to remote sites, the data requested is always returned to the VHA OI SDD HL7 System Monitoring mail group on Outlook. (The alias for this group is HL7SystemMonitoring@med.va.gov.) Unless a site-supplied license number (see next section) is included in the server request message, messages with data will only go to the VHA OI SDD HL7 System Monitoring mail group.

9.2.3. Request Licenses

Support personnel who are not **V*IST*A** HL7 developers, and who are not members of the **V*IST*A** HL7 Development team's VHA OI SDD HL7 System Monitoring mail group, will sometimes have need for server-based information from sites. These non- **V*IST*A** HL7 developer support personnel can also receive data request messages if they secure a license from the site.

The "Grant remote request license [HLEV GRANT REMOTE LICENSE]" menu option is used by the site to generate licenses for remote support personnel wishing to receive site data via remote server request.

The terminal dialogue seen when using this option is shown next. Please read the explanation included in the terminal dialogue.

```
Grant License to Remote Requesters
===========================================================================
Mailman server requests can be sent to your site requesting HL7 data be
returned to the VistA HL7 team.  These requests are normally only sent to the
VistA HL7 team.  However, from time to time, support personnel will have
legitimate need to retrieve critical VistA HL7 data.  In order to receive
return data, anyone not on the VistA HL7 team needs a license.  This option
will generate a license that must be communicated to those (not on the VistA
HL7 team) requesting remote query rights.

Note:  Notification of every remote server request is automatically sent to
       the VistA HL7 team.  And, this includes the messages sent remotely
       to non-VistA HL7 recipients (using the license you are about to grant.)

Press RETURN to continue, or '^' to exit...  <return>
```

(Terminal dialogue continued on next page.)

```
                       Grant License to Remote Requesters
```

```
===========================================================================

No current license exists...

     Select one of the following:

          1         Create new license
          2         Exit

Select ACTION: Create new license// <return>

License 'Z)f-DCe' will be used after you enter cutoff date...

Defaulting 'NOW + 7 days' below...

Enter CUTOFF DATE:  Jul 09, 2003@10:16// <return>  (JUL 09, 2003@10:16)

  ---------------- Current License - Z)f-DCe [07/09@10:16] ----------------
                            No current users exist!

     Select one of the following:

          1         Change cutoff date/time
          2         Add requesters
          3         Create new license (and cancel old license)
          4         Cancel current license
          5         Exit

Select ACTION: Add requesters// <return>

Enter the complete email address of recipients.  (Enter the address of an
existing user and they will be removed.)

Enter REMOTE ADDRESS: FIRSTNAME.LASTNAME@MED.VA.GOV<return>  added...
Enter REMOTE ADDRESS: <return>

  ---------------- Current License - Z)f-DCe [07/09@10:16] ----------------
                        ----- Licensed Requesters ------
                          FIRSTNAME.LASTNAME@MED.VA.GOV

     Select one of the following:

          1         Change cutoff date/time
          2         Add requesters
          3         Create new license (and cancel old license)
          4         Cancel current license
          5         Exit

Select ACTION: Exit// <return>
```

The license that is generated by site must be conveyed in some secure way - usually by telephone call - to support personnel who will be using the license.

When generating a license, the site must also set a termination date for the license. The default value is seven days.

9.3. Installation

Please refer to the patch text contained in patch HL*1.6*109 for installation instructions.

10. Glossary

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.
ACKNOWLEDGMENT – ACCEPT LEVEL	The receiving system commits the message to safe storage in a manner that releases the sending system from any obligation to resend the message. A response is returned to the initiator indicating successful receipt and secure storage of the information.
ACKNOWLEDGMENT - APPLICATION LEVEL	The appropriate application on the receiving system receives the transaction and processes it successfully. The receiving system returns an application dependent response to the initiator.
ACL	**A**ccess **C**ontrol **L**ist
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.
APPLICATION LAYER	Layer 7 of the OSI Model. Responsible for information transfer between two network applications. This involves such functions as security checks, identification of the two participants, availability checks, negotiating exchange mechanisms and most importantly initiating the exchanges themselves. See OSI Model.
CALLABLE ENTRY POINT	Authorized programmer call that may be used in any **V***IST***A** application package. The DBA maintains the list of DBIC-approved entry points.
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 package. Typically, these IAs are created by the requesting package based on their independent examination of the custodian package’s features. For the IA to be approved, the custodian grants permission to other **V***IST***A** packages 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.
COTS	**C**ommercial **O**ff-**T**he-**S**helf
CROSS REFERENCE	Cross-reference—There are several types of cross-references available. Most generally, a VA FileMan cross-reference specifies that some action is 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	A representation of facts, concepts, or instructions in a formalized manner for communication, interpretation, or processing by humans or by automatic means. The information you enter for the computer to store and retrieve. Characters that are stored in the computer system as the values of local or global variables. VA FileMan fields hold data values for file entries.
DATA DICTIONARY (DD)	The Data Dictionary is a global containing a description of what kind of data is stored in the global corresponding to a particular file. The data is used internally by VA FileMan for interpreting and processing files. A **D**ata **D**ictionary contains the definitions of a file's elements (fields or data attributes); relationships to other files; and structure or design. Users generally review the definitions of a file's elements or data attributes; programmers review the definitions of a file's internal structure.
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.
DBA	**D**ata**b**ase **A**dministrator, oversees package development with respect to **V***IST***A** Standards and Conventions (SAC) such as namespacing. Also, this term refers to the **D**ata**b**ase **Administration** function and staff.
DBIA	**D**ata**b**ase **I**ntegration **A**greement, a formal understanding between two or more **V***IST***A** packages, which describes how data is shared or how packages interact. The DBA maintains a list of DBIAs.
DCL	**D**igital **C**ommand **L**anguage
DEFAULT	Response the computer considers the most probable answer to the prompt being given. It is identified by double slash marks (//) immediately following it. This allows you the option of accepting the default answer or entering your own answer. To accept the default you simply press the Enter (or Return) key. To change the default answer, type in your response.
DELIMITER	Special character used to separate a field, record or string. VA FileMan uses the ^ character as the delimiter within strings.
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.
DICOM	Digital Imaging and Communications in Medicine. Draft standard in development by ACR/NEMA for exchange of radiological images. Version 3 of DICOM defines image data as well as patient, study and visit information necessary to provide the context for the images. This version incorporates an object-oriented data model and adds support for ISO Standard communications.
DIRECT MODE UTLITY	A programmer call that is made when working in direct programmer mode. A direct mode utility is entered at the MUMPS prompt (e.g., >D ^XUP). Calls that are documented as direct mode utilities *cannot* be used in application package code.
DSM	**D**igital **S**tandard **M**UMPS

ELECTRONIC SIGNATURE CODE	Secret password that some users may need to establish in order to sign documents via the computer.
ENTRY	VA FileMan record. It is uniquely identified by an internal entry number (the .001 field) in a file.
EXTRINSIC FUNCTION	Extrinsic function is an expression that accepts parameters as input and returns a value as output that can be directly assigned.
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)	The **V***IST***A**'s Database Management System (DBMS). The central component of the Kernel that defines the way standard **V***IST***A** files are structured and manipulated.
FORM	See ScreenMan Form.
FORUM	The central E-mail system within **V***IST***A**. It is used by developers to communicate at a national level about programming and other issues. FORUM is located at the CIO Field Office - Washington, DC (162-2).
FREE TEXT	A DATA TYPE that can contain any printable characters.
GLOBAL VARIABLE	Variable that is stored on disk (M usage).
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 package 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	Health Level Seven (HL7) is an application protocol for electronic data exchange in health care environments. The HL7 protocol is a collection of standard formats which specify the implementation of interfaces between computer applications from different vendors. This communication protocol allows healthcare institutions to exchange key sets of data amount different application systems. Flexibility is built into the protocol to allow compatibility for specialized data sets that have facility-specific needs.
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.
IEN	**i**nternal **e**ntry **n**umber
INPUT TEMPLATE	A pre-defined list of fields that together comprise an editing session.

INTEGRATION AGREEMENTS (IA) (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 SD&D web page.
IRM	Information Resource Management. A service at VA medical centers responsible for computer management and system security.
KERNEL	Set of **V***IST***A** software routines that function as an intermediary between the host operating system and the **V***IST***A** application packages such as Laboratory, Pharmacy, IFCAP, etc. The Kernel provides a standard and consistent user and programmer interface between application packages and the underlying M implementation.
LINK	Non-specific term referring to ways in which files may be related (via pointer links). Files have links into other files.
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.
MENU TEXT	The descriptive words that appear when a list of option choices is displayed. Specifically, the Menu Text field of the OPTION file (#19). For example, User’s Toolbox is the menu text of the XUSERTOOLS option. The option’s synonym is TBOX.
MESSAGE	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.
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.
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.
NT	**N**ew **T**echnology
OPTION	An entry in the OPTION file. 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.

OPTION NAME	Name field in the OPTION file (e.g., XUMAINT for the option that has the menu text “Menu Management”). Options are namespaced according to **V***IST***A** conventions monitored by the DBA.
PACKAGE	The set of programs, files, documentation, help prompts, and installation procedures required for a given software application. For example, Laboratory, Pharmacy, and PIMS are packages. A **V***IST***A** software environment composed of elements specified via the PACKAGE file (#9.4). Elements include files and associated templates, namespaced routines, and namespaced file entries from the OPTION, HELP FRAME, BULLETIN, and FUNCTION files. As public domain software, packages may be requested through the Freedom of Information Act (FOIA).
POINTER	The address at which a data value is stored in computer memory. A relationship between two VA FileMan files, a pointer is a file entry that references another file (forward or backward). Pointers can be an efficient means for applications to access data by referring to the storage location at which the data exists.
PRIMARY KEY	A Data Base Management System construct, where one or more fields uniquely define a record (entry) in a file (table). The fields are required to be populated for every record on the file, and are unique, in combination, for every record on the file.
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.
RECORD	Set of related data treated as a unit. An entry in a VA FileMan file constitutes a record. A collection of data items that refer to a specific entity (e.g., in a name-address-phone number file, each record would contain a collection of data relating to one person).
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.
SAC	**S**tandards **a**nd **C**onventions. Through a process of verification, **V***IST***A** packages are reviewed with respect to SAC guidelines as set forth by the Standards and Conventions Committee (SACC). Package documentation is similarly reviewed in terms of standards set by the Documentation Standards and Conventions Committee (DSCC).

SACC	**V***IST***A**'s **S**tandards **a**nd **C**onventions **C**ommittee. This Committee is responsible for maintaining the SAC.
SCREEN EDITOR	VA FileMan's Screen-oriented text editor. It can be used to enter data into any WORD-PROCESSING field using full-screen editing instead of line-by-line editing.
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.
SCREEN-ORIENTED	A computer interface in which you see many lines of data at a time and in which you can move your cursor around the display screen using screen navigation commands. Compare to Scrolling Mode.
SCROLLING MODE	The presentation of the interactive dialogue one line at a time. Compare to Screen-oriented.
SEGMENT	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.
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).
STANDARD FORM OF A NAME (ALSO CALLED STANDARD FORMAT)	The definition of standard form (or standard format) is a person's name entirely in uppercase letters, containing no Arabic numerals (i.e., 1, 2, 3, etc.). The Family Name (last name) portion of a standard name appears to the left of the comma and contains no spaces and no punctuation except hyphens (-). The Given Name (first name), Middle Name, and Suffix (the portion to the right of the comma) contain no punctuation except for hyphens and spaces. NMI and NMN are not used for the Middle Name. The standard form of a name is: Family_name,Given_name<space>Middle_name<space>Suffix
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 package 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** packages 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).
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	The Department of **V**eterans **A**ffairs, formerly called the **V**eterans **A**dministration.
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.
VARIABLE	Character, or group of characters, that refer 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).
VISN	**V**eterans **I**ntegrated **S**ervice **N**etwork
V*IST***A**	**V**eterans **H**ealth **I**nformation **S**ystems and **T**echnology **A**rchitecture (**V***IST***A**) (formerly the Decentralized Hospital Computer Program [DHCP]) of the Veterans Health Administration (VHA), Department of Veterans Affairs (VA). **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, which undergo a verification process to ensure conformity with namespacing and other **V***IST***A** standards and conventions.
VMS	**V**irtual **M**emory **S**ystem
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.