VistA HL7- Optimized (HLO)

Supplement to VistA Health Level Seven (HL7) Version 1.6

TECHNICAL MANUAL
(System Manager/Developer Guide)
Patch HL*1.6*131

May 2006
Document Revision 1.8

Department of Veterans Affairs
VHA Office of Information - System Design and Development

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
03/2005	Draft V 1.0	Document the re-design of the HL7 engine. HLO is intended to improve data throughput, ease application development, and minimize maintenance overhead.
05/2005		Reviewed and updated by technical writer, Brian Lynch.
06/2005	Draft V 1.1	Updated by Daou Systems, Inc.
08/2005	Draft V 1.2	Updated by Daou Systems, Inc.
08/22/2005	Draft V1.3	Updated by Jim Moore
08/26/2005	Draft V1.4	Updated by Daou Systems, Inc.
09/02/2005	Draft V1.5	Note added by Jim Moore – status of the INTEFACE ENGINE currently should show NOT OPERATIONAL
9/6/2005	Draft V1.5	Warnings added by Jim Moore: 1) When adding a new listener for HLO, if an already-existing entry in the HL LOGICAL LINK File (#870) is used for an existing listener, don't alter certain fields. 2) Sites running under VMS or Cache should use a multi-listener.
9/16/2005	Draft V1.6	Updates by Daou: 1) Synchronized Tech Manual install section with Install Manual. 2) Trouble-Shooting Appendix (same as chapter 6 in Install Manual) 3) Reviewed for flow and content
12/6/2005	Draft V1.7	Updates by Jim Moore for patch HL*1.6*130
5/1/2006	Draft V1.8	Added supports for data types in patch HL*1.6*131. (Jim Moore)

Patch Revisions

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

Comments

Technical Services welcomes your comments on this manual. Please send your comments to:

tsopenvista@med.va.gov

Table Of Contents

Introduction

Welcome to the *VistA HLO Technical Manual (System Manager/Developer Guide)*. The goal of this manual is to provide VistA developers and system managers with all of the information they need to build VistA HL7 interfaces and manage the VistA HLO software package.

The Technical Manual is organized for the following user types:

All Users	Chapters 1 and 2
Installer	Chapters 3 and 4 plus Installation Manual
Manager/Maintenance	Chapter 5
Developer	Chapter 6 and Appendices

Documentation Conventions

Screen Dialog

This manual presents snapshots of computer dialogue or other on-line displays in a non-proportional font. User responses to on-line prompts are highlighted in bold. Pressing the return key is illustrated as <RET>, and is only shown when the user doesn't type anything at the prompt besides pressing Return key. For example, the following indicates that the user should enter two question marks followed by <RET> when prompted:

```
Select Primary Menu option: ??
Following menu options are available……
Select Primary Menu option: <RET>
```

Software Processes and Code

Processes are indicated with single quotes.
Code is indicated with double quotes.

API Parameters

Each API that is documented for developer use also includes input and output parameters. If a parameter isn't specifically documented as being passed by reference, then it is to be passed by value.

1.0 HL7 (Health Level Seven)

1.1 Brief Overview

HL7 is an ANSI messaging transaction standard for healthcare. It is the main strategy used by a variety of healthcare providers and applications vendors to achieve Enterprise Application Integration (EAI) between disparate clinical applications.

From the HL7 standard:

> 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 among different application systems. Flexibility is built into the protocol to allow compatibility for specialized data sets that have facility-specific needs.[1]

Because many vendors support the HL7 standard, it allows healthcare institutions to exchange key sets of data from very different application systems. HL7 defines:

- The data to be exchanged
- The timing of the interchange
- The communication of errors to the sending/receiving application

HL7 *messages* are the unit of data exchange between systems. HL7 message formats, although standardized, are generic in nature, and must be configured (negotiated) to meet the specific needs of the two applications involved. As such, HL7 interfaces between applications are not "plug and play", and must be formally negotiated between the two applications.

The HL7 standard defines standard messages for the following healthcare application areas:

- Patient Administration (e.g., admission, discharge, transfer)
- Order entry
- General Queries
- Financial management (e.g., charges, payments, adjustments, and insurance)
- Observation Reporting
- Master Files
- Medical Records/Information Management
- Scheduling
- Patient Referral
- Patient Care

[1] Health Level Seven, *Health Level Seven, Version 2.4.* © 2000, p. D-17.

1.2 HL7 Standard

An HL7 message is the atomic unit for transferring data between systems in the HL7 standard. Each message has a header segment composed of a number of fields, including fields containing the message type (for HL7 versions 2.2 and above) and event type. These are each three-character codes, defined by the HL7 standard. The type of a transaction is defined by the message type/event type pair (again for HL7 versions 2.2 and above). Rules for constructing message headers and messages are provided in the *Control* chapter of the HL7 standard.

An HL7 message consists of one or more HL7 *segments*. A segment is similar to a record in a file. Each segment consists of one or more fields separated by a special character called the *field separator*. The field separator character is defined in the Message Header (MSH) segment of an HL7 message. The MSH segment is always the first segment in every HL7 message (except for batch HL7 messages, which begin with the Batch Header (BHS) segment).

In addition to the field separator character, four other special characters, called *encoding characters*, are used as delimiters. The encoding characters and the field separator are defined in the MSH or BHS segment. Each encoding character must be unique and serves a specific purpose. None of the encoding characters can be the same as the field separator character. The four encoding characters are defined as:

- *Component separator*. Some data fields can be divided into multiple components. The component separator character separates adjacent components within a data field.
- *Repetition separator*. Some data fields can be repeated multiple times in a segment. The repetition separator character separates multiple occurrences of a field.
- *Escape character*. Data fields defined as text or formatted text can include escape sequences. The escape character is used to start and end escape sequences within the actual text.
- *Sub-component separator*. Some data fields can be divided into components, and each component can be further divided into sub-components. The sub-component separator character separates adjacent sub-components within a component of a field.

The following is an example of an HL7 message:

```
MSH|^~\&|ORDER|BOSTON VAMC|RESULTS|500|19900314130405|ORU^R01|523123|D|2.3|
PID||7777790^2^M10||HL7Patient^One|||||||123456789|
OBR||2930423.08^1^L||199304230800|||||||DERMATOLOGY|
OBX|CE|10040|OV|1^0^0^0^0|
OBX|CE|11041|PR|
OBX|CE|216.6|P|
OBX|ST|VW^WEIGHT^L||120|KG
OBX|ST|VB^BLOOD PRESSURE^L||120/80|MM HG
OBX|ST|VT^TEMPERATURE^L||99|C
OBX|ST|VP^PULSE^L||75|/MIN
```

The following is an analysis of the message:

- The first line of the message is the message header (MSH) segment.
- The message type (from the MSH segment) is Observation Result/Unsolicited (ORU).
- The event type (from the MSH segment) is an unsolicited transmission of an observation message (R01).
- The second line of the message is the second segment, Patient Identification (PID).
- The third line of the message is the third segment, an Observation Request (OBR).
- The subsequent lines of the message are multiple Observation/Results (OBX) segments.

1.2.1 Unsolicited Updates

The primary characteristic of an unsolicited message is that it is broadcast by an application to one or more recipients, without being solicited by those recipients. Unsolicited updates are typically sent by an application when an *event point* occurs in that application:

> The Standard is written from the assumption that an event in the real world of healthcare creates the need for data to flow among systems. The real-world event is called the **trigger event**. For example, when a patient is admitted, a trigger event might occur that will send data about that patient to a number of other systems. The trigger event might occur when an observation (e.g., a CBC result) for a patient is available, causing information from the observation to be sent to a number of other systems. When the transfer of information is initiated by the application system that deals with the triggering event, the transaction is termed an **unsolicited update**.[2]

Healthcare information systems that support HL7 typically provide a mechanism for applications to subscribe to event points that may be of interest. Each unsolicited update, representing a clinical event, is distributed to every "interested" application that subscribes to the event. For example, when a patient is registered by VistA, an ADT A04 message is generated and delivered to all subscriber applications interested in patient registrations.

Unsolicited updates are also used to aggregate data from VistA systems to Austin and other centralized databases.

1.2.2 Acknowledgements

When a message is sent to a system, return of an *acknowledgement* (also referred to as an “ACK”) message from the receiving system may be required as part of the defined transaction:

- An *accept acknowledgement* (also called a commit acknowledgement) confirms that the receiving system has received the message that was sent.
- An *application acknowledgement* confirms that the receiving application was able to process the sender’s message.

The type of acknowledgement returned for any given message depends on the negotiated interface between the systems.

[2] Health Level Seven, *Health Level Seven, Version 2.3.1*, ©1999, p. 2-1.

1.2.3 Queries

The primary characteristic of a query message is that the system that needs the information connects as a client to the server system that has the needed information. For example, to query the Master Patient Index (MPI) for demographic data and a list of treating facilities for a patient, a VQQ (virtual table query) message is sent from the VA facility to the MPI.

1.3 History of the VistA HL7 Package

VistA HL7 is an *implementation* of an HL7 interface engine. It is an M-based software product that assists M-based VistA applications by providing a means for those applications to send and receive HL7 messages.

VistA HL7 does not provide tools to map VistA data directly to HL7 messages. It does provide a repository of supported HL7 transactions, connectivity between systems, and guaranteed delivery of messages using supported Lower Layer Protocols (LLPs). It supports point-to-point interfaces, publish and subscribe, and content-based dynamic routing.

M-based VistA applications can use VistA HL7 to interface with any other system that also supports standard HL7 messaging, including many standalone medical devices, non-M-based applications on other systems, and VistA M-based applications running on a different VA facility's systems. As such, VistA HL7 acts as an Enterprise Application Integration (EAI) solution for VistA applications.

1.3.1 HL7 Standard Support

VistA HL7 supports message transactions for each of the following versions of the HL7 standard:

- 2.1 (HL7 standard publication date: 6/1990)
- 2.2 (HL7 standard publication date: 12/1994)
- 2.3 (HL7 standard publication date: 4/1997)
- 2.3.1 (HL7 standard publication date: 5/1999)
- 2.4 (HL7 standard publication date: 11/2000)

1.3.2 Evolution of VistA HL7

The first released version of VistA HL7 was 1.5, and supported simple point-to-point HL7 transactions between VistA and a local COTS system using Hybrid Lower Layer Protocol (HLLP), and to other VA facilities using VA MailMan. The initial release of version 1.6 added the ability to "broadcast" a message to multiple recipients, and support for the X3.28 LLP. A continuing increase in the demand for additional messaging services has resulted in enhancements to HL 1.6, released through patches, including more complex message routing (dynamic addressing), and messaging using Minimal Lower Layer Protocol (MLLP) over TCP.

The current release of HLO is a substantial redesign of HL 1.6. It is designed to be able to operate alongside HL 1.6, and provides significantly higher operating efficiency and the ability to operate with a substantially decreased system load. In addition, HLO has some enhancements in the way it provides support for building and parsing of HL7 messages.

Since the early 1990's, the VA has maintained a corporate membership with the Health Level Seven standards organization, and has been an active participant in the evolution of the HL7 standard. Application developers for VistA's Radiology, Pharmacy, Lab, PIMS, and other packages have aggressively enhanced those packages to support HL7-defined event points. Examples include "patient registration," "admit a patient," "lab results available," and "place an order." When one of these event points occurs, the VistA application now generates a corresponding HL7 message, which VistA HL7 distributes to all interested subscribers.

2.0 HL Optimized (HLO)

2.1 Overview and Background

- HLO is an optimized version of HL 1.6.
  - All but one file used by HLO is new. The only HL 1.6 file that is used by HLO is the HL LOGICAL LINK File (#870).
  - There are more than 30 new routines used by HLO, as opposed to approximately 200 routines in the existing HL 1.6.
  - The new namespace is HLO*.

- HLO engine will co-exist with the existing HL 1.6 engine.
  - Existing HL7 applications written for the HL 1.6 engine (HL 1.6 applications) can continue to run without modification.
  - The conversion of existing HL 1.6 applications to HLO applications can be fairly straightforward, though testing of the application is essential.
  - There is virtually no shared code between the HL 1.6 and HLO engines. Thus, modifications and enhancements of applications written for one HL7 engine can be made without extensive testing of the application being required on the other HL7 engine.

- Goals of HLO:
  - Improve the messaging throughput.
  - Make it easier for application developers to use.
  - Make the software more robust, less susceptible to bugs, and require minimal maintenance overhead.

2.2 Design Highlights

- In HLO, the HLO MESSAGE BODY File (#777) corresponds to the HL7 MESSAGE TEXT File (#772), the HLO MESSAGES File (#778) corresponds to the HL7 MESSAGE ADMINISTRATION File (#773), and the HLO SUBSCRIPTION REGISTRY File (#779.4) corresponds to the HL7 SUBSCRIPTION REGISTRY File (#774). There are fewer fields and cross-references in HLO, compared to HL7.

- The PROTOCOL File (#101) is no longer used for HLO. In place of protocols, HLO requires that receiving and sending applications be defined in the new HLO APPLICATION REGISTRY File (#779.2).

- HL Logical Links are still used. HLO requires a different port than the original HL 1.6 implementation (5000 production, 5025 test). Port 5001 will be the standard default port for HLO on production systems and port 5026 for main test systems.

WARNING: All VistA sites must use Port #5001 for the HLO Standard Listener for production accounts. For test accounts Port #5026 must be used.

- HLO consists of an assortment of co-operating processes (client links, server links, in-filers, purge processes, etc.). These processes operate under a common framework, the HLO Process Manager, which uses the new HLO PROCESS REGISTRY File (#779.3).

- Processes utilize a pooling method which allows for a greater number of queues (incoming and outgoing) to run concurrently. This allows fewer processes to service more queues and do more work.

- Processes running under the process manager are dynamic, starting and stopping in response to changing workload.

- Concurrency is maximized by allowing applications to designate private queues for their messages, allowing many more queues to be processed concurrently. This pertains to the outgoing message queues & the incoming applications queues.

- Domain becomes the standard method for identifying sending and receiving facilities.

- There are five new globals in HLO. They are:
  - ^HLA - HLO MESSAGE BODY File (#777).
  - ^HLB - HLO MESSAGES File (#778).
  - ^HLC - System counters (i.e., for assigning message IENs).
  - ^HLD - Static HLO parameter files, including system parameters, application registry, process registry, and subscription registry.
  - ^HLTMP - System scratch space.

- Commit acknowledgements will not be stored as separate records. Instead, the MSA segment will be stored with the original message. Application acknowledgements will be stored as separate records.

- HLO supports only TCP messaging.

- HLO does not support:
  - Synchronous mode (direct connect) or original mode acknowledgements.
  - Sequence number protocol.

2.3 HLO Developer Perspective

- The HLO developer APIs are new and include:
  - Message parsing.
  - Improved support for batch messages.
  - Improved support for building messages.
  - Automated escape sequence processing.
  - Support for acknowledgement messages.

- Internal data structures are encapsulated and accessed via a set of programmer interfaces, protecting them from corruption. The many variables of the HL 1.6 implementation have been replaced in HLO by arrays with more meaningful subscripts, allowing parameters which are applicable to different parts of the HLO system to be grouped together.

2.4 HLO System Manager Perspective

The user interface has been completely re-designed and will have its own menu. Tools include:

- A system monitor for starting and stopping the HLO engine, displaying status information, and viewing message queues and HLO processes.
- A message viewer for viewing messages, displaying errors, and message searching.
- A developer front end for the application registry.

3.0 HLO Installation and Configuration

Patch HL*1.6*126 contains all of the components needed to support HLO and was created using the Kernel Installation and Distribution System (KIDS). Please review the KIDS documentation patch description, and familiarize yourself with KIDS prior to installing this package.

Note: The following patches are required before installation of HL*1.6*126:

- XU*8*388
- HL*1.6*84
- HL*1.6*118

ALWAYS back up your system prior to loading any software.

To correctly install HLO and configure it for proper development and usage:

VistA Steps:

1) Install the HLO Software Patch

2) Define the Server Logical Link

3) Update the HLO SYSTEM PARAMETERS File (#779.1)

4) Update the HLO PROCESS REGISTRY File (#779.3)

5) Schedule the HLO COUNT RECORDS Option

6) Schedule the HLO SYSTEM STARTUP Option

7) Start HLO using the HLO System Monitor.

VMS Step:

8) Create and activate the TCPIP Services for OpenVMS

WARNING: Incoming messages and application acknowledgements are dependent upon the TCPIP Services for OpenVMS being defined and active.

3.1 Install the HLO Software Patch

The HLO package arrives as a standard KIDS Build. It will install the following elements:

1) New fields in the HL LOGICAL LINK File (#870). TCP/IP PORT (OPTIMIZED) field (#400.08) and DNS DOMAIN field (#.08) are added.

2) New files, HLO MESSAGE BODY (#777) and HLO MESSAGES (#778), for holding messages.

3) A new file, HLO SYSTEM PARAMETERS (#779.1), which contains system parameters specific to the installing site.

A Note about System Parameters
The System Parameters are automatically configured as part of the installation. However, if it becomes necessary to modify them, they can be accessed in the HLO SYSTEM PARAMETERS File (#779.1). The key fields are:

- Domain Name – The domain name of your system.
- Station Number – A number which uniquely identifies your site from others.

4) A new file, HLO APPLICATION REGISTRY (#779.2), which contains information for both sending and receiving applications.

5) A new file, HLO PROCESS REGISTRY (#779.3), which contains information on HLO processes. This file will arrive configured and should not be modified except for adding links and activating listeners.

6) A new file, HLO SUBSCRIPTION REGISTRY (#779.4). This file is very similar to file HL7 SUBSCRIPTION REGISTRY (#774), with the exception that it contains subscriptions in format appropriate to the HLO package.

7) New entries for the OPTION File (#19) for monitoring and changing the behavior of the HLO system.

8) A set of routines in the HLO* namespace.

First, load the KIDS distribution and install the HL*1.6*126 package. For more details on the installation of packages, please see the KIDS manual.

Example Install of patch HL*1.6*126

```
Select Kernel Installation & Distribution System  Option: Installation

Select Installation Option: ?

   1     Load a Distribution [XPD LOAD DISTRIBUTION]
   2     Verify Checksums in Transport Global [XPD PRINT CHECKSUM]
   3     Print Transport Global [XPD PRINT INSTALL]
   4     Compare Transport Global to Current System [XPD COMPARE TO SYSTEM]
   5     Backup a Transport Global [XPD BACKUP]
   6     Install Package(s) [XPD INSTALL BUILD]
         Restart Install of Package(s) [XPD RESTART INSTALL]
         Unload a Distribution [XPD UNLOAD DISTRIBUTION]

Select Installation Option: 6  Install Package(s)
Select INSTALL NAME: HL*1.6*126       Loaded from Distribution  9/15/05@13:34:50
     => HL*1.6*126 (SEP14)

This Distribution was loaded on Sep 15, 2005@13:34:50 with header of
   HL*1.6*126 (SEP14)
   It consisted of the following Install(s):
     HL*1.6*126
Checking Install for Package HL*1.6*126

Install Questions for HL*1.6*126

Incoming Files:
```

```
   777       HLO MESSAGE BODY

   778       HLO MESSAGES

   779.1     HLO SYSTEM PARAMETERS

   779.2     HLO APPLICATION REGISTRY

   779.3     HLO PROCESS REGISTRY  (including data)

   779.4     HLO SUBSCRIPTION REGISTRY

   870       HL LOGICAL LINK
Note:  You already have the 'HL LOGICAL LINK' File.

Want KIDS to Rebuild Menu Trees Upon Completion of Install? YES// NO


Want KIDS to INHIBIT LOGONs during the install? YES// NO
Want to DISABLE Scheduled Options, Menu Options, and Protocols? YES// YES

Enter options you wish to mark as 'Out Of Order': HL MAIN MENU       HL7 Main Menu

Enter options you wish to mark as 'Out Of Order':<RET>

Enter protocols you wish to mark as 'Out Of Order':<RET>

Delay Install (Minutes):  (0-60): 0//:<RET>


Enter the Device you want to print the Install messages.
You can queue the install by enter a 'Q' at the device prompt.
Enter a '^' to abort the install.

DEVICE: HOME// <RET>
-------------------------------------------------------------------------------

 Install Started for HL*1.6*126 :
               Sep 15, 2005@13:35:46

Build Distribution Date: Sep 14, 2005

 Installing Routines:
               Sep 15, 2005@13:35:47

 Installing Data Dictionaries:
               Sep 15, 2005@13:35:48

 Installing Data:
               Sep 15, 2005@13:35:48

 Installing PACKAGE COMPONENTS:

 Installing INPUT TEMPLATE

 Installing FORM
```

```
 Installing PROTOCOL
  Located in the HL (HEALTH LEVEL SEVEN) namespace.
  Located in the HL (HEALTH LEVEL SEVEN) namespace.
  Located in the HL (HEALTH LEVEL SEVEN) namespace.
  Located in the HL (HEALTH LEVEL SEVEN) namespace.
  Located in the HL (HEALTH LEVEL SEVEN) namespace.
  Located in the HL (HEALTH LEVEL SEVEN) namespace.
  Located in the HL (HEALTH LEVEL SEVEN) namespace.
  Located in the HL (HEALTH LEVEL SEVEN) namespace.
  Located in the HL (HEALTH LEVEL SEVEN) namespace.
  Located in the HL (HEALTH LEVEL SEVEN) namespace.
  Located in the HL (HEALTH LEVEL SEVEN) namespace.
  Located in the HL (HEALTH LEVEL SEVEN) namespace.
  Located in the HL (HEALTH LEVEL SEVEN) namespace.
  Located in the HL (HEALTH LEVEL SEVEN) namespace.
  Located in the HL (HEALTH LEVEL SEVEN) namespace.
  Located in the HL (HEALTH LEVEL SEVEN) namespace.
  Located in the HL (HEALTH LEVEL SEVEN) namespace.
  Located in the HL (HEALTH LEVEL SEVEN) namespace.
  Located in the HL (HEALTH LEVEL SEVEN) namespace.

 Installing LIST TEMPLATE

 Installing OPTION
              Sep 15, 2005@13:35:49

 Running Post-Install Routine: ^HLOPOST

 Updating Routine file...

 Updating KIDS files...

 HL*1.6*126 Installed.
              Sep 15, 2005@13:35:49
-------------------------------------------------------------------------------
          [-----------------------------------------------------------]
  100%    |             25             50             75              |
Complete  [-----------------------------------------------------------]

 Install Message sent # nnnnnnn

 Install Completed
```

WARNING: As part of HLO configuration, DSM sites should check the settings of all global max string lengths. They should be set to the maximum of 512. This enables HLO to read and process long HL7 segments correctly.

WARNING: Error trap displays of extremely long variables may be limited to 255 characters and may be truncated under certain versions of M.

3.2 Define the Server Logical Link

HLO requires a Server Logical Link for receiving messages. If HLO (either server or client) is using the same link as HL 1.6, the only requirements are to define the TCP/IP PORT (OPTIMIZED) and the DNS DOMAIN fields and to verify that all other elements are properly defined for HLO. If this is not possible, a new link must be created.

The default port number for the HLO Server Logical Link (listener) is 5001 on production systems and 5026 on main test systems. If port number 5001 is already in use by another listener, that listener must be re-assigned to a new port number.

> The HL 1.6 listener and the HLO listener can use the same HL Logical Link entry, but cannot use the same port number. If using an all-ready existing entry for HLO, don't delete or modify any of the existing fields, and the 'old' listener uses them even if the 'new' HLO listener doesn't.

The preferred listener method for running HLO is the TCPIP Services for Open VMS. It is unlikely that more than one listener will be needed. However, HLO is capable of serving several listeners at the same time.

> Multi-Listener vs. Single Listener –
>
> **IRM staffs initially installing HLO**: If your system is a VMS or Cache system, use a multi-listener! If your system is VMS, the multi-listener should run under VMS TCP. If it is not VMS, but is Cache, you should set up a Taskman multi-listener. Only if your system is neither VMS nor Cache should you set up a single listener.
>
> **Application Developers**: Normally, your application should use the site's standard listener. If you must create your own listener (highly discouraged), if only one connection request will be created at a time and the interfacing application requires its own server, then a single listener would be applicable. Otherwise, if there is a possibility of multiple connection requests, then the multi-listener is appropriate.

For more details on setting up listeners, please refer to the next chapter, 'Listeners'.

WARNING: The TaskMan Multi-Listener should NOT be used on systems running Cache under OpenVMS, use a multi-listener running as a VMS TCPIP Service instead. For any system required to use the TaskMan Multi-Listener (such as those running Cache under NT), **patch XU*8.0*388 must be installed first**.

To create or edit a server Logical Link definition, use the *Link Edit* option, on the Interface Developer Options menu:

```
Select Interface Developer Options Option: Link Edit

Select HL LOGICAL LINK NODE: VABAY
```

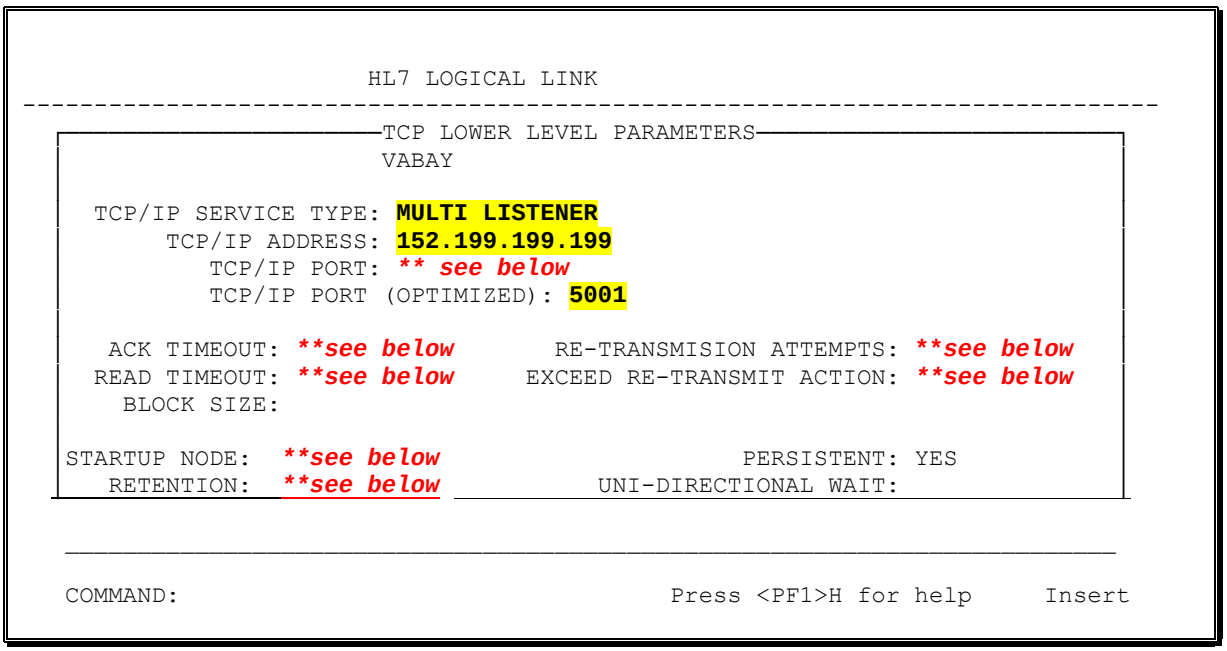
To edit the TCP/IP server level parameters, tab down to the LLP Type field (in the *Link Edit* option) and press **<RET>** to display a form to edit the field's specific to the LLP type of the selected Link:

```
                          HL7 LOGICAL LINK
------------------------------------------------------

                 NODE: VABAY

          INSTITUTION: BAY PINES VAMC

      MAILMAN DOMAIN: BAY-PINES.MED.VA.GOV

            AUTOSTART: **see below

           QUEUE SIZE: **see below

             LLP TYPE: TCP

           DNS DOMAIN: HL7.BAY-PINES.MED.VA.GOV
```

- Production system's domain name should be registered on the VHA DNS Domain server. If not currently registered, sites should do this as soon as possible.
- If the TCP/IP Address is not entered, or if it changes after being entered, it will be resolved automatically using the system's registered domain name via the VHA DNS Domain server.
- If the domain name is not registered on the VHA DNS Domain server, the TCP/IP Address must be defined.

For creating a server logical link, key LLP set-up information includes:

Field	Description
TCP/IP SERVICE TYPE	Set to 'MULTI LISTENER'
TCP/IP ADDRESS	IP Address of the site's server
TCP/IP PORT (OPTIMIZED)	Port to listen on, e.g., 5001 for production systems and 5026 for test systems (make note of the exact port number)



WARNING: Please make sure that the *TCP/IP PORT (OPTIMIZED)* field is used and not the *TCP/IP PORT* field. All VistA sites must use Port #5001 for the HLO Standard Listener for production accounts. For test accounts Port #5026 must be used.

**** WARNING:** If the existing HL Logical Link for the HL7 1.6 listener is reused for HLO, **DO NOT** modify or delete any of the fields marked above (****see below**)! Even though HLO does not use them, HL7 1.6 does.

Please contact the site IRM to obtain the specific domain name and port number to be used by the client side communicating with VistA. For example, at many sites the HL 1.6 multi-listener uses HL7.SITENAME.MED.VA.GOV with port 5000.

Currently, HLO uses the same domain name with port 5001 on production systems and 5026 on test systems.

If a site has more than one test system, they may use additional port numbers.

Start/Stop Link – For HLO processing the server and client HL Logical Link entries DO NOT need to be started or stopped via the HL 1.6 Start/Stop links option as currently done. The links must be defined with the specific HLO fields (DNS Domain and TCP/IP Port (Optimized)) and HLO System started for the link to be available for receiving or transmitting messages. However, if the link must continue to process applications for the existing HL 1.6 messaging engine then the link still needs to be started as before.

Please obtain the specific domain name and port number to be used by the client side communicating with VistA.

For example, at many sites the HL 1.6 multi-listener uses: HL7.SITENAME.MED.VA.GOV with port 5000.

Currently, HLO uses the same domain name with port 5001 on production systems and 5026 on test systems.

If a site has more than one test system, additional port numbers may be used.

- The port number you select must be an available TCP/IP port number. The port number will also be used in the configuration and naming of the TCP/IP service described in the following sections.
- The port numbers recommended in this chapter, 5001 for production, 5026 for test, have been registered for use by VistA HLO. Everything should be done to free port 5001 for use by HLO.
- If the HLO multi-listener is to be used by an application at the national level and you are not using port number 5001, you must register the port number with the DBIA manager on Forum.

For configuring client logical links please refer to Section 6.2 of this document.

3.3 Update the HLO SYSTEM PARAMETERS File (#779.1)

On the HLO System Monitor screen, the Standard Listener is used to monitor the primary server link (or "listener") for HLO. In most instances, this is the only listener that is configured. In order for the Standard Listener status to function properly, the name of the link must be added to the HLO STANDARD LISTENER field in the HLO SYSTEM PARAMETERS File (#779.1). This must be done using FileMan, as shown below.

A Note about System Parameters

The System Parameters are automatically configured as part of the installation. However, if it becomes necessary to modify them, they can be accessed in the HLO SYSTEM PARAMETERS File (#779.1). The key fields are:

- Domain Name – The domain name of your system.
- Station Number – A number which uniquely identifies your site from others.

```
Select OPTION: ENTER OR EDIT FILE ENTRIES

INPUT TO WHAT FILE: //HLO SYSTEM PARAMETERS
EDIT WHICH FIELD: ALL// HLO STANDARD LISTENER
THEN EDIT FIELD: <RET>

Select HLO SYSTEM PARAMETERS DOMAIN NAME: HL7.VAABC.VA.MED.GOV  ←  This is the DNS domain
name for this system.  Also, there will always be only one entry in the HLO System Parameters file and its
IEN is 1.

HLO STANDARD LISTENER: VABAY←  This is the name of the entry in the HL LOGICAL LINK File (#870)
that is the default listener to which most remote applications will send messages, as shown above

Select HLO SYSTEM PARAMETERS DOMAIN NAME: <RET>
Select OPTION: <RET>
```

3.4 Update the HLO PROCESS REGISTRY File (#779.3)

The following fields need to be updated in the HLO PROCESS REGISTRY File (#779.3) for the site's HLO Standard Listener process (for most sites this is the VMS TCP Listener):

- **ACTIVE Field (#.02)** – Set to "YES"
- **DEDICATED LINK Field (#.14)** – Site's listener link (HL Logical Link record defined in section 4.2)

The following table summarizes recommendations for the HLO PROCESS REGISTRY entry to activate as the HLO Standard Listener.

SYSTEM TYPE	HLO PROCESS REGISTRY ENTRY
Cache on OpenVMS DSM on Open VMS	VMS TCP LISTENER
Cache on NT Other M systems	TASKMAN MULTI-LISTENER
COTS interfaces unable to use a multi-listener	SINGLE LISTENER

```
Select HLO PROCESS REGISTRY PROCESS NAME: VMS TCP LISTENER
PROCESS NAME: VMS TCP LISTENER// <RET>
ACTIVE: NO// YES
MINIMUM ACTIVE PROCESSES: 1// <RET>
MAXIMUM ACTIVE PROCESSES: 1// <RET>
SCHEDULING FREQUENCY (minutes): 9999// <RET>
DT/TM LAST STARTED OR STOPPED: <RET>
HANG TIME (seconds): 1// <RET>
GET WORK FUNCTION (TAG):  // <RET>
GET WORK FUNCTION (ROUTINE):  // <RET>
DO WORK FUNCTION (TAG): <RET>
DO WORK FUNCTION (ROUTINE): <RET>
MAX TRIES FINDING WORK: 0// <RET>
PERSISTENT: NO// <RET>
DEDICATED LINK: VABAY←  This is the name of the entry in the HL LOGICAL LINK File
(#870) that is the default listener to which most remote applications will send
messages, as shown above
VMS TCP SERVICE: YES// <RET>

Select HLO PROCESS REGISTRY PROCESS NAME: <RET>
```

3.5 Schedule the HLO COUNT RECORDS Option

The HLO COUNT RECORDS option triggers HLO to count incoming and outgoing messages at a user specified frequency. This option should be scheduled to run at least once a day and can be scheduled to run more frequently, if desired. The "RESCHEDULING FREQUENCY" parameter determines how often this process runs. In the example below, the process is scheduled to run once every six hours. While "H" is used here to denote hours, "M" can be used to denote minutes.

```
Select OPTION NAME: XUTM MGR       Taskman Management

          Schedule/Unschedule Options
          One-time Option Queue
          Taskman Management Utilities ...
          List Tasks
          Dequeue Tasks
          Requeue Tasks
          Delete Tasks
          Print Options that are Scheduled to run
          Cleanup Task List
          Print Options Recommended for Queuing

Select Taskman Management Option: Schedule/Unschedule Options

Select OPTION to schedule or reschedule: HLO COUNT RECORDS       COUNT HL7 MESSAGE
RECORDS
  Are you adding 'HLO COUNT RECORDS' as
    a new OPTION SCHEDULING (the 201ST)? No// Y (Yes)

------------------------------------------------------------------------
                          Edit Option Schedule
    Option Name: HLO COUNT RECORDS
    Menu Text: COUNT HL7 MESSAGE RECORDS                TASK ID:
  ______________________________________________________________________

  QUEUED TO RUN AT WHAT TIME: T+1@1AM  (AUG 25, 2005@01:00)
 DEVICE FOR QUEUED JOB OUTPUT:
  QUEUED TO RUN ON VOLUME SET:
       RESCHEDULING FREQUENCY: 6H
              TASK PARAMETERS:
             SPECIAL QUEUEING:
_____________________________________________________________________
Exit     Save     Next Page     Refresh

Enter a command or '^' followed by a caption to jump to a specific field.
------------------------------------------------------------------------
```

(Save and Exit from the Edit Option Schedule screen)

3.6 Schedule the HLO SYSTEM STARTUP Option

The HLO SYSTEM STARTUP option should be scheduled after installing and configuring HLO. Once this option is scheduled, HLO starts automatically at system startup. This option can be scheduled from the Taskman Management Menu.

WARNING: Not scheduling this option requires the IRM to start HLO manually from the HLO System Monitor.

```
Select OPTION NAME: XUTM MGR       Taskman Management

          Schedule/Unschedule Options
          One-time Option Queue
          Taskman Management Utilities ...
          List Tasks
          Dequeue Tasks
          Requeue Tasks
          Delete Tasks
          Print Options that are Scheduled to run
          Cleanup Task List
          Print Options Recommended for Queuing

Select Taskman Management Option: Schedule/Unschedule Options

Select OPTION to schedule or reschedule: HLO SYSTEM STARTUP

  Are you adding 'HLO SYSTEM STARTUP' as
    a new OPTION SCHEDULING (the 202ND)? No// Y (Yes)


------------------------------------------------------------------
                     Edit Option Schedule
    Option Name: HLO SYSTEM STARTUP
    Menu Text: HL7 (Optimized) SYSTEM STARTUP            TASK ID:
  ________________________________________________________________

  QUEUED TO RUN AT WHAT TIME:
 DEVICE FOR QUEUED JOB OUTPUT:
  QUEUED TO RUN ON VOLUME SET:
       RESCHEDULING FREQUENCY:
              TASK PARAMETERS:
             SPECIAL QUEUEING: STARTUP
 ________________________________________________________________
Exit     Save     Next Page     Refresh

Enter a command or '^' followed by a caption to jump to a specific field.
------------------------------------------------------------------


(Save and Exit from the Edit Option Schedule screen)
```

3.7 Start HLO using the HLO System Monitor

To start HLO, select the START HLO action protocol from the HLO System Monitor. Please refer to Section 5.2 of this document for more detailed instructions.

```
HLO SYSTEM MONITOR            Aug 26, 2005@09:40:25          Page:    1 of    1
 Brief Operational Overview
SYSTEM STATUS:              STOPPED
PROCESS MANAGER:            STOPPED
STANDARD LISTENER:          NOT OPERATIONAL
INTERFACE ENGINE:           NOT OPERATIONAL
TASKMAN:                    RUNNING
DOWN LINKS:                 0
CLIENT LINK PROCESSES:      0
IN-FILER PROCESSES:         0
MESSAGES PENDING TRANSMISSION:      0
STOPPED OUTGOING QUEUES:
MESSAGES PENDING ON APPLICATIONS:   0
STOPPED INCOMING QUEUES:
FILE 777 RECORD COUNT:              7     --> as of Aug 14, 2005@07:53:01
FILE 778 RECORD COUNT:              7     --> as of Aug 14, 2005@07:53:01


          Brief System Status
LP  LIST PROCESSES        BS  BRIEF STATUS          TL  TEST TCP LINK
DL  DOWN LINKS            ML  MONITOR LINK          RT  RealTime Mode
OQ  OUTGOING QUEUES           STOP HLO              SM  Scroll Mode
IQ  INCOMING QUEUES           START HLO             SQ  STRT/STP QUE
Select Action:Quit// START    START HLO
```

After start up of HLO, the following should be verified:

- SYSTEM STATUS is RUNNING
- PROCESS MANAGER is RUNNING
- STANDARD LISTENER will still be NOT OPERATIONAL until TCP Service is created and enabled, see 3.8 below
- INTERFACE ENGINE is OPERATIONAL, if the Interface Engine is used.
- TASKMAN is RUNNING

3.8 Create and Activate the TCPIP Services for Open VMS

Please refer to the next chapter 'Listeners' for information on configuring the TCPIP Services for Open VMS.

WARNING: The TaskMan Multi-Listener should NOT be used on systems running Cache under OpenVMS. For any system required to use the TaskMan Multi-Listener (such as those running Cache under NT), **patch XU*8.0*388 must be installed first**.

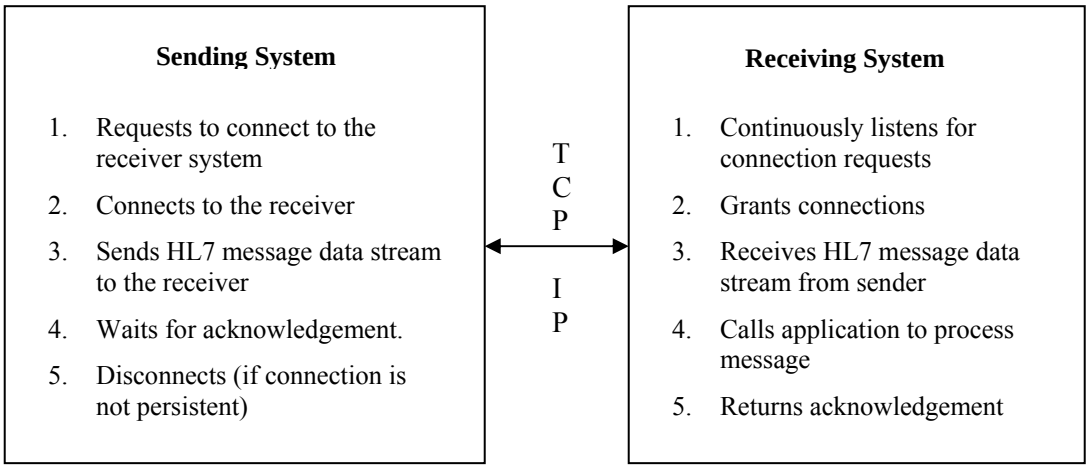
4.0 Listeners

4.1 Introduction

HLO uses TCP/IP listeners to "listen" on a particular port for incoming TCP/IP connections from other systems. Listeners are necessary so that other systems may initiate a connection to this VistA system over TCP/IP.

Client and Server Roles in HLO over TCP/IP
Two separate sets of M code define the roles of client and server over TCP/IP channels:

- Sending System = TCP Client (initiates connection to the Receiving System)
- Receiving System = TCP Server (listens for connections)



4.1.1 TCP/IP Connection Requirements

If the system connecting to VistA is a non-VistA system, it must support synchronous bi-directional TCP/IP transmission. This means that when a message is sent over a TCP/IP connection, the expected response to that message is returned immediately over the same open connection. The sending system must not initiate a new transmission until the current transmission is complete. The receiving system must respond to the original message without attempting to initiate a new connection.

The sending and receiving system can not change roles over the same connection. If the receiver needs to send transmissions (other than commit acknowledgements), then it must open a new connection.

If VistA is to connect to a target system, the target system must have its own TCP/IP listener process that responds to connection requests.

The TCP/IP connection can be *persistent* or *non-persistent*. This is determined by the connecting system. If the connecting system drops the connection after a transmission completes, the connection is non-persistent. If it is left open, the connection is persistent.

Three types of listeners or server processes are provided in the current HLO software distribution. The three listeners included are:

1. TCPIP Services for Open VMS
2. TaskMan Multi-Threaded Listener
3. Single Listener

WARNING:

- One Single Listener server process is currently provided in the HLO software. However, configuration of multiple Single Listeners is NOT supported by HLO at this time.

- Applications that require a dedicated Single Listener should continue to use the original HL 1.6 implementation. A subsequent HLO patch is being developed for a future release that provides full support of multiple Single Listeners and additional Multi-Threaded Listeners.

- When trying to decide between a single listener and a multi-listener, if only one connection request will be created at a time and the interfacing application requires its own server, then a single listener would be applicable. Otherwise, if there is a possibility of multiple connection requests then the multi-listener is appropriate.

- The remainder of this chapter will focus on the two types of Multi-Threaded Listener.

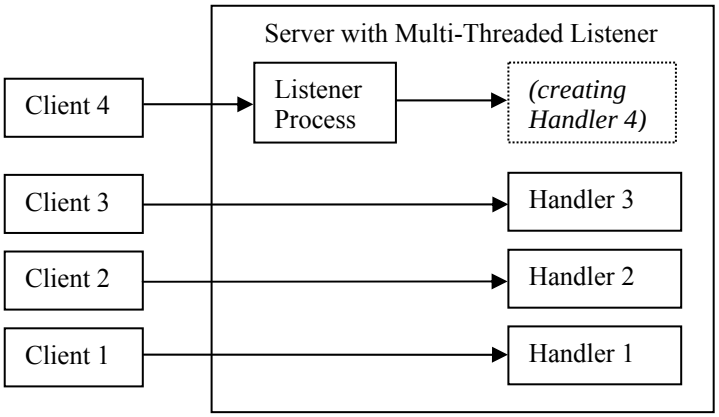
WARNING: HLO highly recommends use of the TCPIP Services for Open VMS. However, sites NOT on the OpenVMS platform will be required to use the TaskMan Multi-Threaded Listener.

To reference additional information on listener configuration, please refer to the following documents in the VistA Documentation Library:

- User Manual: TCP/IP Supplement – HL*1.6*19 (January 1999)
- Site Manager & Developer Manual – HL*1.6*56 (December 1999)

4.2 Multi-Threaded Listeners

Multi-threaded listeners are useful when multiple connection requests come to a single port from many devices or systems. Because multi-threaded listeners spawn off separate handlers for each client connection request, they enable multiple concurrent connections.



VistA HL7 supports two types of multi-threaded listeners:

- TCPIP Services for Open VMS (for DSM or Cache)
- TaskMan Multi-Threaded Listener

4.3 TCPIP Services for Open VMS

Sections 4.3.1 through 4.3.5 are for both Cache on OpenVMS and DSM on Open VMS. 4.3.6 is for Cache users only and 4.3 7 is for DSM users only

4.3.1 Introduction

Multi-Listeners using TCPIP Services for Open VMS for Cache/DSM sites were introduced in patch HL*1.6*84. This chapter documents the setup for creating multi-listeners for HLO using TCPIP Services for OpenVMS. It assumes that a DCL command file for HL7 1.6 already exists and can be copied as the starting basis for the new HLO service.

4.3.2 TCP/IP Services for OpenVMS

TCP/IP is an open communications standard that enables any connected host to communicate with any other connected host. TCP/IP Services for OpenVMS is a product that implements several of the protocols in the TCP/IP standard for the OpenVMS operating system. This section focuses only on those TCP/IP services configured to run as a TCP/IP server (listener) process.

4.3.3 TCP/IP Services and VistA HLO

A TCP/IP service configured to run as a server permits multiple remote TCP/IP clients to connect and run concurrently up to the limits established by the service. A server listens on a particular TCP/IP communication port and launches a specified DCL (Digital Command Language) Command file that serves as a startup process for each client connection process. This startup file contains the necessary commands to execute the entry point into VistA HLO.

4.3.4 Requirements for Setting up a TCP/IP Service on OpenVMS

To configure a TCP/IP service, the following components within VistA HLO and OpenVMS will need to be configured:

- VistA HLO logical link for the Multi-Threaded Listener.
- An OpenVMS account. **(If an account already exists for HL7 1.6, use the same user and home directory.)**
- An OpenVMS home directory. **(If an account already exists for HL7 1.6, use the same user and home directory.)**
- An OpenVMS DCL command procedure. This is the startup command file that executes on every concurrent process. Default DCL command files are provided in this document.
- An OpenVMS TCP/IP service.

The person implementing the instructions in this document must have OpenVMS system administrator privileges to create the above components and be familiar with the OpenVMS TCP/IP Services Management Control Program.

4.3.5 Recommended Naming Conventions

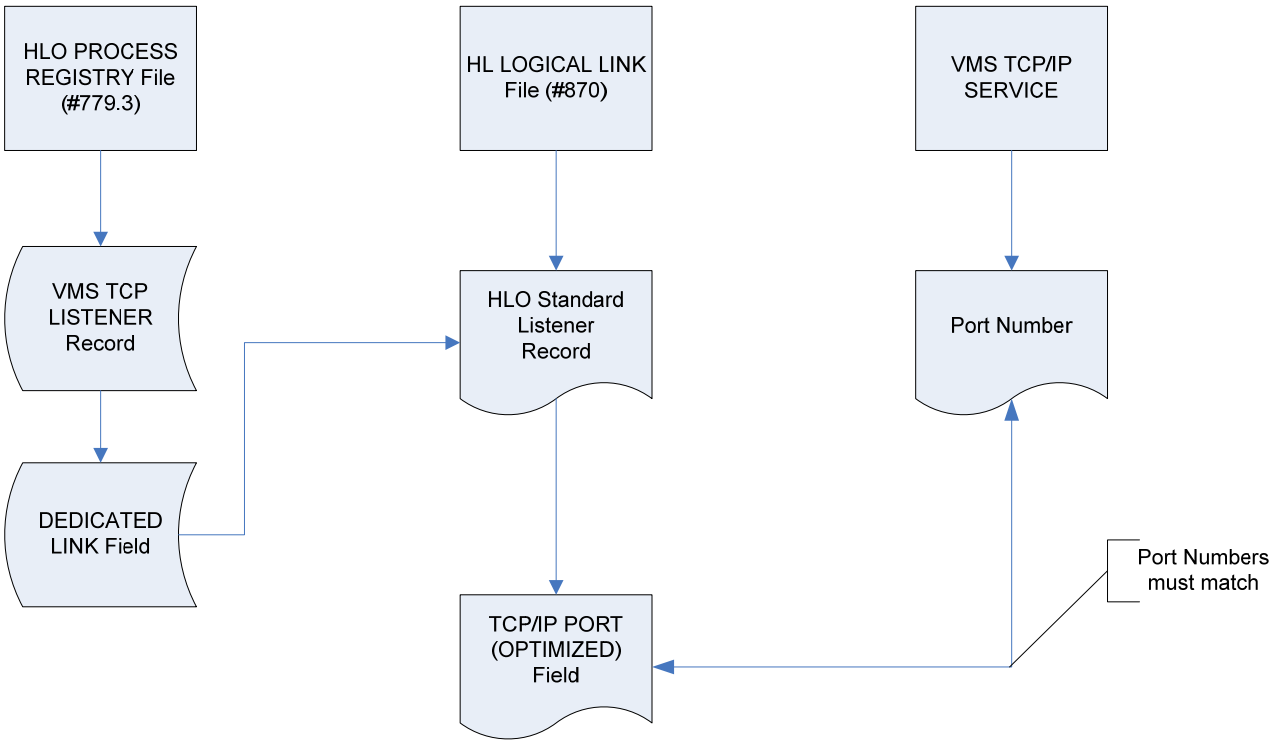
The following names are used in the description for creating a TCP/IP service and are referenced throughout this chapter. All these names are suggestions. Your site might already have its own naming convention:

- HLSEVEN — OpenVMS user account name for an HLO TCP/IP service.
- [HLSEVEN] — Name of home directory for the above HLSEVEN user.
- HLSEVEN — Name of the owner.

    - Sites that have previously established a VMS user account for HL7 1.6 may reuse the same account for HLO, i.e., another VMS account need not be created.
    - The same user name, HLSEVEN, is recommended for HL7 prior to the release of HLO.
    - However, a new TCPIP service command procedure specific for HLO must be created and placed in the same home directory as the old HL7 1.6 TCPIP service.

- HLS<port><M environment>.COM— Name of DCL command procedure, where the <port> is the actual port number where the service will be listening, and the <M environment> is the actual VistA M environment. For example:
    - HLS5001DSM.COM— represents the command procedure for a TCP/IP service listening on port 5001 (production systems) that starts up a DSM HLO listener process.
    - HLS5001CACHE.COM— represents the command procedure for a TCP/IP service listening on port 5001 (production systems) that starts up a CACHE HLO listener process.
    - HLS5026DSM.COM— represents the command procedure for a TCP/IP service listening on port 5026 (test systems) that starts up a DSM HLO listener process.
    - HLS5026CACHE.COM— represents the command procedure for a TCP/IP service listening on port 5026 (test systems) that starts up a CACHE HLO listener process.
- HLS<port><M environment>— Name of a TCP/IP service, where the <port> is the actual port number where the service will be listening, and the <M environment> is the actual VistA M environment. For example:
    - HLS5001DSM— represents the TCP/IP service listening on port 5001 that starts up a DSM HLO listener process.
    - HLS5001CACHE— represents the TCP/IP service listening on port 5001 that starts up a CACHE HLO listener process.
    - HLS5026DSM— represents the TCP/IP service listening on port 5026 (test systems) that starts up a DSM HLO listener process.
    - HLS5261CACHE— represents the TCP/IP service listening on port 5026 (test systems) that starts up a CACHE HLO listener process.

WARNING: The TCP/IP PORT (OPTIMIZED) field value for the DEDICATED LINK assigned to the VMS TCP LISTENER process must match the Port Number associated with the VMS TCP/IP service.



- More than one TCP/IP service for HLO may be set up, although it is not necessary to do this. To set up more than one TCP/IP service for HLO, follow the steps in this document for each listener. However, a different command file name, TCP/IP service name, and port number must be defined for each listener.
- Optionally, different user accounts and directories may be specified for each listener.

4.3.6 Creating a TCPIP Services for Open VMS with Cache

General steps for creating a TCPIP Services for Open VMS with Cache are as follows:

1. Create OpenVMS User Account **(If a user account already exists for HL7 1.6, use the same user account and home directory.)**
2. Create OpenVMS Home Directory **(If a user account already exists for HL7 1.6, use the same user account and home directory.)**
3. Create a DCL Command Procedure

4. Set up the TCP/IP Service

5. Enable and Save the TCP/IP Service

6. Control the Number of Log Files Created by TCP/IP Services

- Other TCP/IP Service Commands

Note for Multi-Node Cluster Sites:

For sites configured with a multi-node cluster, more than one node may be advertised under the domain name HL7.SITENAME.MED.VA.GOV and the TCP/IP service may be running on multiple nodes.

In addition, the impersonator VMS feature allows for the possibility of all nodes in the cluster to become the surrogate. This allows for the listening process to remain uninterrupted if the TCP/IP service is enabled on all nodes in the cluster.

If this is the case for your site, be sure to enable the service on all these nodes, after setting up the TCP/IP service and COM file on one of these nodes.

4.3.6.1 Create OpenVMS User Account

To create an OpenVMS User Account:

- **If an account already exists for HL7 1.6, use the same user account.** Review the settings for that user account to insure conformance to the screen below, then skip to section 4.3.6.3 'Create a DCL Command Procedure'.
- Determine an unused User Identification Code (UIC), typically in the same group as other Cache for OpenVMS accounts.
- Using the OpenVMS Authorize utility, add the new HLSEVEN account with the unused UIC. You must have SYSPRV to do this.
- Verify that the account settings for the new HLSEVEN account are the same as they appear in the example that follows; or, if they are different, verify that the impact of the different settings is acceptable for your system. For security, make sure that the DisCtlY, Restricted, and Captive flags are set.

There are two different ways to set up a new user account, and you are free to choose the one you prefer. The following two examples illustrate two different ways to set up an OpenVMS User account:

One way to set up an OpenVMS User account is to copy your existing XMINET (TCP/IP MailMan) account to a new account with an unused UIC. For example:

```
$ MC AUTHORIZE
UAF> COPY /ADD XMINET HLSEVEN/UIC=[51,45]/DIR=[HLSEVEN]
%UAF-I-COPMSG, user record copied
%UAF-W-DEFPWD, copied or renamed records must receive new password
%UAF-I-RDBADDMSGU, identifier HLSEVEN value [000051,000045] added to rights
database
UAF>
```

The other way to set up an Open VMS User account is to add the new HLSEVEN OpenVMS account directly. For example:

```
$ MC AUTHORIZE
 UAF> ADD HLSEVEN /UIC=[100,45]/OWNER="HLSEVEN" - (must use continuation
character "-")
_UAF> /DEVICE=USER$/DIRECTORY=[HLSEVEN] -
_UAF> /NOACCESS/NETWORK/FLAGS=(DISCTLY,RESTRICTED,NODISUSER) -
_UAF> /PRIV=(NETMBX,TMPMBX) -
_UAF> /DEF=(NETMBX,TMPMBX)/LGICMD=NL:
%UAF-I-ADDMSG, user record successfully added
%UAF-I-RDBADDMSGU, identifier HLSEVEN value [000100,000045] added to rights
data
base
UAF>

UAF> SHOW HLSEVEN

Username: HLSEVEN                          Owner:  HLSEVEN
Account:                                   UIC:    [100,45] ([HLSEVEN])
CLI:      DCL                              Tables: DCLTABLES
Default:  USER$:[HLSEVEN]
LGICMD:   NL:
Flags:  DisCtlY Restricted
Primary days:   Mon Tue Wed Thu Fri
Secondary days:                      Sat Sun
Primary   000000000011111111112222  Secondary 000000000011111111112222
Day Hours 012345678901234567890123  Day Hours 012345678901234567890123
Network:  ##### Full access ######            ##### Full access ######
Batch:    -----  No access  ------             -----  No access  ------
Local:    -----  No access  ------             -----  No access  ------
Dialup:   -----  No access  ------             -----  No access  ------
Remote:   -----  No access  ------             -----  No access  ------
Expiration:            (none)    Pwdminimum:  6   Login Fails:     0
Pwdlifetime:         90 00:00    Pwdchange:      (pre-expired)
Last Login:            (none) (interactive),            (none) (non-
interactive)
```

```
Maxjobs:          0  Fillm:       100  Bytlm:        64000
Maxacctjobs:      0  Shrfillm:      0  Pbytlm:           0
Maxdetach:        0  BIOlm:       150  JTquota:       4096
Prclm:            8  DIOlm:       150  WSdef:         2000
Prio:             4  ASTlm:       250  WSquo:         4000
Queprio:          4  TQElm:        10  WSextent:     16384
CPU:         (none)  Enqlm:      2000  Pgflquo:      50000
Authorized Privileges:
  NETMBX       TMPMBX
Default Privileges:
  NETMBX       TMPMBX
UAF>  Exit
%UAF-I-DONEMSG, system authorization file modified
%UAF-I-RDBDONEMSG, rights database modified
$
```

4.3.6.2 Create OpenVMS Home Directory

If a home directory already exists for HL7 1.6, then use the same home directory. Skip to section 4.3.6.3 'Create a DCL Command Procedure'.

This directory will house the DCL command procedure, which is executed whenever a client connects. A log file is created for every instance of a connection for that listener. Make sure that the owner of the directory is the HLSEVEN account.

For example, to create a home directory named [HLSEVEN] with ownership of HLSEVEN:

```
$ CREATE/DIR [HLSEVEN]/OWNER=HLSEVEN
```

4.3.6.3 Create a DCL Command Procedure

Create a DCL command procedure (shown below) in the home directory for the HLSEVEN account and name it according to the recommended convention. Make sure the command procedure file is owned by the HLSEVEN account.

1. To create a DCL command procedure that will use a given port, for port 5001, name your command procedure file as HLS5001CACHE.COM.
2. Adjust the Cache command line (Cache configuration, UCI, and volume set) for your system.
3. Ensure that the name of the DCL command file, as described in step 1, matches the port assignment. For example, if you changed the port number from 5001 to 6788, rename your HLS5001CACHE.COM file to HLS6788CACHE.COM.

WARNING: All VistA sites must use Port #5001 for the HLO Standard Listener for production accounts. For test accounts Port #5026 must be used.

Before creating the Command Procedure file determine the proper Cache configuration to use for the environment where you want to start your listener. To do that, use command "CCONTROL LIST" and it will all Cache configurations that are defined. The Cache configuration you will most likely need is the one marked as (default).

If you are not running a cluster or if the listener is to run on only a single node of the cluster, you can use the name of that default Cache configuration as the first parameter to the 'CSESSION' command.

If you are running a cluster and the listener is to run on multiple nodes of that cluster, then you need to make sure that DCL Command Procedure file can resolve the proper name of the default Cache configuration on **each node** of the cluster where it is to run. Keep in mind that same DCL command file has to work on each participating node.

On most VistA system, the name of Cache configuration will be the same as name of the node or perhaps a derivative of the name of the node. So, if the node is 74A01, the configuration will be 74A01 or maybe BRX74A01. In those cases, DCL Command Procedure file will need to use 'F$GETSYS("NODENAME")' to obtain node name or BRX'F$GETSYS("NODENAME") to obtain node name and stick "BRX" in front of it.

For your convenience, you can cut and paste the following DCL command procedure file into your OpenVMS HLSEVEN device and directory.

Sample DCL Command Procedure file:

```
$! HLS5001CACHE.COM - for incoming tcp connect requests with port=5001 and
$! using "Cache" command line to enter the M environment
$! File name HLS5001CACHE.COM is recommended to be changed to reflect the
$! change of the TCPIP port number.  For example, file name could be
$! changed to HLS6788CACHE.COM if port=6788.
$!
$!this file is copied and modifed from HLSEVEN.COM
$! Revision History:
$! Patch HL*1.6*19 & HL*1.6*56--Documentation only
$! Patch HL*1.6*70--HL71_6P70.COM
$! Patch HL*1.6*84--HLS5001CACHE.COM and HLS5001DSM.COM
$!-------------------------------------------------------------
$ set noon        !Don't stop
$ set noverify    !change as needed
$! set verify     !change as needed
$ purge/keep=5 sys$login:*.log !Purge log files only
$ set proc/priv=(share)  !Required for MBX device
$ x=f$trnlnm("sys$net")  !This is our MBX device
$!
$ write sys$output "Opening "+x !This can be viewed in the log file
$! Check status of the BG device before going to either DSM or Cache'
$ cnt=0
$ CHECK:
$ stat=f$getdvi("''x'","STS")
$ if cnt .eq. 10
$ then
$ write sys$output "Could not open ''x' - exiting"
$ goto EXIT
$ else
$    if stat .ne. 16
$    then
$    cnt=cnt+1
```

```
$    write sys$output "''cnt'> ''x' not ready!"
$    wait 00:00:01 !Wait one second to assure connection
$    goto CHECK
$    else
$    write sys$output "''x' is now ready for use - entering DSM or Cache"
$!-------------------------------------------------------------
$! **Be sure the command line(s) in the COMMAND LINE SECTION
$! **below is correct for your system and if access control is
$! **enabled, that this account has access to this uci,vol & routine.
$! **An entry in file 870 for this logical link with the specified
$! **unique port number and its device type as "MS"(Multi-threaded
$! **server) must be existed.
$!
$! **Also, comment or uncomment the appropriate lines for your system.
$!
$!-------------------------------------------------------------
$! COMMAND LINE SECTION:
$! ====================
$!-------------------------------------------------------------
$! for DSM
$!dsm/env=dsmmgr/uci=vah/vol=tou VMS^HLOSRVR
$!-------------------------------------------------------------
$! for Cache
$! The first parameter after csession is the name of Cache configuration that
$! corresponds to the environment where listener should be run
$ assign 'f$trnlnm("SYS$NET")' SYS$NET
$!
$! The following line is an example for single or integrated sites
$ csession ‘F$GETSYI("NODENAME") "-U" "VAH" "VMS^HLOSRVR"
$!
$! The following line is an example for consolidated sites
$! csession BRX’F$GETSYI("NODENAME") "-U" "VAH" "VMS^HLOSRVR"
$!
$! The following line is an example for non-cluster sites
$! csession MYSITE "-U" "VAH" "VMS^HLOSRVR"
$!
$! The following live is an example for a single test account
$! csession MYSITE "-U" "TST" "VMS^HLOSRVR"
$!-------------------------------------------------------------
$ endif
$ exit:
$ logout/brief
```

Double Checking the Setup:

After creating/editing the command file, run the following command and make sure that the 'Owner:' field matches the user account which will be running the command file. (In our examples that is 'HLSEVEN' as shown.)

```
$ dir/full hls5001cache.com

Directory DKC0:[USER.HLSEVEN]

HLS5001CACHE.COM;1                File ID:  (13869,1,0)
Size:            6/6          Owner:    [USERS,HLSEVEN]
Created:    23-FEB-2005 16:59:14.91
Revised:    23-FEB-2005 17:01:16.70 (3)
Expires:    <None specified>
Backup:     <No backup recorded>
```

```
Effective:  <None specified>
Recording:  <None specified>
Accessed:   <None specified>
Attributes: <None specified>
Modified:   <None specified>
Linkcount:  1
File organization:  Sequential
Shelved state:      Online
Caching attribute:  Writethrough
File attributes:    Allocation: 6, Extend: 0, Global buffer count: 0
                    No version limit
Record format:      Variable length, maximum 255 bytes, longest 77 bytes
Record attributes:  Carriage return carriage control
RMS attributes:     None
Journaling enabled: None
File protection:    System:RWED, Owner:RWED, Group:RE, World:
Access Cntrl List:  None
Client attributes:  None

HLS5001CACHE.COM;1                File ID:  (13868,1,0)
Size:            6/6          Owner:    [USERS,HLSEVEN]
Created:    23-FEB-2005 16:54:25.63
Revised:    23-FEB-2005 17:01:16.72 (3)
Expires:    <None specified>
Backup:     <No backup recorded>
Effective:  <None specified>
Recording:  <None specified>
Accessed:   <None specified>
Attributes: <None specified>
Modified:   <None specified>
Linkcount:  1
File organization:  Sequential
Shelved state:      Online
Caching attribute:  Writethrough
File attributes:    Allocation: 6, Extend: 0, Global buffer count: 0
                    No version limit
Record format:      Variable length, maximum 255 bytes, longest 77 bytes
Record attributes:  Carriage return carriage control
RMS attributes:     None
Journaling enabled: None
File protection:    System:RWED, Owner:RWED, Group:RE, World:
Access Cntrl List:  None
Client attributes:  None

Total of 2 files, 12/12 blocks.

NOTE: Could also use the following command if you think this is more clear.

$ dir/owner HLS5001CACHE.COM

Directory DKC0:[USER.HLSEVEN]

HLS5001CACHE.COM;1       [USERS,HLSEVEN]

Total of 1 files.
```

4.3.6.4 Set up the TCP/IP Service

To create the TCP/IP service to listen for connections:

- Choose the OpenVMS node where you want to run the TCP/IP service listener. This is also the node whose IP address will be advertised to other systems as the location of your HL7 listener.
- Use TCP/IP port number 5001 for production systems and 5026 for the main test system.

WARNING: All VistA sites must use Port #5001 for the HLO Standard Listener for production accounts. For test accounts Port #5026 must be used.

- Use user account HLSEVEN.
- Use DCL command file name HLS5001CACHE.COM.

Since TCP/IP Services are node specific, make sure to set up the listener on the same node on which it will be running.

Ensure that your new TCP/IP service uses the recommended naming convention. For example, to set up a service that will be listening on port 5001 and use a corresponding DCL command file HLS5001CACHE.COM.

Set the service name as HLS5001CACHE as follows:

```
$ TCPIP          (must use continuation character "-" at end of long lines)
TCPIP> SET SERVICE HLS5001CACHE/USER=HLSEVEN/PROC=HLS5001CACHE/PORT=-
_TCPIP> 5001/PROTOCOL=TCP/REJECT=MESSAGE="All channels busy" -
_TCPIP> /LIMIT=50/FILE=USER$:[HLSEVEN]HLS5001CACHE.COM/INACTIVITY=1
```

In this command, **/LIMIT=50** specifies the maximum number of TCP/IP connections that can be made at any time. The limit of 50 is appropriate for most local sites, but for systems that serve as national databases the limit should be set initially to 500. The system manager is responsible for monitoring the peak number of connections made, and if the peak approaches the limit, the limit should be increased.

- If you get an error because you mistyped any of the above lines or forgot to use the continuation character "-", we suggest you do the following to remove the corrupted service and repeat the above commands.

```
TCPIP> SET CONFIG ENABLE NOSERVICE HLS5001CACHE
TCPIP> SET NOSERVICE HLS5001CACHE
```

```
TCPIP> SHO SERVICE HLS5001CACHE/FULL
```

```
Service: HLS5001CACHE
                           State:     Disabled
Port:             5001     Protocol:  TCP             Address:  0.0.0.0
User_name: not defined     Process:   HLS5001CACHE
```

4.3.6.5 Enable and Save the TCP/IP Service

Since TCP/IP Services is node specific, make sure you are on the same node that you want the listener to run on.

```
TCPIP> ENABLE SERVICE HLS5001CACHE          (enable service immediately)
TCPIP> SET CONFIG ENABLE SERVICE HLS5001CACHE (save service for reboot)
TCPIP> SHO SERVICE/FULL HLS5001CACHE

Service: HLS5001CACHE
                           State:     Enabled
Port:          5001     Protocol:  TCP          Address:  0.0.0.0
Inactivity:       5     User_name: HLSEVEN      Process:  HLS5001CACHE
Limit:           50     Active:      0          Peak:       0

File:          USER$:[HLSEVEN] HLS5001CACHE.COM
Flags:         Listen

Socket Opts:   Rcheck Scheck
 Receive:             0     Send:               0

Log Opts:      None
 File:         not defined

Security
 Reject msg:   All channels busy

 Accept host: 0.0.0.0
 Accept netw: 0.0.0.0

TCPIP> SHO CONFIG ENABLE SERVICE

Enable service
     FTP, FTP_CLIENT, HLS5001CACHE, MPI, TELNET, XMINETMM
TCPIP> EXIT
```

Note for Multi-Node Cluster Sites:

For sites configured with a multi-node cluster, more than one node may be advertised under the domain name HL7.SITENAME.MED.VA.GOV and the TCP/IP service may be running on multiple nodes.

In addition, the impersonator VMS feature allows for the possibility of all nodes in the cluster to become the surrogate. This allows for the listening process to remain uninterrupted if the TCP/IP service is enabled on all nodes in the cluster.

If this is the case for your site, be sure to enable the service on all these nodes, after setting up the TCP/IP service and COM file on one of these nodes.

4.3.6.6 Control the Number of Log Files Created by TCP/IP Services

The HLS5001CACHE TCP/IP service automatically creates log files (TCP/IP services does this and it cannot be prevented) in the HLSEVEN directory named HLS5001CACHE.LOG;xxx where 'xxx' is a file version number. New versions of this file will be created until that file version number reaches the maximum number of 32767. In order to minimize the number of log files created, you may want to initially rename this log file to the highest version number with the command:

```
$ RENAME USER$:[HLSEVEN]HLS5001CACHE.LOG; USER$:[HLSEVEN]HLS5001CACHE.LOG;32767
```

Alternatively, you can set a limit on the number of versions of the log file that can concurrently exist in the HLSEVEN directory:

```
$ SET FILE /VERSION_LIMIT=10 USER$:[HLSEVEN]HLS5001CACHE.LOG;
```

NOTE: This cannot be done until the first log file has actually been created.

You probably should not limit the number of versions of the log file until you know that your HLS5001CACHE service is working correctly; keeping the log files can help when diagnosing problems with the service/account.

4.3.6.7 Other TCP/IP Service Commands

WARNING: If HLO is stopped or disabled for two hours or more, the VMS Multi-Listener service should be disabled and then re-enabled before restarting HLO.

The definition of a link is required for the multi-threaded listener for Open VMS systems. This link never needs to be started or stopped through the VistA HL 1.6 or HLO options. Instead, it is normally started and stopped via TCP/IP services. For example:

```
TCPIP> DISABLE SERVICE HLS5001CACHE        (Stop TCP/IP service)
```

```
TCPIP> ENABLE SERVICE HLS5001CACHE          (Start TCP/IP service)
```

Any questions about configuring TCPIP Service for OpenVMS should be directed to EVS for assistance.

4.3.7 Creating a TCPIP Services for Open VMS with DSM

General steps for creating a TCPIP Services for Open VMS with DSM are as follows:

(For more detailed information on creating a TCPIP Services for Open VMS with DSM, please refer to Appendix F.)

1. Create OpenVMS User Account **(If an account already exists for HL7 1.6, use the same user and home directory.)**
2. Create OpenVMS Home Directory **(If an account already exists for HL7 1.6, use the same user and home directory.)**
3. Create a DCL Command Procedure
4. Set up the TCP/IP Service
5. Enable and Save the TCP/IP Service
6. Resolve Access Control List (ACL) Issues
7. Control the Number of Log Files Created by TCP/IP Services
8. Other TCP/IP Service Commands

4.4 TaskMan Multi-Threaded Listener

The configuration of the TaskMan Multi-Threaded listener involves the following steps:

1. Set up the server logical link in the HL LOGICAL LINK File (#870). This step may have been done already. If that is the case, proceed to the next step. Otherwise refer to Section 3.2 of this document for setup details.
2. Configure the TASKMAN MULTI-LISTENER record in the HLO PROCESS REGISTRY File (#779.3): Please refer to Section 3.4 of this document for setup details.
   a. Assign the server logical link name (defined in step #1) to the DEDICATED LINK field.
   b. Set the ACTIVE field to 'YES'.

WARNING: The TaskMan Multi-Listener should NOT be used on systems running Cache under OpenVMS. For any system required to use the TaskMan Multi-Threaded Listener (such as those running Cache under NT), **patch XU*8.0*388 must be installed first**.

Please contact the site IRM to obtain the specific domain name and port number to be used by the client side communicating with VistA. For example, at many sites the HL 1.6 multi-listener uses HL7.SITENAME.MED.VA.GOV with port 5000. Currently, HLO uses the same domain name with port 5001on production systems and 5026 on test systems.

REMINDER: TaskMan must be running for the TaskMan Multi-Threaded Listener to function.

4.4.1 Set up the Server Logical Link

In VistA HL LOGICAL LINK File (#870), create an entry for the Multi-Threaded Listener with the fields populated as follows. If more assistance is required for defining the HL Logical Link, please refer to Section 2 of the chapter 'HLO Installation and Configuration' for more specific instructions.

Link Settings for the TaskMan Multi-Threaded Listener in the Logical Link File (#870)

Field	Description
LLP TYPE	Set to 'TCP'
TCP/IP SERVICE TYPE	Set to 'MULTI LISTENER'
TCP/IP ADDRESS	IP Address for your server
TCP/IP PORT (OPTIMIZED)	Port to listen on, e.g., 5001 for production systems and 5026 for test systems (make note of the exact port number)

- The port number you select must be an available TCP/IP port number. The port number will also be used in the configuration and naming of the TCP/IP service described in the following sections.
- The port number recommended in this chapter, 5001 for production and 5026 for test, has been registered for use by VistA HLO. Everything should be done to free port 5001 for use by HLO.
- If the HLO multi-listener is to be used by an application at the national level and you are not using port number 5001 for production, you must register the port number with the DBIA manager on Forum.

WARNING: All VistA sites must use Port #5001 for the HLO Standard Listener for production accounts. For test accounts Port #5026 must be used.

If the TCP/IP Address is not defined, it is resolved automatically by the VHA DNS Domain server. If the system's domain is NOT registered in the VHA DNS server, the IP address should be defined.

For configuring client logical links please refer to Section 6.2 of this document.

4.4.2 Configure the TaskMan Multi-Listener Record in the HLO Process Registry

The Logical Link name defined in section 4.4.1 must be added to the DEDICATED LINK field (#.14) in the VMS TCP Listener Process entry of the HLO PROCESS REGISTRY File (#779.3). The ACTIVE field (#.02) should also be checked to make sure that it is set to "YES."

```
Select HLO PROCESS REGISTRY PROCESS NAME: TASKMAN MULTI LISTENER
PROCESS NAME: TASKMAN MULTI LISTENER// <RET>
ACTIVE: NO// YES<RET>
MINIMUM ACTIVE PROCESSES: 1// <RET>
MAXIMUM ACTIVE PROCESSES: 1// <RET>
SCHEDULING FREQUENCY (minutes): 9999// <RET>
DT/TM LAST STARTED OR STOPPED: <RET>
HANG TIME (seconds): 1// <RET>
GET WORK FUNCTION (TAG):  // <RET>
GET WORK FUNCTION (ROUTINE):  // <RET>
DO WORK FUNCTION (TAG): <RET>
DO WORK FUNCTION (ROUTINE): <RET>
MAX TRIES FINDING WORK: 0// <RET>
PERSISTENT: NO// <RET>
DEDICATED LINK: // VABAY<RET>
VMS TCP SERVICE: NO// <RET>

Select HLO PROCESS REGISTRY PROCESS NAME: <RET>
```

5.0 HLO Management System

5.1 Main Menu

The HLO Main Menu is the primary entry point for managing and monitoring the HLO system. Two primary types of users will find the Main Menu useful. IRMs will find it useful for managing the various HLO processes, as well as monitoring the status of the system. Developers will find it useful for defining and modifying system parameters, as well as locating, tracking, and analyzing messages.

The HLO Main Menu has three options:

```
  SM      HLO SYSTEM MONITOR
  MV      HLO MESSAGE VIEWER
  APPS    HLO APPLICATION REGISTRY
  STAT    HLO MESSAGE STATISTICS

Select HL7 (Optimized) MAIN MENU Option:
```

NOTE: Some of these options are protected by security keys and may or may not be displayed for certain users who have not been assigned those keys.

5.2 System Monitor

5.2.1 Overview

The System Monitor, a ListMan utility, is the primary utility for managing HLO. Within it, one can start and stop all processes and queues associated with HLO, as well as monitor all queues, links and processes. The System Monitor displays information appropriate to the selected option as well as a list of different actions that can be taken within the monitor. When the System Monitor is opened, the "Brief Status" screen is displayed. This is the same display shown when selecting action "BS" from within the System Monitor.

```
HLO SYSTEM MONITOR            Feb 14, 2005@14:27:23            Page:    1 of  1
 Brief Operational Overview                                                    >
SYSTEM STATUS:              RUNNING
PROCESS MANAGER:            RUNNING
STANDARD LISTENER:          OPERATIONAL
INTERFACE ENGINE:           NOT OPERATIONAL
TASKMAN:                    RUNNING
DOWN LINKS:                 0
CLIENT LINK PROCESSES:      0
IN-FILER PROCESSES:         0
MESSAGES PENDING TRANSMISSION:      0
STOPPED OUTGOING QUEUES:
MESSAGES PENDING ON APPLICATIONS:   0
STOPPED INCOMING QUEUES:
FILE 777 RECORD COUNT:              28     --> as of Feb 14, 2005@14:27
FILE 778 RECORD COUNT:              28     --> as of Feb 14, 2005@14:27



          Brief System Status
LP  LIST PROCESSES        BS  BRIEF STATUS          TL  TEST TCP LINK
DL  DOWN LINKS            ML  MONITOR LINK          RT  RealTime Mode
OQ  OUTGOING QUEUES           STOP HLO              SM  Scroll Mode
IQ  INCOMING QUEUES           START HLO             SQ  STRT/STP QUE
Select Action: Quit//
```

5.2.2 Actions:

LP – List Processes
This is an action protocol for displaying currently running processes and the system parameters governing those processes. Please refer to Appendix D for a complete overview of the HLO Process Registry.

```
HLO SYSTEM MONITOR              Feb 14, 2005@14:48:36           Page:    1  of  1
 Process Type                        MIN          MAX         #RUNNING     #QUEUED
PROCESS MANAGER                       1            1             1            0
CHECK PROCESS COUNTS                  0            1             0            0
OUTGOING CLIENT LINK                  1           10             1            0
INCOMING QUEUES                       1           10             1            0
PURGE OLD MESSAGES                    0            3             0            0
REMOVE BAD MESSAGES                   0            1             0            0
CLIENT MESSAGE UPDATES                1            2             1            0

$J: 14470    ->      OUTGOING CLIENT LINK  <- started at Feb 10, 2005@12:21:01
$J: 18556    ->         PROCESS MANAGER    <- started at Feb 10, 2005@12:21:01
$J: 20103    ->         INCOMING QUEUES    <- started at Feb 10, 2005@12:21:01
$J: 20105    ->     CLIENT MESSAGE UPDATES <- started at Feb 10, 2005@12:21:01

          Running Processes
LP  LIST PROCESSES        BS  BRIEF STATUS          TL  TEST TCP LINK
DL  DOWN LINKS            ML  MONITOR LINK          RT  RealTime Mode
OQ  OUTGOING QUEUES           STOP HLO              SM  Scroll Mode
IQ  INCOMING QUEUES           START HLO             SQ  STRT/STP QUE
Select Action: Quit//
```

The displayed parameters are:

- Process Type: The name of the process.
- MIN: The minimum number of processes for each process type that must be running on a properly functioning system.
- MAX: The maximum number of processes for each process type that can be running on a properly functioning system.
- #RUNNING: The number of processes for each process type that are currently running. When HLO is started, only the HLO PROCESS MANAGER process is started. It is the job of the HLO PROCESS MANAGER to start and manage the remaining HLO processes. The #RUNNING parameter should always be at least as large as the MIN parameter on a fully operational system and no larger than the MAX parameter.
- #QUEUED: The number of processes for each process type that are scheduled to be started. Usually this number will be zero.

In addition to showing process parameters, the System Monitor screen displays a listing of all currently running processes. Each line in the display shows the job number of a running HLO process, the type of process, and when it was started.

BS – Brief Status

This is an action protocol for displaying an overview of the current system.

```
HLO SYSTEM MONITOR            Feb 14, 2005@14:59:46            Page:    1 of  1
 Brief Operational Overview
SYSTEM STATUS:              RUNNING
PROCESS MANAGER:            RUNNING
STANDARD LISTENER:          OPERATIONAL
INTERFACE ENGINE:           NOT OPERATIONAL
TASKMAN:                    RUNNING
DOWN LINKS:                 0
CLIENT LINK PROCESSES:      0
IN-FILER PROCESSES:         0
MESSAGES PENDING TRANSMISSION:      0
STOPPED OUTGOING QUEUES:
MESSAGES PENDING ON APPLICATIONS:   0
STOPPED INCOMING QUEUES:
FILE 777 RECORD COUNT:              28     --> as of Feb 14, 2005@14:59:02
FILE 778 RECORD COUNT:              28     --> as of Feb 14, 2005@14:59:02
MESSAGES SENT TODAY:             63432
MESSAGES RECEIVED TODAY:          4318
           Brief System Status
LP  LIST PROCESSES        BS  BRIEF STATUS          TL  TEST TCP LINK
DL  DOWN LINKS            ML  MONITOR LINK          RT  RealTime Mode
OQ  OUTGOING QUEUES           STOP HLO              SM  Scroll Mode
IQ  INCOMING QUEUES           START HLO             SQ  STRT/STP QUE
Select Action: Quit//
```

The items displayed on this screen are:

- SYSTEM STATUS: (RUNNING or STOPPED). This indicates whether or not the HLO system is running. Note that when starting the system, it will show the status of "RUNNING" even if not all processes have started. Similarly, when stopping, it will display a status of "STOPPED", even if some processes have not yet responded to the STOP signal. When in doubt, it is always a good idea to select "LP" to make sure that all processes are running or stopped, depending on what you are trying to do.

- PROCESS MANAGER: (RUNNING or STOPPED). This indicates whether or not the HLO process manager is running. The process manager is required to be running to maintain and monitor the HLO processes.

- STANDARD LISTENER: (OPERATIONAL or NOT OPERATIONAL). This indicates whether or not the Standard Listener is running. The name of the Standard Listener must be added to the HLO SYSTEM PARAMETERS File (#779.1) in order for it to function properly. Please refer to Chapter 3 for more detailed instructions.

WARNING: If HLO is stopped or disabled for two hours or more and the TCPIP Services for Open VMS is being used, it should be disabled at the VMS level and then re-enabled before restarting HLO. Please refer to Section 4.3.6.9 for further instructions on enabling and disabling the TCPIP Services for Open VMS service.

- INTERFACE ENGINE: (OPERATIONAL or NOT OPERATIONAL). This indicates whether or not the interface engine link, "VA-VIE", can be opened. If "OPERATIONAL", then the Interface Engine is available; if not, then it is not available.

  NOTE: At the time this was written, the status of the Interface Engine will show as 'NOT OPERATIONAL'. This will continue until a new entry in the HL LOGICAL LINK named 'VA-VIE' is distributed in a future patch.

- TASKMAN: (OPERATIONAL or NOT OPERATIONAL). This indicates whether TaskMan is operational. TaskMan is required for the operation of HLO. If TaskMan is not operational, it must be started before HLO can be started properly.

- DOWN LINKS: Displays a count of non-functional links. This number should usually be zero.

- CLIENT LINK PROCESSES: Displays the number of CLIENT LINK processes currently running.

- IN-FILER PROCESSES: Displays the number of IN-FILER processes currently running.

- MESSAGES PENDING TRANSMISSION: Displays the total number of messages currently on the outbound queue waiting to be transmitted.

- STOPPED OUTGOING QUEUES: Displays a list of currently stopped outgoing queues. Only the first three queues are shown. If there are more than three, the line will end in "…".

- MESSAGES PENDING ON APPLICATIONS: Displays the total number of messages currently on the inbound queue waiting to be processed.

- STOPPED INCOMING QUEUES: Displays a list of currently stopped incoming queues. Only the first three queues are shown. If there are more than three, the line ends in "…".

- FILE 777 RECORD COUNT: Displays the number of records in the HLO MESSAGE BODY File (#777) as of the date and time the HLO RECORD COUNT process was run.

- FILE 778 RECORD COUNT: Displays the number of records in the HLO MESSAGES File (#778) as of the date and time the HLO RECORD COUNT process was run.

TL – Test Link

This is an action protocol for determining if a link is operational. User is prompted for a link name. Then the screen displays whether or not the link can be opened.

```
Select Action: Quit// TL   TEST TCP LINK
Select a TCP link: DAYTON OUT
DAYTON OUT IS operational...
Hit any key to continue...
```

DL – Down Links

This action takes the user to another screen that displays a list of currently failing links and links that have been shutdown, including the number of messages waiting to be transmitted on each link and the time the link was first marked as down. The entry is shown as 'SHUTDOWN' if the link was deliberately turned off. The screen contains the actions SHUTDOWN LINK and RESTART LINK for starting and stopping messages flowing out of a particular link.

```
HLO SYSTEM MONITOR            Feb 14, 2005@14:47:43           Page:    1 of  1
Client Link       Pending Messages          Date/Time Down
VACLE:5001                  33             Mar 23, 2005@15:48:51   SHUTDOWN
VAMAR:5001                1232             Mar 23, 2005@15:49:39









          Down Client Links
SL  SHUTDOWN LINK         RL  RESTART LINK
Select Action:Quit//
```

ML – Monitor Link

This is an action protocol for monitoring a link. User is asked for a link name. The protocol displays the number of messages pending transmission for that link. This display automatically refreshes itself periodically.

```
Select Action: Quit// ML   MONITOR LINK
Select a TCP link: DAYTON OUT
Hit any key to stop...
MESSAGES PENDING TRANSMISSION:          0
```

OQ – Outgoing Queues

This is an action protocol for listing all outgoing queues with pending messages and the current number of messages waiting in each queue. Listed links with an asterisk ("*") next to them are down. Listed links with an exclamation mark ("!") next to them are stopped.

```
HLO SYSTEM MONITOR            Feb 14, 2005@14:59:46          Page:    1 of    1
  Link                       Queue                    Count
 VACLE:5001                  DEFAULT                     33
*VAMAR:5001                  DEFAULT                   1232
!VAROA:5001                  DEFAULT                    374









                     Outgoing Queues *down links !stopped queues
LP  LIST PROCESSES        BS  BRIEF STATUS          TL  TEST TCP LINK
DL  DOWN LINKS            ML  MONITOR LINK          RT  RealTime Mode
OQ  OUTGOING QUEUES           STOP HLO              SM  Scroll Mode
IQ  INCOMING QUEUES           START HLO             SQ  STRT/STP QUE
Select Action: Quit//
```

IQ – Incoming Queues

This is an action protocol for listing all incoming queues with pending messages and the current number of messages waiting in each queue. Entries with an exclamation mark ("!") next to them are stopped queues.

```
HLO SYSTEM MONITOR            Feb 14, 2005@14:59:46          Page:    1 of    1
From                                        Queue                      Count
151~FELIX.DAOU.COM:5001~DNS                !DEFAULT                       21
152~DSI-ALPHA.DAOU.COM:5001~DNS             DEFAULT                      227









          Incoming Queues ('!' = stopped queues)
LP  LIST PROCESSES        BS  BRIEF STATUS          TL  TEST TCP LINK
DL  DOWN LINKS            ML  MONITOR LINK          RT  RealTime Mode
OQ  OUTGOING QUEUES           STOP HLO              SM  Scroll Mode
IQ  INCOMING QUEUES           START HLO             SQ  STRT/STP QUE
Select Action: Quit//
```

SQ – Start/Stop Queue

This is an action protocol to start or stop the processing of a selected queue. The user is prompted to:

1. Start or stop the queue
2. Determine if the queue is incoming or outgoing
3. Enter the full name of the queue

WARNING: There is no verification for the queue name. The user must make sure that the name of an existing queue is entered correctly and in full, at the "Enter the name of queue:" prompt.

```
Select Action: Quit// SQ   STRT/STP QUE

Select one of the following:

    1         START
    2         STOP

Do you want to START or STOP a queue: 1// 2  STOP

Select one of the following:

    I         INCOMING
    O         OUTGOING

Do you want to stop an incoming queue or an outgoing queue: I// <RET> NCOMING
Enter the name of queue:
```

STOP HLO

This is an action protocol for stopping HLO. The process manager sends a request to all currently running processes to stop. Each process responds to the request when it sees it. Some processes will stop immediately and others will stop once they have completed their current task. Use the action "LP" to check for processes that are still running after using this action.

WARNING: If HLO is stopped or disabled for two hours or more, the TCPIP Service for Open VMS should be disabled at the VMS level and then re-enabled before restarting HLO. Please refer to Section 4.3.6.9 for further instructions on enabling and disabling the TCPIP Services for Open VMS service.

START HLO

This is an action protocol for starting HLO. When HLO is started, the HLO Process Manager is also started. The HLO Process Manager then starts all the other processes. It will usually take a few seconds for the processes to start. Use the "LP" action to make sure that all necessary processes have started before checking for any results.

WARNING: TaskMan must be running for HLO to start properly.

Q – Quit
This is an action protocol for exiting the System Monitor.

RT – Real Time Mode
This is an action protocol that refreshes whichever screen is currently displayed in "Real Time" mode. The current display is refreshed periodically, with the updated display appearing in the same position as the previous one.

SM – Scroll Mode
This is an action protocol that refreshes whichever screen is currently displayed in "Scroll" mode. The current display is refreshed periodically, with the updated display appearing after the previous one. This allows users to see both the current screen and several previous screens, depending on the number of iterations of the screen that can fit within the terminal display.

5.3 Message Viewer

5.3.1 Overview

The Message Viewer allows users to review individual messages and various error reports. The initial Message Viewer screen is shown below.

```
HLO MESSAGE VIEWER            Aug 03, 2005@09:57:23          Page:    0 of    0
  MsgID          MsgType  Dt/Tm                Error Text
______________________________________________________________________________

_____________________________________________________________________________
           Enter ?? for more actions
_____________________________________________________________________________
  DM  DISPLAY MSG           AE  APPLICATION ERRORS         MS  MESSAGE SEARCH
  SE  SYSTEM ERRORS         TF  TRANSMISSION FAILURES
Select Action: Quit//
Select Action: Next Screen//
```

5.3.2 Actions

DM – Display Message
This is an action protocol for displaying incoming or outgoing messages. The Message ID is required to perform this action. The user is prompted for a message ID (the string contained in MSH-10). The content of the message is displayed with the message ID.

When a message is displayed, there are several items worth noting:

- Detailed administrative information about the message can be found at the top of the display screen.
- The message text can be found in the middle of the display screen.
- Message segments that are too long to display on just one line will wrap to the next line. Segments requiring additional display lines will be denoted with a reverse video dash placed in front of it (“-“). These specially displayed characters are not part of the actual message. There are three examples of this in the sample displayed below.
- Many messages will require more than one display screen to view the entire message. As with the message below, additional screens can be accessed by selecting the “Next Screen” default when not on the last display screen.

```
Single Message Display        Aug 03, 2005@09:59:39          Page:    1 of    1
_____________________________________________________________________________
                            Administrative Information
MsgID: 151 824           Status: SU
Dir:   OUTGOING                        Trans Dt/Tm: 2/12/05@17:11:04
Link:  TIGER OUT                          Queue:       DEFAULT
Accept Ack: 151 824                    At:          2/12/05@17:11:04
   MSA|CA|151 824|||||
Accept Ack Rtn: n/a

                                   Message Text
MSH|^~\&|HLOZ MFN|151^VAABC.MED.VA.GOV:5001^DNS|HLOZMFN|^VAXYZ.MED.VA.GOV:5001^
-DNS|20050212101628-0500||ADT^A08^|151 824|T^|2.4|||AL|NE|USA
EVN|A08|20050212101609-0500
PID||63324|999058704|369-2958097|GROZENSO^TED^M||19571230|M||||||||||101058704
PD1|8|3|||5|||||||N
PV1||O|1425|||||||||||||||||||14|||||||||||||||||||200402031230-0500|2004020
-31230-0500
IN1|1||1348|XYZ FOUNDATION FOR MEDICAL CARE|PO BOX 909^^TUSCAN^AZ^12345^USA||||
-L'AUBERGE|||20010601|20030131||40||||||1|||||||||20020226
IN3|1|1|||||||||||||||||||||1
+         Enter ?? for more actions                                         _

DM  DISPLAY MSG           AE  APPLICATION ERRORS    MS  MESSAGE SEARCH
SE  SYSTEM ERRORS         TF  TRANSMISSION FAILURES
```

AE – Application Errors

This is an action protocol for displaying messages for which HLO received application errors (negative application acknowledgements). Application errors can be viewed from a specified date to the present and for a specific application.

```
Select Action: Quit// AE   APPLICATION ERRORS
Enter the beginning date:  T-1// <RET> (MAR 10, 2005)
Include ALL applications? YES// <RET>
```

Information in the Application Errors screen includes:

- Header Line 1, including:
  - Current date and time.
  - Current page number of the results.
  - Total page count of the results.
- Header Line 2, consisting of data column labels, including:
  - MsgID – Message ID, consisting of the site station number and a sequential number, as stored in the querying system.
  - MsgType – Message and Event Type (separated by "~")
  - Dt/Tm – Message Date and Time (date and time the message was stored for either sending or receiving).
  - Error Text – Brief error description.

- Data Line 1, including:
  - Application – Application Name, based on an application from the HLO APPLICATION REGISTRY File (#779.2).
- Data Line 2, including:
  - Message ID information.
  - Message and Event type.
  - Message Date and Time.
  - Error Text.

Below is an example of a system error display. To display more information about the message, select the "DM" action protocol.

```
HLO MESSAGE VIEWER            Mar 11, 2005@17:22:54            Page:    1 of  1
  MsgID          MsgType  Dt/Tm                 Error Text
Application: HLOZ NMQ
  151 7499       NMQ~N01 3/10/05@23:43:27   INVALID EVENT CODE

           Enter ?? for more actions
DM  DISPLAY MSG             AE  APPLICATION ERRORS          MS MESSAGE SEARCH
SE  SYSTEM ERRORS           TF  TRANSMISSION FAILURES
Select Action: Quit//
```

MS – Message Search

This is an action protocol for querying existing incoming or outgoing HLO message queues. Message search queries are based on six criteria:

Criteria	Default	Required?	Description
Beginning Date	T-1 (yesterday)	Yes	The earliest date for which messages should be selected.
Ending Date/Time	NOW (current date/time)	Yes	The latest date/time for which messages should be selected.
Application	NONE	No	Name of a specific application from the HLO APPLICATION REGISTRY File (#779.2).
Message Type	NONE	No	Three character message type code.
Event Type	NONE	No	Three character event type code.
Incoming or Outgoing	NONE	Yes	"I" for incoming messages "O" for outgoing messages

Selection will be based on all messages transmitted or received from the beginning date to the selected ending date and time. A blank entry for Application, Message Type, or Event Type will cause all entries to be selected regardless of the value in that field.
The six query criteria are presented to the user in a scrolling format at the bottom of the message viewer screen. Prompting begins after the user has selected the action "MS." Refer to the screen below for an example.

```
HLO MESSAGE VIEWER            Mar 20, 2005@14:31:16          Page:    0 of    0
  MsgID          MsgType  Dt/Tm                Error Text                      _




            Enter ?? for more actions                                          _
DM  DISPLAY MSG           AE  APPLICATION ERRORS    MS  MESSAGE SEARCH
SE  SYSTEM ERRORS         TF  TRANSMISSION FAILURES
Select Action: Quit// MS   MESSAGE SEARCH
Enter the beginning date:  Mar 19, 2005// <RET> (MAR 19, 2005)
Enter the ending date/time: NOW// <RET> (MAR 20, 2005@14:31:33)
Application: XYZ VISTA
HL7 Message Type: ADT
HL7 Event: A08

     Select one of the following:

          I         INCOMING
          O         OUTGOING
Incoming or Outgoing: OUTGOING
```

After the message search parameters are entered, all matching messages are listed in the Message Search result screen (see example below). Information in the results screen includes:

- Header Line 1, including:
  - Current date and time.
  - Current page number of the results.
  - Total page count of the results.
- Header Line 2, consisting of data column labels, including:
  - MsgID – Message ID, consisting of the site station number and a sequential number, as stored in the querying system.
  - Application – Application Name, based on an application from the HLO APPLICATION REGISTRY File (#779.2).
  - MsgType – Message and Event Type (separated by "~").
- Header Line 3, consisting of data column labels, including:
  - Dt/Tm – Message Date and Time (date and time the message was stored for either sending or receiving.
  - Facility – Information for the facility either sending the message (for incoming messages) or receiving the message (for outgoing messages).
- Data Line 1, including:
  - Message ID information.
  - Application name.
  - Message and Event type.

- Data Line 2, including:
  - Message date and time.
  - Sending or Receiving facility.

At the bottom of each result screen is the "Select Action" prompt. If the displayed page is not the last page, the default action will be "Next Screen," which displays the next screen of results. If the displayed page is the last page, the default action is "Quit", which takes the user back to the original Message Viewer screen (see the next two example screens).

```
Message Search                Mar 20, 2005@14:31:57          Page:    1 of    1
MsgID                      Application                    MsgType
     Dt/Tm                 Facility
151 94                     XYZ VISTA                       ADT~A08
     3/19/05@10:16:12      ^VAXYZ.VA.MED.GOV:5001^DNS
151 95                     XYZ VISTA                       ADT~A08
     3/19/05@10:16:12      ^VAXYZ.VA.MED.GOV:5001^DNS
151 96                     XYZ VISTA                       ADT~A08
     3/19/05@10:16:12      ^VAXYZ.VA.MED.GOV:5001^DNS
151 97                     XYZ VISTA                       ADT~A08
     3/19/05@10:16:12      ^VAXYZ.VA.MED.GOV:5001^DNS
151 98                     XYZ VISTA                       ADT~A08
     3/19/05@10:16:12      ^VAXYZ.VA.MED.GOV:5001^DNS
151 99                     XYZ VISTA                       ADT~A08
     3/19/05@10:16:12      ^VAXYZ.VA.MED.GOV:5001^DNS
+         Enter ?? for more actions
DM  Display Message
Select Action: Next Screen// <RET>
```

```
Message Search                Mar 20, 2005@14:32:12          Page:   2  of   2
MsgID                      Application                    MsgType
+    Dt/Tm                 Facility
151 247                    XYZ VISTA                       ADT~A08
     3/19/05@10:16:14      ^VAXYZ.VA.MED.GOV:5001^DNS
151 248                    XYZ VISTA                       ADT~A08
     3/19/05@10:16:14      ^VAXYZ.VA.MED.GOV:5001^DNS
151 249                    XYZ VISTA                       ADT~A08
     3/19/05@10:16:14      ^VAXYZ.VA.MED.GOV:5001^DNS
151 250                    XYZ VISTA                       ADT~A08
     3/19/05@10:16:14      ^VAXYZ.VA.MED.GOV:5001^DNS




+         Enter ?? for more actions
DM  Display Message
Select Action: Quit// <RET>
```

To display a message selected by the message search, use the action of "DM" (Display Message). After entering "DM," enter the message ID associated with the message to be displayed (see example below).

```
Message Search                Mar 20, 2005@14:31:57          Page:     1 of   2
MsgID                      Application                    MsgType
     Dt/Tm                 Facility
151 94                     XYZ VISTA                       ADT~A08
     3/19/05@10:16:12      ^VAXYZ.VA.MED.GOV:5001^DNS
151 95                     XYZ VISTA                       ADT~A08
     3/19/05@10:16:12      ^VAXYZ.VA.MED.GOV:5001^DNS
151 96                     XYZ VISTA                       ADT~A08
     3/19/05@10:16:12      ^VAXYZ.VA.MED.GOV:5001^DNS
151 97                     XYZ VISTA                       ADT~A08
     3/19/05@10:16:12      ^VAXYZ.VA.MED.GOV:5001^DNS
151 98                     XYZ VISTA                       ADT~A08
     3/19/05@10:16:12      ^VAXYZ.VA.MED.GOV:5001^DNS
151 99                     XYZ VISTA                       ADT~A08
     3/19/05@10:16:12      ^VAXYZ.VA.MED.GOV:5001^DNS
+         Enter ?? for more actions
DM  Display Message
Select Action: Next Screen// DM   Display Message
Message ID: 151 97
```

Once a message ID has been entered, the Single Message Display screen appears. The screen consists of:

- Administrative Information, including:
  - MsgID – Message ID, consisting of the site station number and a sequential number, as stored in the querying system.
  - Status – Message Status (Successful of Unsuccessful)
  - Dir – Message direction (Incoming or Outgoing)
  - Trans Dt/Tm – Transmission Date and Time
  - Link: Port Number – Sending (for incoming) or Receiving (for outgoing) link followed by a colon and the port number of the link
  - Queue – Message queue
  - Accept Ack – Accept Ack information
  - At – Date and Time Ack received
  - Accept Ack Rtn – Optional routine call for responding to an Accept Ack. (Please see Section 6.1.4 for more details on creating the routine)
- Message Text – Segment-by-segment display of the message. Segments requiring more than one display line are denoted with a reverse video dash placed in front of it (-). These specially displayed characters are not part of the actual message.

In cases where a message requires more than one screen, the user will be prompted to go to the next screen with a default Select Action of "Next Screen." If the Single Message Display screen contains the end of the message, the default Select Action will be "Quit," which will return the user back to the query results. The Select Action of "Quit" can be used at any time to return to the previous activity.

Please refer to the next three sample screens as an example of how the message display works within a message search.

First Message Display Screen

```
Single Message Display        Mar 20, 2005@14:35:18          Page:    1 of    2
                                Administrative Information
MsgID: 151 97             Status: SU
Dir:   OUTGOING                          Trans Dt/Tm: 3/19/05@10:16:57
Link:  VAXYZ                            Queue:       DEFAULT
Accept Ack: 151 97                       At:          3/19/05@10:16:57
   MSA|CA|151 97|||||
Accept Ack Rtn: n/a

                                     Message Text
MSH|^~\&|XYZ VISTA|151^VAABC.MED.VA.GOV:5001^DNS|XYZ HDM|^VAXYZ.MED.VA.GOV:5001^
-DNS|20050319-101612-0500||ADT^A08^|151 97|T^|2.4|||AL|NE|USA
EVN|A08|20050319101529-0500
PID||55126|999074037|649-2959460|ZEAL^ROBERT^U||19350709|M||||||||||101074037
PD1|5||||5|||||||Y
PV1||O|68|||||||||||||||||||14|||||||||||||||||||||200405071030-0500|2004050710
-30-0500
IN1|1||1156|MEDICARE (WNR)|PO BOX ^844 N MAIN ST^PHOENIX^MD^20765-5520^USA||||PA
-RT A|||20000701|||33||||||1|||||||20030709
+         Enter ?? for more actions

Select Action: Next Screen// <RET>
```

Second Message Display Screen

```
Single Message Display        Mar 20, 2005@14:35:36          Page:    2 of    2
+
IN3|1|1|||||||||||||||||||||0
ZN1|1|WITHIN 1YR FROM DOS|
IN3|2|2|||||||||||||||||||||0
ZN1|1|WITHIN 1YR FROM DOS|
IN3|3|3|||||||||||||||||||||0
ZN1|1|WITHIN 1YR FROM DOS|
IN3|4|4|||||||||||||||||||||0
ZN1|1|WITHIN 1YR FROM DOS|
IN3|5|5|||||||||||||||||||||0
ZN1|1|WITHIN 1YR FROM DOS|
Z3M|1|^N|^N|^N|^N|^N||^N




+        Enter ?? for more actions

Select Action: Quit// <RET>
```

Back to Message Search Result Screen After Selecting "Quit"

```
Message Search                Mar 20, 2005@14:38:12        Page:     1 of   28
MsgID                      Application                    MsgType
     Dt/Tm                 Facility
151 94                     XYZ VISTA                       ADT~A08
     3/19/05@10:16:12      ^VAXYZ.VA.MED.GOV:5001^DNS
151 95                     XYZ VISTA                       ADT~A08
     3/19/05@10:16:12      ^VAXYZ.VA.MED.GOV:5001^DNS
151 96                     XYZ VISTA                       ADT~A08
     3/19/05@10:16:12      ^VAXYZ.VA.MED.GOV:5001^DNS
151 97                     XYZ VISTA                       ADT~A08
     3/19/05@10:16:12      ^VAXYZ.VA.MED.GOV:5001^DNS
151 98                     XYZ VISTA                       ADT~A08
     3/19/05@10:16:12      ^VAXYZ.VA.MED.GOV:5001^DNS
151 99                     XYZ VISTA                       ADT~A08
     3/19/05@10:16:12      ^VAXYZ.VA.MED.GOV:5001^DNS
+         Enter ?? for more actions
DM  Display Message
Select Action: Next Screen//
```

SE – System Errors

This is an action protocol for displaying messages for which HLO has received system errors (negative commit acknowledgements). System errors can be viewed from a specified date to the present and for a specific application.

```
Select Action: Quit// SE   SYSTEM ERRORS
Enter the beginning date:  T-1// <RET> (MAR 02, 2005)
Include ALL applications? YES// <RET>
```

Information in the System Error screen includes:

- Header Line 1, including:
  - Current date and time.
  - Current page number of the results.
  - Total page count of the results.
- Header Line 2, consisting of data column labels, including:
  - MsgID – Message ID, consisting of the site station number and a sequential number, as stored in the querying system.
  - MsgType – Message and Event Type (separated by "~").
  - Dt/Tm – Message Date and Time (date and time the message was stored for either sending or receiving).
  - Error Text – Brief error description.
- Data Line 1, including:
  - Application – Application Name, based on an application from the HLO APPLICATION REGISTRY File (#779.2).
- Data Line 2, including:
  - Message ID information.
  - Message and Event type.
  - Message Date and Time.
  - Error Text.

The following graphic is an example of a system error display. To display more information about the message, use the "DM" option as explained in the previous section.

```
HLO MESSAGE VIEWER           Mar 03, 2005@12:47:18           Page:    1 of  1
  MsgID          MsgType  Dt/Tm                Error Text
Application: HLOZ NMQ
  151 2001       NMQ~N01 3/2/05@11:54:38   RECEIVING APPLICATION NOT DEFINED









          Enter ?? for more actions
DM  DISPLAY MSG            AE  APPLICATION ERRORS         MS  MESSAGE SEARCH
SE  SYSTEM ERRORS          TF  TRANSMISSION FAILURES
Select Action: Quit//
```

TF – Transmission Failures

This is an action protocol for displaying messages for which HLO received transmission failures. Transmission failures can be viewed from a specified date to the present and for a specific application.

```
Select Action: Quit// TF   TRANSMISSION FAILURES
Enter the beginning date:  T-1// <RET> (MAR 09, 2005)
Include ALL applications? YES// <RET>
```

Information in the Transmission Failure screen includes:

- Header Line 1, including:
  - Current date and time.
  - Current page number of the results.
  - Total page count of the results.
- Header Line 2, consisting of data column labels, including:
  - MsgID – Message ID, consisting of the site station number and a sequential number, as stored in the querying system.
  - MsgType – Message and Event Type (separated by "~").
  - Dt/Tm – Message Date and Time (date and time the message was stored for either sending or receiving).
  - Error Text – Brief error description.
- Data Line 1, including:
  - Application – Application Name, based on an application from the HLO APPLICATION REGISTRY File (#779.2).
- Data Line 2, including:
  - Message ID information.
  - Message and Event type.

- Message Date and Time.
- Error Text.

The following graphic is an example of a transmission failure screen. To display more information about the message, use the "DM" option.

```
HLO MESSAGE VIEWER            Mar 10, 2005@08:22:10            Page:    1 of  1
  MsgID          MsgType  Dt/Tm                  Error Text
Application: HLOZ NMQ
  151 8767       NMQ~N01 3/9/05@07:12:21       FAILED TRANSMISSIONS




          Enter ?? for more actions
DM  DISPLAY MSG             AE  APPLICATION ERRORS          MS MESSAGE SEARCH
SE  SYSTEM ERRORS           TF  TRANSMISSION FAILURES
Select Action: Quit//
```

Q – Quit
This is an action protocol for exiting the Message Viewer.

5.4 Application Registry (HLO)

5.4.1 Overview

Application developers must create entries in the application registry as part of building a new HLO messaging application. Application registries are usually required in conjunction with the installation of a new or updated messaging application that is using HLO.

WARNING: This option is for the use of application developers only. IRMs should not normally be accessing this option unless directed by a developer. Refer to Chapter 6 for additional information.

5.5 TaskMan-Scheduled Options

In HLO, there are two TaskMan-scheduled options not listed on the main HLO menu: the HLO COUNT RECORDS option and the HLO SYSTEM STARTUP option.

5.5.1 HLO COUNT RECORDS

Please refer to Section 3.5 for detailed instructions on scheduling this option.

5.5.2 HLO SYSTEM STARTUP

Please refer to Section 3.6 for detailed instructions on scheduling this option.

5.6 HLO MESSAGE STATISTICS REPORT

This report provides counts of messages by application over a period selected by the user. The user is given a choice of reporting either monthly, daily, or hourly statistics. Hourly statistics are generally maintained only for the last 48 hours, daily statistics are available for the last 31 days, and monthly statistics are kept indefinitely.

6.0 HLO Application Development

The following steps are needed to create applications in HLO:

1. Develop an Application
2. Configure a Link for the Client Application.
3. Update the Outgoing Client Link Process
4. Application Registration

6.1 Develop an Application

During the development of an HLO application, the developer may need to consider the following:

- HLO APIs
- Outgoing Messages
- Incoming Messages
- Accept Acknowledgements
- Error messages from API calls
- Queue Management

6.1.1 HLO APIs

Some basic APIs have been developed to assist the developer in creating HL7 messages for HLO. Detailed information on each HLO API can be found in Appendix B. Each API that is documented for developer use also includes input and output parameters. If a parameter isn't specifically documented as being passed by reference, then it is to be passed by value.

Below is a brief list and summary.

HLO API	**Brief Description**
Building Messages	
$$NEWMSG^HLOAPI	Begins a new message.
$$NEWBATCH^HLOAPI	Begins a new batch of messages.
$$ADDMSG^HLOAPI	Begins a new message within a batch.
SET^HLOAPI	Builds a message segment.
$$ADDSEG^HLOAPI	Adds a segment to a message.
$$MOVEMSG^HLOAPI	Moves a message built by a traditional pre-HLO message builder to HLO.
Inserting Data Types	
SETTS^HLOAPI4	Sets a timestamp into a segment.
SETDT^HLOAPI4	Sets a date into a segment.
SETCE^HLOAPI4	Sets a coded element into a segment.
SETHD^HLOAPI4	Sets an HL7 hierarchic designator into a segment.
SETCNE^HLOAPI4	Sets a coded value with no exceptions into a segment.
SETCWE^HLOAPI4	Sets a coded value with exceptions into a segment.
SETAD^HLOAPI4	Sets an address into a segment.
Sending Messages	
$$SENDONE^HLOAPI1	Sends messages to a receiving application.
$$SENDMANY^HLOAPI1	Sends messages to multiple receiving applications.
$$SENDSUB^HLOAPI1	Sends messages to subscribers.
Receiving Messages	
$$STARTMSG^HLOPRS	Initiates message parsing, returns header values.
$$NEXTSEG^HLOPRS	Advances to and parses next segment.
$$NEXTMSG^HLOPRS	Advances to next message, returns header.
$$PARSE^HLOPRS1	Called by NEXTSEG, returns parsed values.
$$GET^HLOPRS	Returns specific segment components.
Parsing Data Types	
GETTS^HLOPRS2	Gets a timestamp from a segment.
GETDT^HLOPRS2	Gets a date from a segment.
GETCE^HLOPRS2	Gets a coded element from a segment.

GETHD^HLOPRS2	Gets an HL7 hierarchic designator from a segment.
GETCNE^HLOPRS2	Gets a coded value with no exceptions from a segment.
GETCWE^HLOPRS2	Gets a coded value with exceptions from a segment.
GETAD^HLOPRS2	Gets an address from a segment.

Application Acknowledgements

$$ACK^HLOAPI2	Initiates an application acknowledgement.
$$SENDACK^HLOAPI2	Sends an application acknowledgement.
$$BATCHACK^HLOAPI3	Begins creation of batch acknowledgement.
$$ADDACK^HLOAPI3	Adds an application acknowledgement to a batch.

Subscription Registries

$$CREATE^HLOASUB	Creates a subscription registry entry.
$$ADD^HLOASUB	Adds a new recipient to a subscription registry.
$$END^HLOASUB	Terminates a recipient from a subscription registry.
$$ONLIST^HLOASUB	Determines if a recipient is on a subscription list.
$$NEXT^HLOASUB	Goes to next recipient on subscription list.
$$INDEX^HLOASUB1	Builds an index of subscription list recipients.
$$FIND^HLOASUB1	Finds a specific subscription list.

HL 1.6 to HLO Message Conversion

$$EN^HLOCNRT	Converts HL 1.6 message to HLO and send.
APAR^HLOCVU	Converts HL 1.6 parameters to HLO.
$$HLNEXT^HLOMSG	Returns segment as a set of lines stored in SEG.

Queue Management

STARTQUE^HLOQUE	Set a “Stop” flag on a specific HLO queue.
STOPQUE^HLOQUE	Remove a “Stop” flag on a specific HLO queue.
$$STOPPED^HLOQUE	Check the status of a queue (started or stopped).

Miscellaneous

$$RESEND^HLOAPI3	Retransmits an outgoing message
$$REPROC^HLOAPI3	Reprocesses an incoming message
$$SETPURGE^HLOAPI3	Resets purge date of a currently stored message

For a complete description of all HLO APIs, refer to Appendix B.

6.1.2 Outgoing Messages

For sending messages, HLO APIs include functionality to:

- Send a single message.
- Send a batch message.
- Send a message or messages to multiple recipients.
- Send a batch message or messages to multiple recipients.
- Convert Existing HL 1.6 Outgoing Messages to HLO Messages.

The following sections contain sample code to demonstrate this functionality. For detailed information on the API calls used in the sample code, refer to Appendix B.

6.1.2.1 Send a single message

The most basic function is to send a single message. To build and send a single message, the developer needs to follow these steps

1. Define message parameters (APPARMS array).
2. Create new message ($$NEWMSG^HLOAPI).
3. Create segments (SET^HLOAPI).
4. Add segments to the message ($$ADDSEG^HLOAPI).
5. Define sending and receiving parameters. (APPARMS and WHO arrays).
6. Send the message ($$SENDONE^HLOAPI1).

Below is a sample routine written to generate and send a single HL7 message:

NOTE: In the sample routine below, the first four parameters: APPARMS("COUNTRY"), APPARMS("FIELD SEPARATOR"), APPARMS("ENCODING CHARACTERS"), and APPARMS("VERSION") are not actually required as these are the default values. They are shown here as an example of how to define them when needed. In subsequent sample routines, these parameters will not be declared but the default values will still apply.

```
HLOSNGL      ; Single Messages
       ; Define message parameters
       S APPARMS("COUNTRY")="USA"
       S APPARMS("FIELD SEPARATOR")="|"
       S APPARMS("ENCODING CHARACTERS")="^~\&"
       S APPARMS("VERSION")=2.4
       ;
       S APPARMS("MESSAGE TYPE")="NMQ"
       S APPARMS("EVENT")="N01"
       ; Create new message
       I '$$NEWMSG^HLOAPI(.APPARMS,.HLMSTATE,.ERROR) W !,ERROR Q
       ;
       ; Create segments
       D SET^HLOAPI(.SEG,"SFT",0)
       D SET^HLOAPI(.SEG,."VistA",1)
       D SET^HLOAPI(.SEG,"2.0T3",2)
       D SET^HLOAPI(.SEG,"HL7",3)
       D SET^HLOAPI(.SEG,"43",4)
       D SET^HLOAPI(.SEG,$$HLDATE^HLFNC($$NOW^XLFDT(),"TS"),6)
       ; Add segments to the message
       I '$$ADDSEG^HLOAPI(.HLMSTATE,.SEG) Q
       ;
       ; Create segments
       D SET^HLOAPI(.SEG,"QRD",0)
       D SET^HLOAPI(.SEG,$$HLDATE^HLFNC($$NOW^XLFDT(),"TS"),1)
       D SET^HLOAPI(.SEG,"R",2)
       D SET^HLOAPI(.SEG,"I",3)
       D SET^HLOAPI(.SEG,1,4)
       D SET^HLOAPI(.SEG,"RD",7,2)
```

```
D SET^HLOAPI(.SEG,1,7,1)
D SET^HLOAPI(.SEG,"NCK",9)
D SET^HLOAPI(.SEG,"S",12)
; Add segments to the message
I '$$ADDSEG^HLOAPI(.HLMSTATE,.SEG) Q
;
;Create segments
D SET^HLOAPI(.SEG,"NCK",0)
D SET^HLOAPI(.SEG,$$HLDATE^HLFNC($$NOW^XLFDT(),"TS"),1)
; Add segments to the message
I '$$ADDSEG^HLOAPI(.HLMSTATE,.SEG) Q
;
; Define sending and receiving parameters
S APPARMS("SENDING APPLICATION")="HLO NMQ"
S APPARMS("ACCEPT ACK TYPE")="AL"
S APPARMS("APP ACK RESPONSE")="APACK^HLOZTST1"
S APPARMS("ACCEPT ACK RESPONSE")="MSG^HLOZTST"
S APPARMS("APP ACK TYPE")="AL"
S WHO("RECEIVING APPLICATION")="HLO NMQ"
S WHO("FACILITY LINK NAME")="LINK OUT"
;
; Send the message
I '$$SENDONE^HLOAPI1(.HLMSTATE,.APPARMS,.WHO,.ERROR) W !,ERROR Q
Q
```

WARNING: The facility link name defined in parameter WHO("FACILITY LINK NAME") must be a valid entry from the HL LOGICAL LINK File (#870). A valid entry for HLO must have the fields DNS DOMAIN (#.08) and TCP/IP PORT (OPTIMIZED) (#400.08) populated. Failure to use a valid entry from this file will result in a "DOMAIN NOT FOUND" error.

NOTE: Queues - Important Information about HLO Queues

- Queues are automatically created with a default name of "DEFAULT".
- Developers are permitted to identify uniquely named queues by utilizing the APPARMS("QUEUE") parameter when generating outgoing messages (refer to 'Sending Messages' in Appendix B). It is recommended to use namespace conventions to define queues.
- Uniquely naming a queue has several advantages, including:
    - A shorter queue, causing:
        - Improved concurrency
        - Faster processing
    - The ability to stop the queue without interfering with other applications

6.1.2.2 Send a batch message

Batch messages allow the transmission of a group of messages within a single message structure. An example of this could be a table (or file) of similar records. To build and send a batch message, the developer needs to follow these steps:

1. Define message parameters (APPARMS array).
2. Create new batch message ($$NEWBATCH^HLOAPI).
    a. Create messages within the batch ($$ADDMSG^HLOAPI).
        i. Create segments within each message ($$ADDSEG^HLOAPI).
3. Define sending and receiving parameters (APPARMS and WHO arrays).
4. Send the batch message ($$SENDONE^HLOAPI1).

Below is a sample routine written to generate and send a batch HL7 message:

```
HLOBATCH     ; Batch Messages
       ; Define message parameters
       ;{Declare additional parameters}
       ;
       ; Create new batch message
       I '$$NEWBATCH^HLOAPI(.APPARMS,.HLMSTATE,.ERROR) W !,ERROR Q
       ;
       ;  Create a MFN message for each message in the batch.
       S CT=0
       ; (Create messages for files 11,12, and 13)
       F FILE=11,12,13 D MSG
       I $G(ERR)'="" W !,ERR Q
       ;
       ; Define sending and receiving parameters
       S APPARMS("SENDING APPLICATION")="HLO MFN"
       S APPARMS("APP ACK RESPONSE")="NE"
       S WHO("RECEIVING APPLICATION")="HLO MFN"
       S WHO("FACILITY LINK NAME")="LINK OUT"
       ;
       ; Send the batch message
       I '$$SENDONE^HLOAPI1(.HLMSTATE,.APPARMS,.WHO,.ERROR) W !,ERROR Q
       Q
       ;
MSG    ;  Create messages within the batch
       S APPARMS("EVENT")="M13"
       S APPARMS("MESSAGE TYPE")="MFN"
       I '$$ADDMSG^HLOAPI(.HLMSTATE,.APPARMS,.ERROR) S ERR=ERROR Q
       ;
SEG    ;  Create segments within each message
       D SET^HLOAPI(.SEG,"MFI",0)
       D SET^HLOAPI(.SEG,FILE,1)
       D SET^HLOAPI(.SEG,"UPD",3)
       D SET^HLOAPI(.SEG,"NE",6)
       I '$$ADDSEG^HLOAPI(.HLMSTATE,.SEG) S ERR="Segment not added" Q
       ;
       ;  For each record in the file, create segments within each message
       S GLREF=$$ROOT^DILFD(FILE,"",1)
```

```
       S IEN=0
       F  S IEN=$O(@GLREF@(IEN)) Q:'IEN  D
       . D SET^HLOAPI(.SEG,"MFE",0)
       . D SET^HLOAPI(.SEG,"MUP",1)
       . D SET^HLOAPI(.SEG,$$HLDATE^HLFNC($$DT^XLFDT(),"DT"),3)
       . D SET^HLOAPI(.SEG,IEN,4,1)
       . D SET^HLOAPI(.SEG,$P(@GLREF@(IEN,0),"^",1),4,2)
       . D SET^HLOAPI(.SEG,"CE",5)
       . I '$$ADDSEG^HLOAPI(.HLMSTATE,.SEG) S ERR="MFE not added" Q
       Q
```

6.1.2.3 Send a Single Message to Multiple Recipients

Sending a single message to more than one recipient is not much different than sending to one recipient. To build and send a single message to multiple recipients, the developer needs to follow these steps:

1. Define message parameters (APPARMS array).
2. Create new message ($$NEWMSG^HLOAPI).
3. Create segments (SET^HLOAPI).
4. Add segments to the message ($$ADDSEG^HLOAPI).
5. Define the sending and receiving parameters for all recipients (APPARMS and WHOTO array).
6. Send the message to all recipients ($$SENDMANY^HLOAPI1).

Below is a sample routine written to generate and send a single HL7 message to multiple recipients:

```
HLOTOMNY     ; Send message to multiple recipients.
       ;
       ; Define message parameters.
       ;{Declare additional parameters}
       ;
       S APPARMS("EVENT")="N01"
       S APPARMS("MESSAGE TYPE")="NMQ"
       ;
       ; Create new message
       I '$$NEWMSG^HLOAPI(.APPARMS,.HLMSTATE,.ERROR) W !,ERROR Q
       ;
       ; Create segments
       D SET^HLOAPI(.SEG,"SFT",0)
       D SET^HLOAPI(.SEG,"VISTA",1)
       D SET^HLOAPI(.SEG,"2.0T3",2)
       D SET^HLOAPI(.SEG,"HL7",3)
       D SET^HLOAPI(.SEG,"43",4)
       D SET^HLOAPI(.SEG,$$HLDATE^HLFNC("3041229.092123","TS"),6)
       ; Add segments to the message
       I '$$ADDSEG^HLOAPI(.HLMSTATE,.SEG) Q
       ;
       ; Create segments
       D SET^HLOAPI(.SEG,"QRD",0)
       D SET^HLOAPI(.SEG,$$HLDATE^HLFNC($$NOW^XLFDT(),"TS"),1)
       D SET^HLOAPI(.SEG,"R",2)
       D SET^HLOAPI(.SEG,"I",3)
```

```
       D SET^HLOAPI(.SEG,1,4)
       D SET^HLOAPI(.SEG,"RD",7,2)
       D SET^HLOAPI(.SEG,1,7,1)
       D SET^HLOAPI(.SEG,"NCK",9)
       D SET^HLOAPI(.SEG,"S",12)
       ; Add segments to the message
       I '$$ADDSEG^HLOAPI(.HLMSTATE,.SEG) Q
       ;
       ; Create segments
       D SET^HLOAPI(.SEG,"NCK",0)
       D SET^HLOAPI(.SEG,$$HLDATE^HLFNC($$NOW^XLFDT(),"TS"),1)
       ; Add segments to the message
       I '$$ADDSEG^HLOAPI(.HLMSTATE,.SEG) Q
       ;
       ; Define the sending and receiving parameters for all recipients
       S APPARMS("SENDING APPLICATION")="HLO NMQ"
       S APPARMS("ACCEPT ACK TYPE")="NE"
       S WHOTO(1,"FACILITY LINK NAME")="VAAAA"
       S WHOTO(1,"RECEIVING APPLICATION")="HLO NMQ"
       S WHOTO(2,"FACILITY LINK NAME")="VABBB"
       S WHOTO(2,"RECEIVING APPLICATION")="HLO NMQ"
       S WHOTO(3,"FACILITY LINK NAME")="VACCC"
       S WHOTO(3,"RECEIVING APPLICATION")="HLO NMQ"
       S WHOTO(4,"FACILITY LINK NAME")="LINK OUT"
       S WHOTO(4,"RECEIVING APPLICATION")="HLO NMQ"
       ;
       ; Send the message to all recipients
       I '$$SENDMANY^HLOAPI1(.HLMSTATE,.APPARMS,.WHOTO) D
       . S I=0 F  S I=$O(WHOTO(I)) Q:'I  I $D(WHOTO(I,"ERROR")) W !,WHOTO(I,"FACILITY
LINK NAME"),"  ",WHOTO(I,"ERROR")
       Q
```

WARNING: The facility link name defined in parameter WHOTO(#,"FACILITY LINK NAME") must be a valid entry from the HL LOGICAL LINK File (#870). A valid entry for HLO must have the fields DNS DOMAIN (#.08) and TCP/IP PORT (OPTIMIZED) (#400.08) populated. Failure to use a valid entry from this file will result in a "DOMAIN NOT FOUND" error.

Problems with a link will be returned in the "ERROR" subscript within the WHOTO array.

```
WHOTO(3,"ERROR")="DOMAIN NOT FOUND"
WHOTO(3,"FACILITY LINK NAME")="VACCC"
WHOTO(3,"IEN")=""
WHOTO(3,"QUEUED")=0
WHOTO(3,"RECEIVING APPLICATION")="HLO NMQ"
```

6.1.2.4 Send a Batch Message to Multiple Recipients

Sending batch messages to multiple recipients is a combination of the techniques presented in two previous sections: "Send a Batch Message" and "Send a Message to Multiple Recipients." To build and send a batch message to multiple recipients, the developer needs to follow these steps:

1. Define message parameters (APPARMS array).
2. Create new batch message ($$NEWBATCH^HLOAPI).
    a. Create messages within the batch ($$ADDMSG^HLOAPI).
        i. Create segments within each message ($$ADDSEG^HLOAPI).
3. Define sending and receiving parameters for all recipients. (APPARMS and WHOTO arrays).
4. Send the batch message to all recipients ($$SENDMANY^HLOAPI1).

Below is a sample routine written to generate and send a batch HL7 message to multiple recipients:

```
HLOBTMNY     ; Sending Batches to Many
       ;
       ;  Define message parameters
       ;{Declare additional parameters}
       ;
       ; Create new batch message
       I '$$NEWBATCH^HLOAPI(.APPARMS,.HLMSTATE,.ERROR) W !,ERROR Q
       ;
       ; Create messages within the batch
       F FILE=11,12,13 D BTMSG
       I $G(ERR)'="" W !,ERR Q
       ;
       ; Define sending and receiving parameters for all recipients
       S APPARMS("SENDING APPLICATION")="HLOZ NMQ"
       S APPARMS("ACCEPT ACK TYPE")="NE"
       S WHOTO(1,"FACILITY LINK NAME")="VAALB"
       S WHOTO(1,"RECEIVING APPLICATION")="HLOZ NMQ"
       S WHOTO(2,"FACILITY LINK NAME")="VACLE"
       S WHOTO(2,"RECEIVING APPLICATION")="HLOZ NMQ"
       S WHOTO(3,"FACILITY LINK NAME")="VAMAR"
       S WHOTO(3,"RECEIVING APPLICATION")="HLOZ NMQ"
       S WHOTO(4,"FACILITY LINK NAME")="TIGER OUT"
       S WHOTO(4,"RECEIVING APPLICATION")="HLOZ NMQ"
       ;
       ; Send the batch message to all recipients
       I '$$SENDMANY^HLOAPI1(.HLMSTATE,.APPARMS,.WHOTO) D
       . S I=0 F  S I=$O(WHOTO(I)) Q:'I  I $D(WHOTO(I,"ERROR")) W !,WHOTO(I,"FACILITY
LINK NAME"),"  ",WHOTO(I,"ERROR")
       Q
       ;
BTMSG  ; Create messages within the batch
       S PARMS("EVENT")="M13"
       S PARMS("MESSAGE TYPE")="MFN"
       I '$$ADDMSG^HLOAPI(.HLMSTATE,.PARMS,.ERROR) S ERR=ERROR Q
       ;
       ; Create segments within each message
       D SET^HLOAPI(.SEG,"MFI",0)
```

```
       D SET^HLOAPI(.SEG,FILE,1)
       D SET^HLOAPI(.SEG,"UPD",3)
       D SET^HLOAPI(.SEG,"NE",6)
       I '$$ADDSEG^HLOAPI(.HLMSTATE,.SEG) S ERR=”MFI NOT ADDED” Q
       ;
       ; For Each File Record, create segments within each message
       S GLREF=$$ROOT^DILFD(FILE,"",1)
       S IEN=0
       F  S IEN=$O(@GLREF@(IEN)) Q:'IEN  D
       .D SET^HLOAPI(.SEG,"MFE",0)
       .D SET^HLOAPI(.SEG,"MUP",1)
       .D SET^HLOAPI(.SEG,$$HLDATE^HLFNC($$DT^XLFDT(),"DT"),3)
       .D SET^HLOAPI(.SEG,IEN,4,1)
       .D SET^HLOAPI(.SEG,$P(@GLREF@(IEN,0),"^",1),4,2)
       .D SET^HLOAPI(.SEG,"CE",5)
       .I '$$ADDSEG^HLOAPI(.HLMSTATE,.SEG) S ERR="MFE NOT ADDED." Q
       Q
```

6.1.2.5 Convert Existing HL 1.6 Outgoing Messages to HLO Messages

To make HLO more accessible and usable by existing HL 1.6 applications, two routines have been developed for converting current HL 1.6 messages to HLO messages. They are:

- HLOCNRT – Takes a current HL 1.6 message that follows the standard HL 1.6 methodology, converts it to use the HLO methodology, and places it onto an HLO message queue. The function call EN^HLOCNRT is equivalent to the HL 1.6 function call GENERATE^HLMA.

- HLOCVU – Called by HLOCNRT. Retrieves HL 1.6 parameters from the existing HL 1.6 protocol and translates them to HLO parameters. The translated parameters include:

HL 1.6 APPLICATION PROTOCOL	**=>**	**HLO APPARMS ARRAY PARAMETERS**
COUNTRY CODE	=>	APPARMS(“COUNTRY”)
APPLICATION ACK TYPE	=>	APPARMS(“APP ACK TYPE”)
EVENT TYPE	=>	APPARMS(“EVENT”)
SENDING APPLICATION	=>	APPARMS(“SENDING APPLICATION”)
TRANSACTION MESSAGE TYPE	=>	APPARMS(“MESSAGE TYPE”)
VERSION ID	=>	APPARMS(“VERSION”)
HL7 FIELD SEPARATOR	=>	APPARMS(“FIELD SEPARATOR”)
HL7 ENCODING CHARACTERS	=>	APPARMS(“ENCODING CHARACTERS”)

HL 1.6 Passed Parameters	**=>**	**HLO APPARMS ARRAY PARAMETERS**
HLP("SECURITY")	=>	APPARMS("SECURITY")
HLP("CONTPTR")	=>	APPARMS("CONTINUATION POINTER")
HLP(“QUEUE”)*	=>	APPARMS(“QUEUE”)

NOTE: HLP(“QUEUE”) is not actually a current HL 1.6 parameter but can be added to the HLP array to allow a converted application to define HLO private queues.

For Sending Messages To One Application

RECEIVING APPLICATION	=>	WHO(“RECEIVING APPLICATION”)
LOGICAL LINK	=>	WHO(“FACILITY LINK NAME”)

For Sending Messages To Multiple Applications (where “n” is a numeric index (0,1,2,…))

RECEIVING APPLICATION	=>	WHOTO(n,”RECEIVING APPLICATION”)
LOGICAL LINK	=>	WHOTO(n,”FACILITY LINK NAME”)

Conversion Components

The first component to change is the standard call GENERATE^HLMA (for generating HL 1.6 messages) to the new function call EN^HLOCNRT. Both calls use the variable HLEID, which defines the IEN of the event protocol (file #101) and both return the variable HLRESLT=<message id^error code^error description>.

- If the value of HLFORMAT is not 1 in the current GENERATE^HLMA call, then all coding must be done separately. The $$EN^HLOCNRT API assumes that the message has been pre-formatted in HL7 format.

- If the value of HLMTIEN is anything other than null in the current GENERATE^HLMA call, then all coding must be done separately. The $$EN^HLOCNRT API does not support the partial creation of an HL7 message.

- The value of HLP("NAMESPACE") is not used in HLO. This information is now retrieved from the Package File Link field (#2) in the HLO APPLICATION REGISTRY File (#779.2).

```
      ;
      ; D GENERATE^HLMA(HLEID,"GM",1,.HLRESLT,"",.HLP)
      S HLP("QUEUE")="<queue name>" ; only if a private queue is needed
      S HLRESLT=$$EN^HLOCNRT(HLEID,"GM",.HLP)
      ;
```

The following box contains the conversion routine, HLOCNRT. Within this routine, the new HLO message is created using parameters initialized in HL 1.6. The routine APAR^HLOCVU is called to perform the actual parameter conversion. Once the call to HLOCVU is completed, HLO APIs are used to create the new HLO message, move the HL 1.6 message to an HLO queue, and send the message via the HLO engine.

```
HLOCNRT  ;DAOU/ALA-Generate HL7 Optimized Message ; 17 Jun 2005  12:57 PM
         ;;1.6;HEALTH LEVEL SEVEN;**126**;Oct 13, 1995
         ;
         ;**Program Description**
         ;  This program takes a current HL7 1.6 message and converts
         ;  it to use the new HL Optimized code if it follows the standard
         ;  1.6 methodology of protocols.
         ;
         ;  **If the VistA HL7 Protocol does not exist, calls to HL Optimized
         ;  will have to be coded separately and this program cannot be used**
         Q
         ;
EN(HLOPRTCL,ARYTYP,HLP) ;Entry Point
         ;  Input Parameters
         ;   HLOPRTCL = Protocol IEN or Protocol Name
         ;   ARYTYP = The array where HL7 message resides
         ;   HLP = Additional HL7 message parameters
         ;
         ;  Output
         ;    ZTSTOP = Stop processing flag (used by HDR)
         ;    HLORESL = Error parameter
         ;
         NEW HLORESL,HLMSTATE,APPARMS,WHOTO,WHO,ERROR,HLOMESG
         S ZTSTOP=0,HLORESL=1
         ;
         ;  Get IEN of protocol if name is passed
```

```
         I HLOPRTCL'?.N S HLOPRTCL=+$O(^ORD(101,"B",HLOPRTCL,0))
         I '$D(^ORD(101,HLOPRTCL)) S HLORESL="^99^HL7 1.6 Protocol not found",ZT
STOP=1 Q HLORESL
         ;
         ;  If the VistA HL7 Protocol exists, call the Conversion Utility
         ;  to set up the APPARMS, WHO or WHOTO arrays from protocol
         ;  logical link
         D APAR^HLOCVU(HLOPRTCL,.APPARMS,.WHO,.WHOTO)
         ;
         ; If special HLP parameters are defined, convert them
         I $D(HLP) D
         . I $G(HLP("SECURITY"))'="" S APPARMS("SECURITY")=HLP("SECURITY")
         . I $G(HLP("CONTPTR"))'="" S APPARMS("CONTINUATION POINTER")=HLP("CONTP
TR")
         . I $G(HLP("QUEUE"))'="" S APPARMS("QUEUE")=HLP("QUEUE")
         ;
         ;  Create HL Optimized message
         I '$$NEWMSG^HLOAPI(.APPARMS,.HLMSTATE,.ERROR) S HLORESL="^99^"_ERROR,ZT
STOP=1 Q HLORESL
         I $E(ARYTYP,1)="G" S HLOMESG="^TMP(""HLS"",$J)"
         I $E(ARYTYP,1)="L" S HLOMESG="HLA(""HLS"")"
         ;
         ;  Move the existing message from array into HL Optimized
         D MOVEMSG^HLOAPI(.HLMSTATE,HLOMESG)
         ;
         ;  Send message via HL Optimized
         I $D(WHOTO) D  Q HLORESL
         . I '$$SENDMANY^HLOAPI1(.HLMSTATE,.APPARMS,.WHOTO) S HLORESL="^99^Unabl
e to send message",ZTSTOP=1 Q
         . S HLORESL=1
         ;
         I '$$SENDONE^HLOAPI1(.HLMSTATE,.APPARMS,.WHO,.ERROR) S HLORESL="^99^"_E
RROR,ZTSTOP=1 Q HLORESL
         Q HLORESL
```

Below is the HLOCVU routine. In looking at this routine, one can see how the standard HL 1.6 variables are translated into the HLO APPARMS array and returned to the calling routine (such as HLOCNRT).

WARNING: HLOCVU expects the values of HLFS and HLECH as set from INIT^HLFNC2. Changing these variables before the call to HLOCNRT or HLOCVU may negatively impact the conversion of the HL7 message.

```
HLOCVU   ;DAOU/ALA-Conversion Utility ; 04 Aug 2005  4:26 PM
         ;;1.6;HEALTH LEVEL SEVEN;**126**;Oct 13, 1995
         ;
         Q
         ;
APAR(HLOEID,APPARMS,WHO,WHOTO) ;  Set up APPARMS array from Protocols
         ;
         ;  Input Parameter
         ;   HLOEID = IEN of the event protocol
         ;
         ;  Output
         ;    APPARMS array
         ;    WHO or WHOTO array
         NEW HLOTEXT,HLARY,FLDS,HLOSID,CT
         S FLDS="770.1;770.3;770.4;770.8;770.9;770.95"
```

```
         D GETS^DIQ(101,HLOEID,FLDS,"R","HLOTEXT")
         ;
         S APPARMS("COUNTRY")="USA"
         S APPARMS("EVENT")=$G(HLOTEXT(101,HLOEID_",","EVENT TYPE"))
         S APPARMS("MESSAGE TYPE")=$G(HLOTEXT(101,HLOEID_",","TRANSACTION MESSAG
E TYPE"))
         S APPARMS("VERSION")=$G(HLOTEXT(101,HLOEID_",","VERSION ID"))
         S APPARMS("SENDING APPLICATION")=$G(HLOTEXT(101,HLOEID_",","SENDING APP
LICATION"))
         S APPARMS("APP ACK TYPE")=$G(HLOTEXT(101,HLOEID_",","APPLICATION ACK TY
PE"))
         S APPARMS("ACCEPT ACK TYPE")=$G(HLOTEXT(101,HLOEID_",","ACCEPT ACK CODE
"))
         I $G(HLFS)'="" S APPARMS("FIELD SEPARATOR")=HLFS
         I $G(HLECH)'="" S APPARMS("ENCODING CHARACTERS")=HLECH
         ;
         K HLOTEXT,FLDS
         D ITEM^HLUTIL2(HLOEID,"PTR")
         I $G(HLARY(0))>1 D MANY Q
         S HLOSID=$O(HLARY(0))
         S FLDS="770.2;770.4;770.7"
         D GETS^DIQ(101,HLOSID,FLDS,"R","HLOTEXT")
         S WHO("RECEIVING APPLICATION")=$G(HLOTEXT(101,HLOSID_",","RECEIVING APP
LICATION"))
         S WHO("FACILITY LINK NAME")=$G(HLOTEXT(101,HLOSID_",","LOGICAL LINK"))
         Q
         ;
MANY     ;  If multiple subscribers
         S HLOSID=0,CT=0
         S FLDS="770.2;770.4;770.7"
         F  S HLOSID=$O(HLARY(HLOSID)) Q:'HLOSID  D
         . K HLOTEXT
         . D GETS^DIQ(101,HLOSID,FLDS,"R","HLOTEXT")
         . S CT=CT+1
         . S WHOTO(CT,"RECEIVING APPLICATION")=$G(HLOTEXT(101,HLOSID_",","RECEIV
ING APPLICATION"))
         . S WHOTO(CT,"FACILITY LINK NAME")=$G(HLOTEXT(101,HLOSID_",","LOGICAL L
INK"))
         ;
         Q
```

6.1.3 Incoming Messages

HLO APIs have been written to:

- Parse incoming messages
- Generate Application Acknowledgements
- Convert Existing HL 1.6 Incoming Messages to HLO Messages

The following sections contain sample code to demonstrate this functionality. For detailed information on the API calls used in the sample code, refer to Appendix B.

6.1.3.1 Parse incoming messages

To parse an incoming message, follow the steps below:

1. Parse the header and return individual values ($$STARTMSG^HLOPRS).
2. Advance to each segment in the message. ($$NEXTMSG^HLOPRS).
    a. Parse each segment and return individual values for processing ($$GET^HLOPRS).
3. Determine if an application acknowledgement has been requested (variable HDR("APP ACK TYPE")).

Below is a sample routine written to demonstrate how to parse an incoming message:

```
HLOPARSE     ; Parsing Messages
       ; Parse the header and return individual values
MSG    I '$$STARTMSG^HLOPRS(.HLMSTATE,HLMSGIEN,.HDR) Q
       ;
       ; Advance to each segment in the message
SEQ    I '$$NEXTMSG^HLOPRS(.HLMSTATE,HLMSGIEN,.HDR) G EXIT
       ;
       ; Parse each segment and return individual values for processing
       S SEGTYP=$$GET^HLOPRS(.SEG,0)
       S FLAG=$$GET^HLOPRS(.SEG,1)
       S VIEN=$$GET^HLOPRS(.SEG,4,1)
       S VALUE=$$GET^HLOPRS(.SEG,4,2)
       S ^TMP($J,VIEN)=VALUE
       ;
       ; Continue to parse and process remaining segments and messages
       ……..
       ……..
       G SEQ
EXIT   ; Determine if an application acknowledgement has been requested
       I HDR("APP ACK TYPE")="AL" D ACK
       Q
ACK    ;Generate application acknowledgement
       ……..
       ……..
       Q
```

6.1.3.2 Generate Application Acknowledgements

HLO APIs include functionality to create and send application acknowledgements to the original sending application.

When the sending application sends a message, the receiving system processes the message and determines if an application acknowledgement needs to be returned. In the example below, a simple acknowledgement has been added to the sample routine used for parsing messages.

An important parameter involving application acknowledgements is the APP ACK TYPE. This parameter is passed within the header of the message and has a value of "NE" ("never") or "AL" ("always"). In the example below, an acknowledgement is returned to the sending application once the message is processed.

The receiving application can determine whether the sending application wants an application acknowledgement by checking the value of APP ACK TYPE. The APP ACK TYPE is extracted from the message header by the $$START^HLOPRS API. If, as in this example, the value of HDR("APP ACK TYPE") is "NE", the process should exit before the acknowledgement is created and sent. It is up to the developer of the receiving application to make sure that this type of situation is handled properly and that the APP ACK TYPE is verified before sending the application acknowledgement.

To generate an application acknowledgement, follow the steps below:

1. Start the Application Acknowledgement message ($$ACK^HLOAPI2).
2. Add segments to the message ($$ADDSEG^HLOAPI2).
3. Send the Application Acknowledgement ($$SENDACK^HLOAPI2).

Below is a sample routine written to demonstrate how to generate an application acknowledgement:

```
HLOPARSE     ; Parsing Messages
       ; Parse the header and return individual values
MSG    I '$$STARTMSG^HLOPRS(.HLMSTATE,HLMSGIEN,.HDR) Q
       ;
       ; Advance to each segment in the message.
SEQ    I '$$NEXTMSG^HLOPRS(.HLMSTATE,HLMSGIEN,.HDR) G EXIT
       ;
       ; Parse each segment and return individual values for processing
       S SEGTYP=$$GET^HLOPRS(.SEG,0)
       S FLAG=$$GET^HLOPRS(.SEG,1)
       S VIEN=$$GET^HLOPRS(.SEG,4,1)
       S VALUE=$$GET^HLOPRS(.SEG,4,2)
       S ^TMP($J,VIEN)=VALUE
       ;
       ;Continue to parse and process remaining segments and messages.
       ……..
       ……..
       G SEQ
       ;
EXIT   ; Determine if an application acknowledgement has been requested
       I HDR("APP ACK TYPE")="AL" D ACK
       Q
ACK    ; Generate Application Acknowledgement.
       S PARMS("ACK CODE")="AA"
       S PARMS("MESSAGE TYPE")="MFK"
       ; Start the application acknowledgement message
       I '$$ACK^HLOAPI2(.HLMSTATE,.PARMS,.ACK,.ERROR) W !,ERROR Q
       ; Add segments to the message
       I '$$ADDSEG^HLOAPI2(.HLMSTATE,.PARMS,.ACK,.ERROR) W !,ERROR Q
       I $G(ERR)]"" S PARMS("ACK CODE")="AE"
       ;
       ; Send the Application Acknowledgement
       I '$$SENDACK^HLOAPI2(.ACK,.ERROR) W !,ERROR Q
       Q
```

6.1.3.3 Convert Existing HL 1.6 Incoming Messages to HLO Messages

In the current HL 1.6 software, the processing routine (PROCESSING RTN) is set up in the Subscriber protocol.

```
                    HL7 SUBSCRIBER                          PAGE 2 OF 2
                    VEPEXYZ HDM RESPONSE
------------------------------------------------------------------------

RECEIVING APPLICATION: ABC VISTA

RESPONSE MESSAGE TYPE: ACK                            EVENT TYPE: A08

SENDING FACILITY REQUIRED?:              RECEIVING FACILITY REQUIRED?:

SECURITY REQUIRED?:

LOGICAL LINK: ABC SERVER

PROCESSING RTN: D ^VEXYZ3IN

ROUTING LOGIC:
```

In HLO, the processing routine is set up in the HLO APPLICATION REGISTRY (#779.2), where action calls can vary, depending on the message and event type.

```
Select HLO APPLICATION REGISTRY APPLICATION NAME: ABC HDM
APPLICATION NAME: ABC HDM// <RET>
Package File Link: VEABC// <RET>
RESPONSE LINK (OPTIONAL): <RET>
DEFAULT PRIVATE IN-QUEUE: HDM// <RET>
DEFAULT ACTION TAG: <RET>
DEFAULT ACTION ROUTINE: <RET>
BATCH PRIVATE IN-QUEUE: <RET>
BATCH ACTION TAG: <RET>
BATCH ACTION ROUTINE: <RET>
Select HL7 MESSAGE TYPE: ADT
     1   ADT    A08
     2   ADT    A28
CHOOSE 1-2: 1  ADT  A08
  HL7 MESSAGE TYPE: ADT// <RET>
  HL7 EVENT: A08// <RET>
  PRIVATE IN-QUEUE: A08/ <RET>/
  ACTION TAG: <RET>
  ACTION ROUTINE: ??
     You must enter the action to perform upon receipt of this type by entering
     the ACTION TAG and ACTION ROUTINE fields as <tag>^<routine>.
  ACTION TAG: EN
  ACTION ROUTINE: VEABC3IN
Select HL7 MESSAGE TYPE:
```

In HL 1.6, the standard call to retrieve message segments is ‘X HLNEXT’ where HLNEXT="D HLNEXT^HLCSUTL".

```
HLNEXT ;-- This routine is used to return the next segment from file 772
       ;   during processing of an inbound message. The following variables
       ;   are used for the processing.
       ;   HLMTIEN - Entry in 772 where message is
       ;   HLQUIT  - Current ien of "IN" wp field
       ;   HLNODE  - Data is returned in HLNODE=Segment and HLNODE(n) if
       ;             segment is greater than 245 chars.
```

In the text box below, the method for processing HL 1.6 messages, using ‘X HLNEXT’ is displayed.

Original HL 1.6 processing code:

```
EN     ;  Starting point - put message into a TMP global
       ;
       ;  Load the HL7 message into temporary global
       K ^TMP($J,"VEPHLI")
       F SEGCNT=1:1 X HLNEXT Q:HLQUIT'>0  D
       . S CNT=0
       . S ^TMP($J,"VEPHLI",SEGCNT,CNT)=HLNODE
       . F  S CNT=$O(HLNODE(CNT)) Q:'CNT  D
       .. S ^TMP($J,"VEPHLI",SEGCNT,CNT)=HLNODE(CNT)
       ;
       S SEGMT=$G(^TMP($J,"VEPHLI",1,0))
       I $E(SEGMT,1,3)'="MSH" S MSG(1)="MSH Segment is not the first
       segment found" D ERR Q
```

To create or rewrite incoming message processing for HLO, the $$NEXTSEG^HLOPRS function should be used and is preferred. To directly replace the ‘X HLNEXT’ process without changing the current processing, the $$HLNEXT^HLOMSG function can be used. The one difference between the ‘X HLNEXT’ and the $$HLNEXT API is that $$HLNEXT always returns the data in an array. ‘X HLNEXT’ sets the initial value into the variable not at a subscript level and only returns an array if the segment is greater than 245 characters.

In the text box below, 'X HLNEXT' has been replaced with two HLO APIs, including $$STARTMSG^HLOPRS, which initiates the parsing of the message, and $$HLNEXT^HLOMSG, which parses each segment.

HLO Processing Code (using HL 1.6 Equivalent functions):

```
EN      ;  Starting point
        I '$$STARTMSG^HLOPRS(.HLMSTATE,HLMSGIEN,.HDR) Q
        ;
        ;  Load the HL7 message into temporary global
        K ^TMP($J,"VEPHLI")
        ;
        S SEGCNT=1
        S ^TMP($J,"VEPHLI",SEGCNT,0)=HLMSTATE("HDR",1)_HLMSTATE("HDR",2)
        ;
        F  Q:'$$HLNEXT^HLOMSG(.HLMSTATE,.SEG)  D
        . S SEGCNT=SEGCNT+1
        . S CNT=0,SCNT=0
        . F  S CNT=$O(SEG(CNT)) Q:'CNT  D
        .. S ^TMP($J,"VEPHLI",SEGCNT,SCNT)=SEG(CNT),SCNT=SCNT+1
        ;
        S SEGMT=$G(^TMP($J,"VEPHLI",1,0))
        I $E(SEGMT,1,3)'="MSH" S MSG(1)="MSH Segment is not the first
        segment found" D ERR Q
        ;
```

In the text box below, the $$NEXTSEG^HLOPRS API is utilized by HLO to process each segment, one at a time. Within this API, $$HLNEXT^HLOMSG is utilized to parse each data value into a local array. $$GET^HLOPRS is then utilized for assigning parsed values to application variables.

New HLO Processing Code:

```
EN     ;  Starting point
       I '$$STARTMSG^HLOPRS(.HLMSTATE,HLMSGIEN,.HDR) Q
       ;
       F  Q:'$$NEXTSEG^HLOPRS(.HLMSTATE,.SEG)  D
       . I $$GET^HLOPRS(.SEG,0)="PID" D GTPID
       . ……
       . ……
       . Q
       Q
       ;
GTPID  S NAME("FAMILY")=$$GET^HLOPRS(.SEG,5,1)
       S NAME("GIVEN")=$$GET^HLOPRS(.SEG,5,2)
       S NAME("MIDDLE")=$$GET^HLOPRS(.SEG,5,3)
       S PATNAME=$$BLDNAME^XLFNAME(.NAME,30)
```

EXAMPLE:

As HLO loops through each segment (using $$NEXTSEG^HLOPRS), the following PID segment is identified:

```
PID||9211|999028467|849-2959913|HLpatient^ONE^J||19580525|M||||||||||999028467|||||||19
```

$$NEXTSEG^HLOPRS returns parsed values from the segment by placing the data into a local array that looks something like this:

```
SEG(1,1,1,1)="PID"
SEG(3,1,1,1)=9211
SEG(4,1,1,1)=999028467
SEG(5,1,1,1)="849-2959913"
SEG(6,1,1,1)="HLpatient"
SEG(6,1,2,1)="ONE"
SEG(6,1,3,1)="J"
SEG(8,1,1,1)=19580525
SEG(9,1,1,1)="M"
SEG(20,1,1,1)=999028467
SEG(27,1,1,1)=19
SEG("SEGMENT TYPE")="PID"
```

Within the specific message/segment process by the application, $$GET^HLOPRS takes values from the local array and assigns them to application specific variables. In this particular example, the patient's name is parsed and rebuilt into a newly formatted variable called 'PATNAME' (which is "HLpatient,ONE J").

```
S NAME("FAMILY")=$$GET^HLOPRS(.SEG,5,1)
S NAME("GIVEN")=$$GET^HLOPRS(.SEG,5,2)
S NAME("MIDDLE")=$$GET^HLOPRS(.SEG,5,3)
S PATNAME=$$BLDNAME^XLFNAME(.NAME,30)
```

HL7 Attribute Table - PID - Patient Identification

SEQ	LEN	DT	OPT	RP/#	TBL#	ITEM#	ELEMENT NAME
1	4	S	O			00104	Set ID - PID
2	20	CX	B			00105	Patient ID
3	250	CX	R	Y		00106	Patient Identifier List
4	20	CX	B	Y		00107	Alternate Patient ID - PID
5	**250**	**XPN**	**R**	**Y**		**00108**	**Patient Name**
6	250	XPN	O	Y		00109	Mother's Maiden Name

It is very important to be aware that the field parameter used by $$GET^HLOPRS is always the HL7 SEQ (sequence number) which is different from the subscript value in the SEG array. This is explained by the difference between $Piece and the HL7 sequence (SEQ).

In Mumps programming, the first data field in a delimited string is defined as the "first piece," the second field as the "second piece," and so on. However, in HL7, the first data field (which is the segment type) is not assigned a specific sequence value and so for HLO purposes is considered to be "sequence zero." The first actual data field value is assigned sequence 1 (one), and so on. In order to properly process and manage the data according to the standards of the two processes being used, it is important that these two distinct numbering methods remain intact.

To illustrate the difference between piece and sequence, here is an example where piece 6 equates to sequence 5. As long as the developer uses the HL7 sequence number, the software handles the translation automatically.

```
SEG(6,1,1,1)="HLpatient
SEG(6,1,2,1)="ONE"
SEG(6,1,3,1)="J"

S NAME("FAMILY")=$$GET^HLOPRS(.SEG,5,1)
S NAME("GIVEN")=$$GET^HLOPRS(.SEG,5,2)
S NAME("MIDDLE")=$$GET^HLOPRS(.SEG,5,3)
```

6.1.4 Accept Acknowledgements

HLO includes functionality for calling a routine in response to receiving or not receiving an Accept Acknowledgement. The parameter for this function is APPARMS("ACCEPT ACK RESPONSE").

A simple example of this is shown below, where an error message is generated if the Accept Ack is not received. In order for this routine to be called, the sending application includes the line:

```
S APPARMS("ACCEPT ACK RESPONSE")="MSG^HLOZTST"
```

```
MSG    ; Accept ACK Routine test
       S CACK=$G(^HLB(HLSMGIEN,4))
       I $P(CACK,"^",3)["|CA|" Q
       I $P(CACK,"^",3)'["|CA|" D
       . NEW DIFROM
       . S XMSUB="HL7 ERROR",XMDUZ="HL Optimized Interface"
       . S MSG(1)="HL7 Message "_^HLB(HLMSGIEN,1)_^HLB(HLSMGIEN,2)
       . S MSG(2)=" "
       . S MSG(3)="did not receive a valid commit acknowledgement."
       . S MSG(4)=CACK
       . S XMY("G.HL MESSAGING")=""
       . S XMTEXT="MSG("
       . D ^XMD
       . K XMSUB,XMY,XMTEXT,MSG,XMZ,XMDUZ,CACK
       Q
```

Program MSG^HLOZTST returns the following MailMan message:

```
Subj: HL7 ERROR  [#6277] 08/03/05@10:25  4 lines
From: HL OPTIMIZED INTERFACE  In 'IN' basket.   Page 1  *New*
---------------------------------------------------------------------------
HL7 Message MSH|^~\&|HLO NMQ|151^FELIX.DAOU.COM:5001^DNS|HLO NMQ|^DSI-ALPHA.DAO
U.COM:5001^DNS|20050803102510-0500||NMQ^N01^|151 13|T^|2.4|||AL|NE|USA

did not receive a valid commit acknowledgement.
3050803.102511^152 100000000012^MSA|CR|151 13|RECEIVING APPLICATION NOT DEFINED
|

Enter message action (in IN basket): Ignore//
```

6.1.5 Error messages from API calls

See Appendix H for a list of some of the potential error messages that might be returned by various APIs in variable ERROR or the WHOTO array "ERROR" subscript.

6.1.6 Queue Management

Manage queues automatically, outside of the HLO System Monitor, including:

- Stopping a queue (STOPQUE^HLOQUE)
- Re-Start a queue (STARTQUE^HLOQUE)
- Check the status of a queue ($$STOPPED^HLOQUE)

NOTE: The three queue APIs are intended for use in a KIDS install when a VistA module utilizes a specific HLO queue. Pre and post-build processes that use these APIs should be developed to stop and restart the queue. This will prevent errors from occurring during the installation process.

When stopping a queue, the queue processor only checks for the flag after processing the current message. A hang may need to be included after setting a "Stop" flag on a named queue depending on the size of the messages being processed.

The first example below illustrates the practical use of the queue management APIs as part of the pre-installation process. In this case two incoming and outgoing queues are checked to see if they are running. If they are running, they are stopped.

```
HLOZPRE  ;Pre-Install Program ; 17 Jun 2005  1:10 PM
         ;
         ;  Define queues and whether you want to stop both
         ;  incoming and outgoing queues
         F QUEUE="VDEF ALLERGY","VDEF ADVERSE" D
         . F DIR="IN","OUT" D
         .. I '$$STOPPED^HLOQUE(DIR,QUEUE) D STOPQUE^HLOQUE(DIR,QUEUE)
         Q
```

The second example illustrates the practical use of the queue management APIs as part of the post-installation process. In this case two incoming and outgoing queues are checked to see if they are running. If they are not running, they are started.

```
HLOZPOST ;Post-Install Program ; 16 Jun 2005  4:14 PM
         F QUEUE="VDEF ALLERGY","VDEF ADVERSE" D
         . F DIR="IN","OUT" D
         .. I $$STOPPED^HLOQUE(DIR,QUEUE) D STARTQUE^HLOQUE(DIR,QUEUE)
         Q
```

6.2 Configure a Link for the Client Application

To define new or edit existing link definitions, use the *Link Edit* option on the Interface Developer Options menu:

```
Select Interface Developer Options Option: Link Edit

Select HL LOGICAL LINK NODE: PSU SEND
```

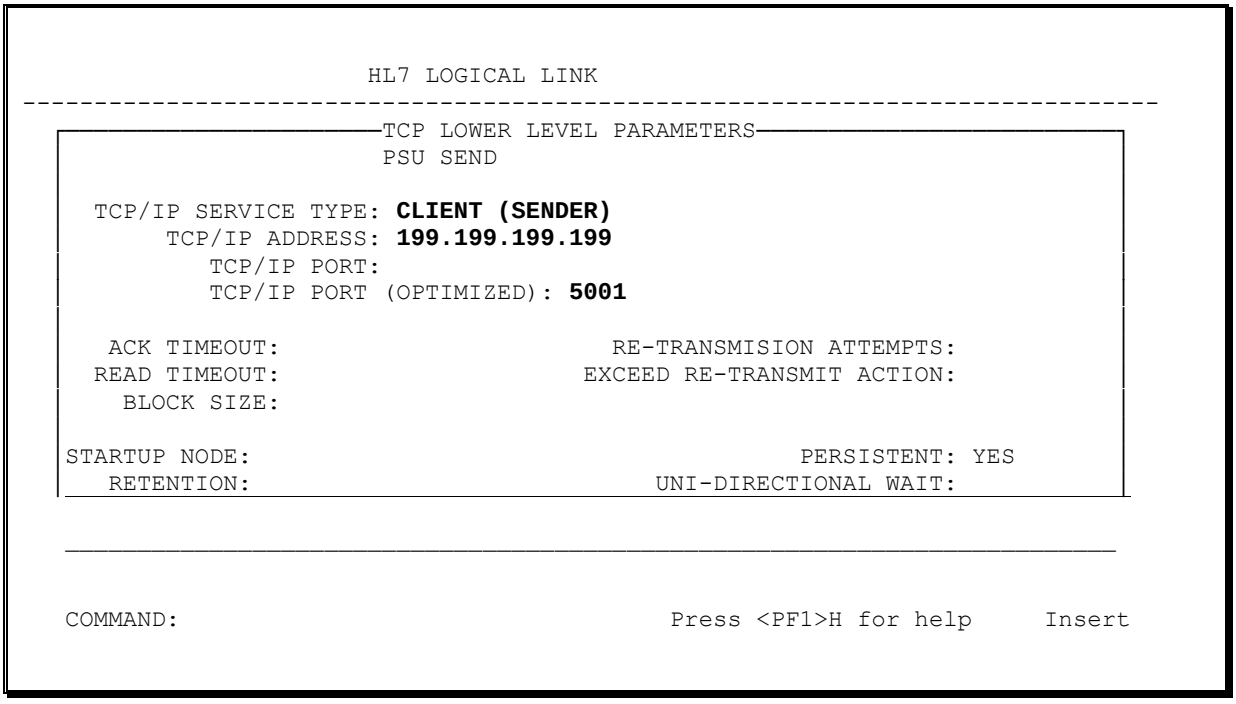
To edit the TCP/IP parameters, tab down to the LLP Type field and press <RET>. A form is displayed to edit the fields specific to the LLP type of the selected Link:

```
                          HL7 LOGICAL LINK
-----------------------------------------------------

                NODE: PSU SEND

         INSTITUTION:

      MAILMAN DOMAIN:

            AUTOSTART:

          QUEUE SIZE: 10

             LLP TYPE: TCP

          DNS DOMAIN: CMOP.NAT.MED.VA.GOV
```

To define client logical links, key set-up information includes:

- TCP/IP SERVICE TYPE – Set to "CLIENT"
- TCP/IP ADDRESS – Enter the IP Address of the server you are accessing.
- TCP/IP PORT (OPTIMIZED) – Enter the port number of the server you are accessing. If the receiving site is also running HLO, the port number should be 5001 for production systems and 5026 for test systems, otherwise check with the owner of the receiving application.
- DNS DOMAIN – Enter the Domain Name of the server you will be accessing. Note that DNS DOMAIN is required for HLO. The DNS DOMAIN field is found on the primary HL7 Logical Link screen after exiting from this screen.

WARNING: If an existing HL Logical Link for HL7 1.6 is reused for HLO, **DO NOT** modify or delete any of the other fields already entered! Even though HLO does not use them, HL7 1.6 does.



If the TCP/IP Address is not defined, it is resolved automatically by the VHA DNS Domain server. If the system's domain is NOT registered in the VHA DNS server, the TCP/IP address should be defined.

Please make sure that the 'TCP/IP PORT (OPTIMIZED)' field and not the 'TCP/IP PORT' field is used.

6.3 Client Link Processes

All HLO processes run under the framework of the HLO Process Registry, with the exception of the VMS TCP/IP Service for HLO. Each type of process has an entry in the HLO PROCESS REGISTRY File (#779.1), including one for the client link processes called OUTGOING CLIENT LINK. As long as the HLO Process Manager is running, processes will be started automatically to send outgoing messages pending on all client links. The number of client link processes will vary throughout the day, responding automatically to workload.

6.4 Application Registration

This section details information on the following topics:

- Creating an Application Registry
- Application Registry Entry Example

For any HL7 application both the sending and receiving applications need to be identified and defined. For HLO, these applications are identified and defined in the HLO APPLICATION REGISTRY File (#779.2).

In the previous program examples, the values of APPARMS("SENDING APPLICATION") and APPARMS("RECEIVING APPLICATION") are both defined with the name of an application found in the HLO APPLICATION REGISTRY File (#779.2). For newly created HLO applications, it is not necessary to have the sending and receiving applications as separate entries. As shown in previous program examples, "HLOZ NMQ" was used for both sending and receiving. However, since existing HL 1.6 applications were defined separately, the applications still must be defined separately in the HLO APPLICATION REGISTRY File (#779.2) when using the HLO conversion APIs.

When a sending system requires an acknowledgement from the receiving system, the original sending application becomes the receiving application for the acknowledgement.

6.4.1 Creating an Application Registry

The primary function of the HLO APPLICATION REGISTRY File (#779.2) is to define actions to be taken by the receiving system with HL7 messages. Fields within the application entry specify routine calls to be used to process messages associated with the application.

NOTE: When defining an Application Registry at the receiving end, the registry must have, at a minimum, a Default Action Routine and a Default Action Tag configured and in place.

Prompts for creating an Application Registry are broken down as follows:

1. Application Name

```
Select HLO APPLICATION REGISTRY APPLICATION NAME: HLOZ NMQ
APPLICATION NAME: HLOZ NMQ// <RET>
```

This is the application name, which is the main identifier for the application. It is used to reference this application and is included in the header of messages associated with this application. If this is a receiving application, make sure that the APPLICATION NAME matches the 'receiving application' (MSH-5) or 'batch receiving application' (BHS-5).

2. Package File Link

```
Package File Link: HEALTH LEVEL SEVEN// ?
     Enter the package responsible for these messages.
 Answer with PACKAGE NAME, or PREFIX, or ADDITIONAL PREFIXES, or SYNONYM
 Do you want the entire 493-Entry PACKAGE List? N  (No)
Package File Link: HEALTH LEVEL SEVEN// <RET>
```

This is a package name associated with this application. It uses a package from the PACKAGE File (#9.4).

3. Application Specific Listener

```
APPLICATION SPECIFIC LISTENER: ??
    Applications are highly discouraged from establishing their own listeners.
    The use of the multi-listeners provide concurrent processing of many
    connections over the same port, so a dedicated listener will not provide an
    application with a performance boost, while it will cause the site
    additional work to maintain. So before establishing a dedicated listener,
    the application developer should verify the need.
APPLICATION SPECIFIC LISTENER: <RET>
```

This is the name of the logical link to use for applications which need to listen on a port other than 5001.

4. Default Private In-Queue

```
DEFAULT PRIVATE IN-QUEUE: ?
     You may create an optional default private in-queue by entering a string
     up to 20 characters in length, for messages that the default or batch
     action apply.
DEFAULT PRIVATE IN-QUEUE: <RET>
```

This is the name of the queue on which incoming single messages are stored. If this field is left blank, the queue with the name of "DEFAULT" will be used. Note that this field is for single messages only. Batch messages use BATCH PRIVATE IN-QUEUE.

5. Default Action

```
DEFAULT ACTION TAG: ?
     You can enter the action to perform upon  receipt of a message where no
     other action applies by entering the DEFAULT ACTION TAG and DEFAULT ACTION
     ROUTINE fields as <tag>^<routine>.
DEFAULT ACTION TAG: <RET>
DEFAULT ACTION ROUTINE: ?
     You can enter the action to perform upon receipt of a message where no
     other action applies by entering the DEFAULT ACTION TAG and DEFAULT ACTION
     ROUTINE fields as <tag>^<routine>.
DEFAULT ACTION ROUTINE: <RET>
```

The default action consists of two parts – the default action tag and the default action routine. The default action is used when multiple message or event types should be processed in the same manner. This is invoked for all messages for which no specific message/event action is defined.

For example, if an application receives seven types of ADT messages and treats them all the same way, it would be a good idea to define the Default Action for processing all of them. If six of them get treated the same way, but the seventh is treated differently, it is a good idea to define the Default Action to handle the six that are similar and to define a message/event specific action for the seventh.

Alternatively, this can be used for custom error processing, where all legitimate Message/Event types have their own definition, and Default Action is used for error handling, in the event that an unexpected message or event type is received.

6. Batch Private In-Queue

```
BATCH PRIVATE IN-QUEUE: ?
     You may establish a private incoming queue for your batch messages
     entering a unique name (name-spaced) up to 20 characters.
BATCH PRIVATE IN-QUEUE: <RET>
```

This is the name of the queue in which to store incoming batch messages. If this field is left blank, the queue with the name of "DEFAULT" will be used. Note that this field is for batch messages only. Single messages use DEFAULT PRIVATE IN-QUEUE.

7. Batch Action

```
BATCH ACTION TAG: ?
     If the application utilizes batch messages, the action to perform upon
     receipt of the message should be entered in the BATCH ACTION TAG and BATCH
     ACTION ROUTINE fields as <tag>^<routine>.
BATCH ACTION TAG: <RET>

BATCH ACTION ROUTINE: ?
     If the application utilizes batch messages, the action to perform upon
     receipt of the message should be entered in the BATCH ACTION TAG and BATCH
     ACTION ROUTINE fields as <tag>^<routine>.
BATCH ACTION ROUTINE: <RET>
```

The batch action consists of two parts – the batch action tag and the batch action routine. The batch action is used when multiple message or event types should be processed in the same manner. This is invoked for all messages for which no specific message/event action is defined.

For batch messages, the application is responsible for parsing each message within the batch and processing it. Generally, all individual messages within a batch consist of the same type of transaction. However, this is not a requirement of the HL7 standard. If an application does allow different types of transactions within the batch, it is the application's responsibility to check the message and event type to determine the appropriate action.

8. HL7 Message Type

```
Select HL7 MESSAGE TYPE: ?
       You may enter a new MESSAGE TYPE ACTIONS, if you wish
        Enter the 3 character HL7 Message Type.
Select HL7 MESSAGE TYPE: NMQ
```

If there are messages whose processing is not covered by the Default Action, they would need to be defined individually. This section asks the first question to setup message-specific processing – the HL7 MESSAGE TYPE. It is a three-letter code, such as ADT or ORU.

These fields should only be defined for non-batch messages.

8a. HL7 Message Type-HL7 Event

```
   HL7 EVENT: ?
     Enter the 3 character HL7 event type.
   HL7 EVENT: N01
```

This is the second part of the identification for the message. Together, HL7 MESSAGE TYPE (above) and HL7 EVENT define which messages are processed by this particular entry.

8b. HL7 Message Type-Private In-Queue

```
PRIVATE IN-QUEUE: ?
   You may create a private in-queue for message of this type
   by entering a string up to 20 characters.
PRIVATE IN-QUEUE: <RET>
```

If this particular type of message should be placed onto a specific queue, enter the name of that queue here. If this field is left blank the application level DEFAULT PRIVATE IN-QUEUE will be used. The queue name "DEFAULT" is used when both fields are left blank.

NOTE: Important Information about HLO Queues

- If a unique queue name is not assigned to a message queue, the "DEFAULT" queue will be used.
- Developers can assign messages to uniquely named queues by utilizing the APPARMS("QUEUE") parameter when generating outgoing messages (refer to 'Sending Messages' in Appendix B). It is recommended to use namespace conventions to define queues.
- Uniquely naming a queue has several advantages, including:
    - A shorter queue, causing:
        - Improved concurrency.
        - Faster processing.
    - The ability to stop the queue without interfering with other applications.
- Queues are processed 1000 messages at a time. The Process Manager then checks if there are other queues to be processed. If there are, the Outgoing Client begins to process the next queue. The Process Manager may then start another Outgoing Client process if there are more queues to be processed.
- If the link on the receiving side is a single listener, then a maximum of one queue should be used for that link. Multiple queues will not increase efficiency of transmissions in this case and may negatively impact the processing of that link.

8c. HL7 Message Type-Processing Routine

```
ACTION TAG: ?
   You must enter the action to perform upon receipt of this
   type by entering the ACTION TAG and ACTION ROUTINE fields
   as <tag>^<routine>. The tag is optional.
ACTION TAG:REC

ACTION ROUTINE: ?
   You must enter the action to perform upon receipt of this
   type by entering the ACTION TAG and ACTION ROUTINE fields
   as <tag>^<routine>.
ACTION ROUTINE:HLOZTST
```

This is the processing routine to parse and file this particular type of message.

NOTE: The ACTION TAG is optional, but the ACTION ROUTINE is required at this level. If Default Action is defined, these fields do not have to be defined. If the HL7 Message Type processing routine is defined, then it is executed. If not defined, it checks for the Default Action and executes it.

6.4.2 Application Registry Entry Example

Below is an example of defining the application named "HLOZ NMQ." This application processes messages with message type NMQ and event type N01 using routine call REC^HLOZTST.

```
(From the HLO Main Menu)

Select HL7 (Optimized) MAIN MENU Option: ?

   SM    HLO SYSTEM MONITOR
   MV    HLO MESSAGE VIEWER
   APPS  HLO APPLICATION REGISTRY

Enter ?? for more options, ??? for brief descriptions, ?OPTION for help text.

Select HL7 (Optimized) MAIN MENU Option: APPS  HLO APPLICATION REGISTRY

Select HLO APPLICATION REGISTRY APPLICATION NAME: HLOZ NMQ
  Are you adding 'HLO NMQ' as a new HLO APPLICATION REGISTRY (the 16TH)? No// Y  (Yes)
APPLICATION NAME: HLOZ NMQ// <RET>
Package File Link: HEALTH LEVEL SEVEN       HL
RESPONSE LINK (OPTIONAL): <RET>
DEFAULT PRIVATE IN-QUEUE: <RET>
BATCH ACTION TAG: <RET>
BATCH ACTION ROUTINE: <RET>
DEFAULT ACTION TAG: <RET>
DEFAULT ACTION ROUTINE: <RET>
BATCH PRIVATE IN-QUEUE: <RET>
Select HL7 MESSAGE TYPE: NMQ
  Are you adding 'NMQ' as a new HL7 MESSAGE TYPE (the 1ST for this HLO APPLICATION
REGISTRY)? No// Y  (Yes)
   HL7 MESSAGE TYPE HL7 EVENT: N01
  HL7 EVENT: N01// <RET>
  PRIVATE IN-QUEUE: <RET>
  ACTION TAG: REC
  ACTION ROUTINE: HLOZTST
Select HL7 MESSAGE TYPE: <RET>


Select HLO APPLICATION REGISTRY APPLICATION NAME: <RET>
```

6.5 Attaching an Application to a KIDS Build

Below is an example of how to transport the entry “HLOZ NMQ” in the HLO APPLICATION REGISTRY File (#779.2) in a KIDS build.

File List Screen of KIDS:

```
                        Edit a Build                          PAGE 2 OF 4

Name: HL*1.6*230                          TYPE: SINGLE PACKAGE

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

                            File List  (Name or Number)


      HLO APPLICATION REGISTRY
```

DD Export Options Screen:

```
                       File List  (Name or Number)

                            DD Export Options


                       File: HLO APPLICATION REGISTRY


   Send Full or Partial DD...: FULL


Update the Data Dictionary: NO          Send Security Code: NO


Screen to Determine DD Update



      Data Comes With File...: YES
```

Data Export Options Screen – Shows the screen to export only the application's entry, the application only has to edit the name of the entry:

```
                          File List  (Name or Number)

                           DD Export Options

                                Data Export Options


     Site's Data: OVERWRITE


Resolve Pointers: YES             May User Override Data Update: NO


       Data List:


Screen to Select Data
I $P($G(^HLD(779.2,+Y,0)),"^")="HLOZ NMQ"
```

Substitute the name of the application to be transported in place of "HLOZ NMQ".

Currently, KIDS cannot transport the DNS DOMAIN field (# .08) as part of the HL LOGICAL LINK record. This is a known issue with the KIDS package and will be addressed in a future update to KIDS.

As a work-around, HLO developers recommend that a post-install routine be created and attached to any HLO applications distributed via KIDS. The post-install routine should include appropriate FileMan calls to create and/or modify the link in HL LOGICAL LINK File (# 870) at the site. At a minimum, the routine should populate the DNS DOMAIN field (# .08)..

Appendix A – HLO Data Dictionaries

HLO MESSAGE BODY File (#777)

STANDARD DATA DICTIONARY #777 -- HLO MESSAGE BODY FILE

STORED IN ^HLA

Contains the body of an HL7 message, which excludes the message header segment. For batch messages, it does not include the individual message header segments or the batch trailer segment.

POINTED TO BY: MESSAGE BODY field (#.02) of the HLO MESSAGES File (#778)

CROSS REFERENCED BY: DATE/TIME ENTERED(B)

```
777,.01       DATE/TIME ENTERED      0;1 DATE (Required)

              INPUT TRANSFORM:  S %DT="ESTXR" D ^%DT S X=Y K:Y<1 X
              LAST EDITED:      JUL 26, 2004
              HELP-PROMPT:      Enter the exact time that this entry was
                                created.
              CROSS-REFERENCE:  777^B
                                1)= S ^HLA("B",$E(X,1,30),DA)=""
                                2)= K ^HLA("B",$E(X,1,30),DA)


777,.02       BATCH                  0;2 SET

                                '0' FOR NO;
                                '1' FOR YES;
              LAST EDITED:      JUL 26, 2004
              HELP-PROMPT:      Enter YES if this is a batch message.

777,.03       MESSAGE TYPE           0;3 FREE TEXT

              INPUT TRANSFORM:  K:$L(X)>3!($L(X)<3) X
              LAST EDITED:      JUL 26, 2004
              HELP-PROMPT:      Enter the 3 character HL7 message type.

777,.04       EVENT                  0;4 FREE TEXT

              INPUT TRANSFORM:  K:$L(X)>3!($L(X)<3) X
              LAST EDITED:      JUL 26, 2004
              HELP-PROMPT:      Enter the 3 character HL7 event.

777,.05       HL7 VERSION            0;5 FREE TEXT (Required)

              INPUT TRANSFORM:  K:$L(X)>20!($L(X)<1) X
              LAST EDITED:      JUL 26, 2004
              HELP-PROMPT:      Identify the version of the HL7 standard that
                                this message conforms to.

777,.2        HL7 ENCODING CHARACTERS 0;20 FREE TEXT (Required)

              INPUT TRANSFORM:  K:$L(X)>5!($L(X)<5) X
              LAST EDITED:      MAR 17, 2005
              HELP-PROMPT:      The 5 HL7 encoding characters in this order:
```

```
                                Field Separator, Component Separator,
                                Repetition Separator, Escape Character,
                                Subcomponent Separator
                    TECHNICAL DESCR:  No other fields may follow this one on the 0
                                node!


777,1           SEGMENTS (NOT BATCHED) 1;0   WORD-PROCESSING #777.01 (NOWRAP)


777,2           BATCHED MESSAGES       2;0 Multiple #777.02
                                     (Add New Entry without Asking)


777.02,.01        MESSAGE                0;1 NUMBER

                  INPUT TRANSFORM:  K:+X'=X!(X>999999)!(X<1)!(X?.E1"."1N.N) X
                  LAST EDITED:      JUL 26, 2004
                  HELP-PROMPT:      Enter a number to identify the sequence of
                                    this message within the batch, starting with
                                    1, 2, 3, etc.
                  CROSS-REFERENCE:  777.02^B
                                    1)= S ^HLA(DA(1),2,"B",$E(X,1,30),DA)=""
                                    2)= K ^HLA(DA(1),2,"B",$E(X,1,30),DA)


777.02,.02        MESSAGE TYPE           0;2 FREE TEXT (Required)

                  INPUT TRANSFORM:  K:$L(X)>3!($L(X)<3) X
                  LAST EDITED:      JUL 26, 2004
                  HELP-PROMPT:      Enter the 3 character HL7 message type.

777.02,.03        EVENT                  0;3 FREE TEXT (Required)

                  INPUT TRANSFORM:  K:$L(X)>3!($L(X)<3) X
                  LAST EDITED:      JUL 26, 2004
                  HELP-PROMPT:      Enter the 3 character HL7 event type.

777.02,1          MESSAGE SEGMENTS       1;0   WORD-PROCESSING #777.21 (NOWRAP)



777,3           BTS                    3;1 FREE TEXT

                INPUT TRANSFORM:  K:$L(X)>250!($L(X)<1) X
                LAST EDITED:      NOV 16, 2004
                HELP-PROMPT:      Answer must be 1-250 characters in length.
                DESCRIPTION:      For incoming batch messages only, the BTS
                                  segment is stored in this field, which is a
                                  separate node.


INPUT TEMPLATE(S):

PRINT TEMPLATE(S):

SORT TEMPLATE(S):

FORM(S)/BLOCK(S):
```

HLO MESSAGES File (#778)

STANDARD DATA DICTIONARY #778 -- HLO MESSAGES FILE

```
STORED IN ^HLB(

Used to record each message as it is sent or received.  The content of the
message is stored in a file #777, as it might be sent to multiple locations and
applications.
.

CROSS REFERENCED BY: MESSAGE ID(AC), SCHEDULED PURGE DATE/TIME(AD),
                     MESSAGE ID(AE), MESSAGE ID(B), MESSAGE BODY(C)

778,.01       MESSAGE ID             0;1 FREE TEXT (Required)

              INPUT TRANSFORM:  K:$L(X)>20!($L(X)<3) X
              LAST EDITED:      MAY 04, 2005
              HELP-PROMPT:      Answer must be 3-20 characters in length
              CROSS-REFERENCE:  778^B
                                1)= S ^HLB("B",$E(X,1,30),DA)=""
                                2)= K ^HLB("B",$E(X,1,30),DA)

              CROSS-REFERENCE:  778^AC^MUMPS
                                1)= Q
                                2)= Q
                                This x-ref is maintained by the HL7 package.
                                It is used to detect if an incoming message is
                                a duplicate.  The format is: ^HLB("AC"sending
                                facility>_<sending application>_<msg id>,ien)=0



778,.02       MESSAGE BODY           0;2 POINTER TO HLO MESSAGE BODY FILE (#777
                                    ) (Required)

              LAST EDITED:      MAR 17, 2005
              HELP-PROMPT:      Which record in file #777 contains the body of
                                the message?
              CROSS-REFERENCE:  778^C^MUMPS
                                1)= S ^HLB("C",X,DA)=""
                                2)= K ^HLB("C",X,DA)
                                Used to find all the messages that point to the
                                same entry in file 777. Only set for outgoing
                                messages.



778,.03       APPLICATION ACKNOWLEDGMENT TO 0;3 FREE TEXT

              INPUT TRANSFORM:  K:$L(X)>30!($L(X)<3) X
              LAST EDITED:      MAR 17, 2005
              HELP-PROMPT:      Enter the Message Control ID of the message to
                                which this one is an acknowledgment.
              DESCRIPTION:      This is the Message Control ID of the original
                                message to which this message is an application
                                acknowledgment.


778,.04       DIRECTION              0;4 SET (Required)

                                    'I' FOR INCOMING;
```

```
                    'O' FOR OUTGOING;
             LAST EDITED:      NOV 16, 2004
             HELP-PROMPT:      Is the message INCOMING or OUTGOING?

778,.05      LINK                   0;5 FREE TEXT (Required)

             INPUT TRANSFORM:  K:$L(X)>10!($L(X)<3) X
             LAST EDITED:      NOV 16, 2004
             HELP-PROMPT:      Enter the name of the logical link over which
                               the message is being transmitted.

778,.06      QUEUE                  0;6 FREE TEXT

             INPUT TRANSFORM:  K:$L(X)>20!($L(X)<3) X
             LAST EDITED:      JUL 29, 2004
             HELP-PROMPT:      Is the queue on which this message was placed.

778,.07      APPLICATION ACKNOWLEDGMENT BY 0;7 FREE TEXT

             INPUT TRANSFORM:  K:$L(X)>30!($L(X)<3) X
             LAST EDITED:      SEP 30, 2004
             HELP-PROMPT:      If this message has received an application
                               response then enter the responses Message
                               Control ID.

778,.08      REMOTE PORT            0;8 NUMBER

             INPUT TRANSFORM:  K:+X'=X!(X>65535)!(X<1)!(X?.E1"."1.N) X
             LAST EDITED:      MAY 04, 2005
             HELP-PROMPT:      Enter the remote port indicated in the Facility
                               field of the message header.
             DESCRIPTION:      This is the remote port # that may be found in
                               the message header. For outgoing messages, it
                               is in the Receiving Facility field, for
                               incoming messages it is in tthe Sending
                               Facility. For application acknowledgments, the
                               port provided in the original message is used
                               to return the acknowledgment.


778,.09      SCHEDULED PURGE DATE/TIME 0;9 DATE

             INPUT TRANSFORM:  S %DT="ESTXR" D ^%DT S X=Y K:Y<1 X
             LAST EDITED:      AUG 19, 2004
             HELP-PROMPT:      When can this message be purged?
             CROSS-REFERENCE:  778^AD^MUMPS
                               1)= Q
                               2)= Q
                               3)= DO NOT DELETE!
                               This cross-reference will be used to control
                               the purging process.  It will be maintained
                               within the HL7 package and will not be set via
                               Fileman.  The format is: ^HLB("AD",<"IN" or
                               "OUT">,<dt/tm for purging>,<message ien>)="


778,.1       APPLICATION ACK RSPNS TAG 0;10 FREE TEXT

             INPUT TRANSFORM:  K:$L(X)>8!($L(X)<1) X
             LAST EDITED:      JUL 27, 2004
             HELP-PROMPT:      Answer must be 1-8 characters in length.
             DESCRIPTION:      The sending application routine to execute when
```

```
                              the application ack is received.

                              This is part one of a two-part field which is
                              the entry point.  The second part names the
                              routine.


778,.11       APPLICATION ACK RSPNS RTN 0;11 FREE TEXT

              INPUT TRANSFORM:  K:$L(X)>8!($L(X)<1) X
              LAST EDITED:      JUL 27, 2004
              HELP-PROMPT:      Answer must be 1-8 characters in length.
              DESCRIPTION:      Answer must be 1-8 characters in length. The
                                sending application routine to execute when the
                                application ack is received.

                                This is part two of a two-part field which is
                                the routine name.  The first part names the
                                entry point.


778,.12       ACCEPT ACK RSPNS TAG   0;12 FREE TEXT

              INPUT TRANSFORM:  K:$L(X)>8!($L(X)<1) X
              LAST EDITED:      NOV 16, 2004
              HELP-PROMPT:      Answer must be 1-8 characters in length.
              DESCRIPTION:      The sending application's routine to execute
                                when the transmission of the message fails,
                                i.e., the message cannot be sent or no
                                acceptack is received.

                                This is part one of a two-part field, naming
                                the entry point of the routine to be called.
                                The second part names the routine.



778,.13       ACCEPT ACK RESPNS RTN  0;13 FREE TEXT

              INPUT TRANSFORM:  K:$L(X)>8!($L(X)<1) X
              LAST EDITED:      NOV 16, 2004
              HELP-PROMPT:      Answer must be 1-8 characters in length.
              DESCRIPTION:      The sending application routine to execute whe
                                the commit ack is received.

                                This is part two of a two-part field,
                                consisting of the routine name. The first part
                                names the entry point with the M routine.



778,.14       TRANSMISSION FAILURE RSPNS TAG 0;14 FREE TEXT

              INPUT TRANSFORM:  K:$L(X)>8!($L(X)<1) X
              LAST EDITED:      JUL 27, 2004
              HELP-PROMPT:      Answer must be 1-8 characters in length.
              DESCRIPTION:      The sending application's routine to execute
                                when the transmission of the message fails,
                                i.e., the message can not be sent or no commit
                                ack is received.
```

```
                          This is part one of a two-part field which is
                          the entry point.  The second part names the
                          routine.

778,.15       TRANSMISSION FAILURE RSPNS RTN 0;15 FREE TEXT

              INPUT TRANSFORM:  K:$L(X)>8!($L(X)<1) X
              LAST EDITED:      JUL 27, 2004
              HELP-PROMPT:      Answer must be 1-8 characters in length.
              DESCRIPTION:      The sending application's routine to execute
                                when the transmission of the message fails,
                                i.e., the message can not be sent or no commit
                                ack is received.

                                This is part two of a two-part field which is
                                the routine's name.  The first part names the
                                entry point.

778,.16       TRANSMISSION DATE/TIME 0;16 DATE

              INPUT TRANSFORM:  S %DT="ESTXR" D ^%DT S X=Y K:Y<1 X
              LAST EDITED:      JUL 28, 2004
              DESCRIPTION:      This is the date and time that the message was
                                either received or sent.

778,.17       ACCEPT ACK'D           0;17 SET

                                '1' FOR YES;
              LAST EDITED:      AUG 23, 2004
              HELP-PROMPT:      Enter 1 if an accept ack was sent or received.

778,.18       APPLICATION ACK'D      0;18 SET

                                '1' FOR YES;
              LAST EDITED:      NOV 16, 2004
              DESCRIPTION:      For incoming messages, this flag indicates that
                                an application ack was returned.  For outgoing
                                messages, this field indicates that the
                                application ack was sent.

778,.19       APPLICATION HANDOFF    0;19 SET

                                '1' FOR YES;
              LAST EDITED:      JUL 28, 2004
              DESCRIPTION:      This flag indicates that this message was
                                handed to the application.  That may be for
                                initial processing, or it may be in response to
                                one of the other conditions that an application
                                may register its need to respond, such as a
                                failure of the remote system to accept the
                                message.

778,.2        COMPLETION STATUS      0;20 SET

                                'SU' FOR SUCCESSFUL;
                                'TF' FOR TRANSMISSION FAILURE;
                                'SE' FOR RECEIVING SYSTEM DETERMINED ERROR;
                                'AE' FOR RECEIVING APPLICATION DETERMINED ERROR
```

```
                        ;
               LAST EDITED:      AUG 20, 2004
               HELP-PROMPT:      Enter the code that indicates the final status
                                 of the message.
               DESCRIPTION:      This field indicates the final status of the
                                 message.  Any code but SU (SUCCESSFUL)
                                 indicates that an error occurred.  No value
                                 indicates that the message has not completed.


778,.21        ERROR TEXT             0;21 FREE TEXT

               INPUT TRANSFORM:  K:$L(X)>30!($L(X)<1) X
               LAST EDITED:      SEP 28, 2004
               HELP-PROMPT:      Answer must be 1-30 characters in length.
               DESCRIPTION:      The HL7 package may use this field to document
                                 errors that prevent transmission.  Errors
                                 determined by the remote system are contained
                                 in the MSA segment of the response message.


778,1          HDR SEGMENT,COMPONENTS 1-6 1;E1,250 FREE TEXT (Required)

               INPUT TRANSFORM:  K:$L(X)>250!($L(X)<8) X
               LAST EDITED:      JUL 29, 2004
               HELP-PROMPT:      The first 6 components of the message header
                                 segment.

778,2          HDR SEGMENT,COMPONENTS 7-END 2;E1,250 FREE TEXT (Required)

               INPUT TRANSFORM:  K:$L(X)>250!($L(X)<15) X
               LAST EDITED:      NOV 16, 2004
               HELP-PROMPT:      Enter the header segment begining with
                                 component 7

778,3          MSH SEGMENTS FOR BATCH 3;0 Multiple #778.03
                                  (Add New Entry without Asking)


778.03,.01       MESSAGE IN BATCH       0;1 NUMBER

                 INPUT TRANSFORM:  K:+X'=X!(X>99999)!(X<1)!(X?.E1"."1N.N) X
                 LAST EDITED:      JUL 26, 2004
                 HELP-PROMPT:      Enter a number to sequence each message
                                   within the batch, starting with 1,2,3,...etc.
                 CROSS-REFERENCE:  778.03^B
                                   1)= S ^HLB(DA(1),3,"B",$E(X,1,30),DA)=""
                                   2)= K ^HLB(DA(1),3,"B",$E(X,1,30),DA)


778.03,.02       MESSAGE ID             0;2 FREE TEXT (Required)

                 INPUT TRANSFORM:  K:$L(X)>30!($L(X)<1) X
                 LAST EDITED:      OCT 06, 2004
                 HELP-PROMPT:      Answer must be 1-30 characters in length.
                 CROSS-REFERENCE:  778^AE^MUMPS
                                   1)= Q
                                   2)= Q
                                   This cross-reference is maintained outside of
                                   Fileman. It is for finding individual
                                   messages within a batch using the individual
                                   Message Control ID.  Its format is:
```

```
                              "AE",<Message Control ID>,<ien, file
                              778>^<subien>)=""


778.03,.03      APPLICATION ACKNOWLEDGMENT TO 0;3 FREE TEXT

                INPUT TRANSFORM:  K:$L(X)>30!($L(X)<3) X
                LAST EDITED:      SEP 28, 2004
                HELP-PROMPT:      Enter the Message Control ID of the message
                                  to which this one is a response.
                DESCRIPTION:      This field is completed only if this message
                                  is an application acknowledgment.


778.03,.04      APPLICATION ACKNOWLEDGMENT BY 0;4 FREE TEXT

                INPUT TRANSFORM:  K:$L(X)>30!($L(X)<3) X
                LAST EDITED:      SEP 29, 2004
                HELP-PROMPT:       Enter the Message Control ID of the
                                  application response.

778.03,.05      COMPLETION STATUS      0;5 SET

                                  'SU' FOR SUCCESS;
                                  'AE' FOR APPLICATION ERROR;
                LAST EDITED:      SEP 30, 2004
                HELP-PROMPT:      Enter only if an application acknowledgment
                                  is received. SU is for successfully completed
                                  messages, AE if an error is returned.

778.03,1        MSH SEGMENT, COMPONENTS 1-6 1;E1,250 FREE TEXT (Required)

                INPUT TRANSFORM:  K:$L(X)>250!($L(X)<8) X
                LAST EDITED:      JUL 29, 2004
                HELP-PROMPT:      Answer must be 8-250 characters in length.

778.03,2        MSH SEGMENT, COMPONENTS 7-END 2;E1,250 FREE TEXT

                INPUT TRANSFORM:  K:$L(X)>250!($L(X)<15) X
                LAST EDITED:      JUL 29, 2004
                HELP-PROMPT:      Answer must be 15-250 characters in length.


778,4.01      DATE/TIME OF ACCEPT ACK 4;1 DATE

              INPUT TRANSFORM:  S %DT="ESTXR" D ^%DT S X=Y K:Y<1 X
              LAST EDITED:      SEP 28, 2004
              HELP-PROMPT:      Enter the date and time of the ack.
              DESCRIPTION:      This field will be used to record the accept
                                ack.


778,4.02      ACCEPT ACK MESSAGE ID  4;2 FREE TEXT

              INPUT TRANSFORM:  K:$L(X)>30!($L(X)<1) X
              LAST EDITED:      SEP 28, 2004
              HELP-PROMPT:      Answer must be 1-30 characters in length.

778,4.03      ACCEPT ACK MSA SEGMENT 4;3 FREE TEXT

              INPUT TRANSFORM:  K:$L(X)>210!($L(X)<3) X
```

```
              LAST EDITED:      SEP 28, 2004
              HELP-PROMPT:      Answer must be 3-210 characters in length.


      FILES POINTED TO                      FIELDS

HLO MESSAGE BODY (#777)           MESSAGE BODY (#.02)



INPUT TEMPLATE(S):

PRINT TEMPLATE(S):

SORT TEMPLATE(S):

FORM(S)/BLOCK(S):
HLOL                           MAR 10, 2005@14:17  USER #0
  HLOLHEADER                     DD #101
  HLOLDATA                       DD #101
```

HLO SYSTEM PARAMETERS File (#779.1)

STANDARD DATA DICTIONARY #779.1 -- HLO SYSTEM PARAMETERS FILE

.

STORED IN ^HLD(779.1,

This file contains parameters used by the HLO (HL7 Optimized) that are specific to the system the software is installed on.

CROSS REFERENCED BY: DOMAIN NAME(B)

779.1,.01 DOMAIN NAME 0;1 FREE TEXT (Required)

INPUT TRANSFORM: K:$L(X)>64!($L(X)<3)!'(X'?1P.E) X
LAST EDITED: AUG 05, 2004
HELP-PROMPT: The domain name for this system. It will be used to populate component 2 of the Sending Facility field of the HL7 message header.
DESCRIPTION: The domain name for this system. It will be used to populate component 2 of the Sending Facility field of the HL7 message headers.

CROSS-REFERENCE: 779.1^B
1)= S ^HLD(779.1,"B",$E(X,1,60),DA)=""
2)= K ^HLD(779.1,"B",$E(X,1,60),DA)

779.1,.02 STATION NUMBER 0;2 FREE TEXT

INPUT TRANSFORM: K:$L(X)>7!($L(X)<3) X
LAST EDITED: AUG 05, 2004
HELP-PROMPT: Enter the station number with suffix that this system belongs under. It will be used in component 1 of the Sending Facility field of the HL7 message header.

779.1,.03 PRODUCTION ID 0;3 SET (Required)

'P' FOR production;
'T' FOR training;
LAST EDITED: AUG 05, 2004
HELP-PROMPT: ENTER P if this is a production system, T otherwise.

779.1,.04 MAXIMUM STRING LENGTH 0;4 NUMBER

INPUT TRANSFORM: K:+X'=X!(X>99999)!(X<1)!(X?.E1"."1N.N) X
LAST EDITED: JUL 06, 2005
HELP-PROMPT: This is the maximum length for strings built by HLO when local applications create new messages to send.
DESCRIPTION: This parameter determines the maximum length for strings that HLO will create when messages are being built. It doesn't apply to servers, as the size of input buffer used by TCP/IP determines the maximum string length created by a single read.

779.1,.05 BUFFER SIZE FOR HL7 (BYTES) 0;5 NUMBER

```
             INPUT TRANSFORM:  K:+X'=X!(X>20000)!(X<10000)!(X?.E1"."1.N) X
             LAST EDITED:      AUG 05, 2005
             HELP-PROMPT:      This parameter represents the size of the
                               buffer used by HLO for its background
                               processes.  It defaults to 15000 bytes, but may
                               be set from 10,000 bytes to 20,000 bytes.

779.1,.06    BUFFER SIZE FOR USER (BYTES) 0;6 NUMBER

             INPUT TRANSFORM:  K:+X'=X!(X>10000)!(X<512)!(X?.E1"."1.N) X
             LAST EDITED:      AUG 05, 2005
             HELP-PROMPT:      This parameter is the size of the buffer used
                               by HLO in the context of an online user.  It
                               defaults to 5000, but may be reset to between
                               512 and 10000 bytes.

779.1,.07    NORMAL MSG RETENTION (HOURS) 0;7 NUMBER

             INPUT TRANSFORM:  K:+X'=X!(X>96)!(X<36)!(X?.E1"."1.N) X
             LAST EDITED:      MAR 17, 2005
             HELP-PROMPT:      How many hours should successfully completed
                               messages remain on your system? (36-96 hours,
                               defaults to 36 hours)
             DESCRIPTION:      This field controls the purging of HL7 messages
                               whose completion status is SUCCESSFUL. It is in
                               hours, since messages normally should be purged
                               very soon after completion, with an allowed
                               range of 36 to 94 hours.

                               36 is the default because this 1) will result
                               in most messages being purged at night and 2)
                               provides sufficient time for the Capacity
                               Planning statistics to be extracted.


779.1,.08    BAD MESSAGE RETENTION (DAYS) 0;8 NUMBER

             INPUT TRANSFORM:  K:+X'=X!(X>45)!(X<5)!(X?.E1"."1.N) X
             LAST EDITED:      NOV 15, 2004
             HELP-PROMPT:      How many days should message with errors remain
                               on your system?  (7-45 days, defaults to 7
                               days)
             DESCRIPTION:      This field controls the purging of HL7 messages
                               that do not complete successfully.  The period
                               should be reasonably long to allow
                               investigation, but because of the extremely
                               high daily volume of messages purging must
                               occur quickly.


779.1,.09    HLO ON/OFF SWITCH      0;9 SET

                               '0' FOR OFF;
                               '1' FOR ON;
             LAST EDITED:      MAY 03, 2005
             HELP-PROMPT:      Set to 0 to turn off messaging and all HL7
                               processes.

779.1,.1     HLO STANDARD LISTENER  0;10 POINTER TO HL LOGICAL LINK FILE (#870
                               )

             INPUT TRANSFORM:  S DIC("S")="I ($P($G(^HLCS(870,Y,400)),""^"",3)
                               =""M"")!($P($G(^HLCS(870,Y,400)),""^"",3)=""S""
```

```
                        )" D ^DIC K DIC S DIC=$G(DIE),X=+Y K:Y<0 X
               LAST EDITED:     JUN 01, 2005
               HELP-PROMPT:     Select an entry from the HL Logical Link file
                                that is the listener that remote applications
                                will normally connect to.
               SCREEN:          S DIC("S")="I ($P($G(^HLCS(870,Y,400)),""^"",3)
                                =""M"")!($P($G(^HLCS(870,Y,400)),""^"",3)=""S""
                                )"
               EXPLANATION:     This screen allows only server entries to be se
                                lected.


      FILES POINTED TO                      FIELDS

HL LOGICAL LINK (#870)            HLO STANDARD LISTENER (#.1)


INPUT TEMPLATE(S):

PRINT TEMPLATE(S):

SORT TEMPLATE(S):

FORM(S)/BLOCK(S):
```

HLO APPLICATION REGISTRY File (#779.2)

```
STANDARD DATA DICTIONARY #779.2 -- HLO APPLICATION REGISTRY FILE
.
STORED IN ^HLD(779.2,

This file is used to register sending and receiving applications for HL7
messaging. For receiving applications, the process of registration consists of
registering what messages the application is prepared to receive.

For both sending and receiving applications, it is necessary to specify what
package the application belongs to.  For sending applications, that is the only
field that applies, other than the name of the sending application.

An application can be either a sender or a receiver of messages, or both. In
order for an application to receive messages, it must specify an action (M
tag^routine) for each type of message that it is capable of receiving, or a
default action that applies when no messsage-specific action is defined.


PRIMARY KEY:        A (#42)
  Uniqueness Index: C (#429)
       File, Field: 1) APPLICATION NAME (779.2,.01)

CROSS REFERENCED BY: APPLICATION NAME(B)

INDEXED BY:    APPLICATION NAME (C)


779.2,.01     APPLICATION NAME       0;1 FREE TEXT (Required) (Key field)

              INPUT TRANSFORM:  K:$L(X)>60!($L(X)<3)!'(X'?1P.E) X
              LAST EDITED:      JAN 25, 2005
              HELP-PROMPT:      Answer must be 3-60 characters in length. It
                                must be unique and should be name-spaced.
              CROSS-REFERENCE:  779.2^B
                                1)= S ^HLD(779.2,"B",$E(X,1,30),DA)=""
                                2)= K ^HLD(779.2,"B",$E(X,1,30),DA)

              FIELD INDEX:      C (#429)    REGULAR    IR    LOOKUP & SORTING
                   Unique for:  Key A (#42), File #779.2
                  Short Descr:  Uniqueness Index for Key 'A' of File #779.2
                    Set Logic:  S ^HLD(779.2,"C",X,DA)=""
                   Kill Logic:  K ^HLD(779.2,"C",X,DA)
                   Whole Kill:  K ^HLD(779.2,"C")
                         X(1):  APPLICATION NAME  (779.2,.01)  (Subscr 1)


779.2,.02     RESPONSE LINK (OPTIONAL) 0;2 FREE TEXT

              INPUT TRANSFORM:  K:'$$CHKLINK^HLOTLNK(X) X
              LAST EDITED:      NOV 16, 2004
              HELP-PROMPT:      If the return link cannot be identified via the
                                Sending Facility (i.e., sent via an IE), what
                                link should the application ack be sent
                                through?
              DESCRIPTION:      This field applies only if: 1) The receiving
                                application is expected to return application
                                acknowledgments.  2) The initial message is
                                received indirectly through the IE, and the
                                receiving application in turn does not want to
                                send the application acknowledgment directly
```

```
                        back to the sending facility identified in the
                        message header.

              NOTES:    XXXX--CAN'T BE ALTERED EXCEPT BY PROGRAMMER


779.2,.03     DEFAULT PRIVATE IN-QUEUE 0;3 FREE TEXT

              INPUT TRANSFORM:  K:$L(X)>20!($L(X)<3) X
              LAST EDITED:      MAR 17, 2005
              HELP-PROMPT:      You may create an optional default private
                                in-queue by entering a unique name up to 20
                                characters in length. Queues specified for
                                specific message types take precedence.

779.2,.04     BATCH ACTION TAG       0;4 FREE TEXT

              INPUT TRANSFORM:  K:$L(X)>8!($L(X)<1) X
              LAST EDITED:      AUG 14, 2004
              HELP-PROMPT:      If the application utilizes batch messages, the
                                action to perform upon receipt of the message
                                should be entered in the BATCH ACTION TAG and
                                BATCH ACTION ROUTINE fields as <tag>^<routine>.

779.2,.05     BATCH ACTION ROUTINE   0;5 FREE TEXT

              INPUT TRANSFORM:  K:$L(X)>8!($L(X)<3) X
              LAST EDITED:      AUG 14, 2004
              HELP-PROMPT:      If the application utilizes batch messages, the
                                action to perform upon receipt of the message
                                should be entered in the BATCH ACTION TAG and
                                BATCH ACTION ROUTINE fields as <tag>^<routine>.

779.2,.06     DEFAULT ACTION TAG     0;6 FREE TEXT

              INPUT TRANSFORM:  K:$L(X)>8!($L(X)<1) X
              LAST EDITED:      AUG 15, 2004
              HELP-PROMPT:      You can enter the action to perform upon
                                receipt of a message where no other action
                                applies by entering the DEFAULT ACTION TAG and
                                DEFAULT ACTION ROUTINE fields as
                                <tag>^<routine>.

779.2,.07     DEFAULT ACTION ROUTINE 0;7 FREE TEXT

              INPUT TRANSFORM:  K:$L(X)>8!($L(X)<3) X
              LAST EDITED:      AUG 14, 2004
              HELP-PROMPT:      You can enter the action to perform upon
                                receipt of a message where no other action
                                applies by entering the DEFAULT ACTION TAG and
                                DEFAULT ACTION ROUTINE fields as
                                <tag>^<routine>.

779.2,.08     BATCH PRIVATE IN-QUEUE 0;8 FREE TEXT

              INPUT TRANSFORM:  K:$L(X)>20!($L(X)<3) X
              LAST EDITED:      MAR 17, 2005
              HELP-PROMPT:      You may establish a private queue for your
                                batch messages by entering a unique name
                                (name-spaced) up to 20 characters long.

779.2,.09     APPLICATION SPECIFIC LISTENER 0;9 POINTER TO HL LOGICAL LINK FILE
                          (#870)
```

```
              INPUT TRANSFORM:  S DIC("S")="I $E($P(^HLCS(870,Y,0),""^"",4),2)=
                                ""S""" D ^DIC K DIC S DIC=$G(DIE),X=+Y K:Y<0 X
              LAST EDITED:      MAY 27, 2005
              HELP-PROMPT:      If your application requires its own listener
                                (HIGHLY DISCOURAGED), enter it here.
              DESCRIPTION:      Applications are highly discouraged from
                                establishing their own listeners.  The use of
                                the multi-listeners provide concurrent
                                processing of many connections over the same
                                port, so a dedicated listener will not provide
                                an application with a performance boost, while
                                it will cause the site additional work to
                                maintain. So before establishing a dedicated
                                listener, the application developer should
                                verify the need.

              SCREEN:           S DIC("S")="I $E($P(^HLCS(870,Y,0),""^"",4),2)=
                                ""S"""
              EXPLANATION:      The link entered must be a listener.

779.2,1       MESSAGE TYPE ACTIONS   1;0 Multiple #779.21


              PRIMARY KEY:      A (#43)
              Uniqueness Index: C (#430)
                   File, Field: 1) HL7 MESSAGE TYPE (779.21,.01)  2) HL7 EVENT
                                (779.21,.02)

              INDEXED BY:       HL7 MESSAGE TYPE & HL7 EVENT (C)

779.21,.01      HL7 MESSAGE TYPE       0;1 FREE TEXT (Multiply asked)
                                     (Key field)

                INPUT TRANSFORM:  K:$L(X)>3!($L(X)<3) X
                LAST EDITED:      AUG 15, 2004
                HELP-PROMPT:      Enter the 3 character HL7 Message Type.
                DESCRIPTION:      An application should use this multiple to
                                  define the action that the receiving
                                  application needs to perform upon receipt of
                                  a specific type of HL7 message, identified by
                                  the HL7 MESSAGE TYPE and HL7 EVENT fields.

                CROSS-REFERENCE:  779.21^B
                                  1)= S ^HLD(779.2,DA(1),1,"B",$E(X,1,30),DA)="
                                  "

                                  2)= K ^HLD(779.2,DA(1),1,"B",$E(X,1,30),DA)

                RECORD INDEXES:   C (#430)

779.21,.02      HL7 EVENT              0;2 FREE TEXT (Required) (Key field)

                INPUT TRANSFORM:  K:$L(X)>3!($L(X)<3) X
                LAST EDITED:      AUG 15, 2004
                HELP-PROMPT:      Enter the 3 character HL7 event type.
                RECORD INDEXES:   C (#430)

779.21,.03      PRIVATE IN-QUEUE       0;3 FREE TEXT

                INPUT TRANSFORM:  K:$L(X)>20!($L(X)<3) X
                LAST EDITED:      MAR 17, 2005
                HELP-PROMPT:      You may create a private in-queue for message
```

```
                                of this type by entering a unique name up to
                                20 characters long..

779.21,.04      ACTION TAG            0;4 FREE TEXT

                INPUT TRANSFORM:  K:$L(X)>8!($L(X)<1) X
                LAST EDITED:      AUG 15, 2004
                HELP-PROMPT:      You must enter the action to perform upon
                                  receipt of this type by entering the ACTION
                                  TAG and ACTION ROUTINE fields as
                                  <tag>^<routine>. The tag is optional.

779.21,.05      ACTION ROUTINE        0;5 FREE TEXT (Required)

                INPUT TRANSFORM:  K:$L(X)>8!($L(X)<3) X
                LAST EDITED:      AUG 14, 2004
                HELP-PROMPT:      You must enter the action to perform upon
                                  receipt of this type by entering the ACTION
                                  TAG and ACTION ROUTINE fields as
                                  <tag>^<routine>.



779.2,2         Package File Link     2;1 POINTER TO PACKAGE FILE (#9.4)
                                     (Required)

                LAST EDITED:      JAN 05, 2005
                HELP-PROMPT:      Enter the package responsible for these
                                  messages.
                DESCRIPTION:      This field holds a pointer to the Package File
                                  for the Package responsible for thes messages.


      FILES POINTED TO                     FIELDS

HL LOGICAL LINK (#870)            APPLICATION SPECIFIC LISTENER (#.09)

PACKAGE (#9.4)                    Package File Link (#2)


Subfile #779.21

  Record Indexes:

  C (#430)    RECORD    REGULAR    IR    LOOKUP & SORTING
       Unique for:  Key A (#43), File #779.21
      Short Descr:  Uniqueness Index for Key 'A' of Subfile #779.21
        Set Logic:  S ^HLD(779.2,DA(1),1,"C",X(1),X(2),DA)=""
       Kill Logic:  K ^HLD(779.2,DA(1),1,"C",X(1),X(2),DA)
       Whole Kill:  K ^HLD(779.2,DA(1),1,"C")
             X(1):  HL7 MESSAGE TYPE  (779.21,.01)  (Subscr 1)
             X(2):  HL7 EVENT  (779.21,.02)  (Subscr 2)


INPUT TEMPLATE(S):
HLOAPREG                       MAY 27, 2005@11:12  USER #0

PRINT TEMPLATE(S):

SORT TEMPLATE(S):

FORM(S)/BLOCK(S):
```

HLO PROCESS REGISTRY File (#779.3)

```
STANDARD DATA DICTIONARY #779.3 -- HLO PROCESS REGISTRY FILE
.
STORED IN ^HLD(779.3,

The process registry is used by the HLO process manager to start, stop, and
manage all of the processes used by the HLO system.


PRIMARY KEY:        A (#44)
  Uniqueness Index: D (#432)
       File, Field: 1) PROCESS NAME (779.3,.01)

CROSS REFERENCED BY: PROCESS NAME(B)

INDEXED BY:    ACTIVE (C), PROCESS NAME (D)


779.3,.01     PROCESS NAME           0;1 FREE TEXT (Required) (Key field)

              INPUT TRANSFORM:  K:$L(X)>30!(X?.N)!($L(X)<3)!'(X'?1P.E) X
              LAST EDITED:      NOV 15, 2004
              HELP-PROMPT:      Give the type of process a unique name, 3-30
                                characters.
              DESCRIPTION:
                                A unique name for the type of process.

              CROSS-REFERENCE:  779.3^B
                                1)= S ^HLD(779.3,"B",$E(X,1,30),DA)=""
                                2)= K ^HLD(779.3,"B",$E(X,1,30),DA)

              FIELD INDEX:      D (#432)    REGULAR    IR    LOOKUP & SORTING
                   Unique for:  Key A (#44), File #779.3
                  Short Descr:  Uniqueness Index for Key 'A' of File #779.3
                    Set Logic:  S ^HLD(779.3,"D",X,DA)=""
                   Kill Logic:  K ^HLD(779.3,"D",X,DA)
                   Whole Kill:  K ^HLD(779.3,"D")
                         X(1):  PROCESS NAME  (779.3,.01)  (Subscr 1)


779.3,.02     ACTIVE                 0;2 SET

                                '0' FOR NO;
                                '1' FOR YES;
              LAST EDITED:      AUG 05, 2005
              HELP-PROMPT:      Enter 1 to activate these processes, 0 to
                                inactivate.
              DESCRIPTION:      A flag that indicates whether or not this type
                                of process is active under the HLO Process
                                Manager. Some processes may not apply to some
                                systems, for example, a particular site may not
                                use the Taskman multi-listener.

              FIELD INDEX:      C (#431)    REGULAR    IR    LOOKUP & SORTING
                  Short Descr:  Used to find active entries in the process
                                registry.
                    Set Logic:  S ^HLD(779.3,"C",X,DA)=""
                     Set Cond:  S X=$S('X(1):0,1:1)
                   Kill Logic:  K ^HLD(779.3,"C",X,DA)
                   Whole Kill:  K ^HLD(779.3,"C")
                         X(1):  ACTIVE  (779.3,.02)  (Subscr 1)  (forwards)
```

779.3,.03 MINIMUM ACTIVE PROCESSES 0;3 NUMBER

```
INPUT TRANSFORM:  K:+X'=X!(X>99)!(X<0)!(X?.E1"."1.N) X
LAST EDITED:      AUG 05, 2005
HELP-PROMPT:      How many of these processes should be running
                  at a minimum when HL7 messaging is on?
DESCRIPTION:      This field indicates the minimum number of
                  concurrent processes of this type.  The exact
                  number changes as the HLO Process Manager
                  starts and stops processes in response to
                  changes in workload.
```

779.3,.04 MAXIMUM ACTIVE PROCESSES 0;4 NUMBER

```
INPUT TRANSFORM:  K:+X'=X!(X>999)!(X<1)!(X?.E1"."1.N) X
LAST EDITED:      SEP 07, 2004
HELP-PROMPT:      How many of these processes should be running
                  at a maximum when the HL7 messaging system is
                  on?
```

779.3,.05 SCHEDULING FREQUENCY (minutes) 0;5 NUMBER

```
INPUT TRANSFORM:  K:+X'=X!(X>9999)!(X<0)!(X?.E1"."1N.N) X
LAST EDITED:      DEC 21, 2004
HELP-PROMPT:      This is how long the Process Manager should
                  wait between checks to see ifanother process of
                  this type should be started.  Enter 0 to 9999.
DESCRIPTION:      This is how long the Process Manager should
                  wait between checks to see if another process
                  of this type should be started.
```

779.3,.06 DT/TM LAST STARTED OR STOPPED 0;6 DATE

```
INPUT TRANSFORM:  S %DT="ESTR" D ^%DT S X=Y K:Y<1 X
LAST EDITED:      NOV 15, 2004
HELP-PROMPT:      DT/TM the process manager last started or
                  stopped one of these.
DESCRIPTION:      The date and time when a process of this type
                  was last started or stopped.
```

779.3,.07 HANG TIME (seconds) 0;7 NUMBER

```
INPUT TRANSFORM:  K:+X'=X!(X>999)!(X<0)!(X?.E1"."1.N) X
LAST EDITED:      NOV 15, 2004
HELP-PROMPT:      If the process cannot find work, how many
                  seconds should it hang before looking again?
DESCRIPTION:      This is how long a process should wait between
                  attempts to find work to do.
```

779.3,.08 GET WORK FUNCTION (TAG) 0;8 FREE TEXT (Required)

```
INPUT TRANSFORM:  K:$L(X)>8!($L(X)<1) X
LAST EDITED:      NOV 15, 2004
HELP-PROMPT:      What function will this process call to find
                  work (optional routine entry point)
DESCRIPTION:      The M entry point to the GET WORK function for
                  this process type.
```

```
779.3,.09     GET WORK FUNCTION (ROUTINE) 0;9 FREE TEXT (Required)

              INPUT TRANSFORM:  K:$L(X)>8!($L(X)<1) X
              LAST EDITED:      NOV 15, 2004
              HELP-PROMPT:      What function should this process call to find
                                work? (routine name only)
              DESCRIPTION:      The routine in which this process type has
                                located its GET WORK function.


779.3,.1      DO WORK FUNCTION (TAG) 0;10 FREE TEXT

              INPUT TRANSFORM:  K:$L(X)>8!($L(X)<1) X
              LAST EDITED:      NOV 15, 2004
              HELP-PROMPT:      What function should this process call to do
                                its work? Enter the optional entry point only
                                in this field.
              DESCRIPTION:      The M entry point for the process's DO WORK
                                function.


779.3,.11     DO WORK FUNCTION (ROUTINE) 0;11 FREE TEXT

              INPUT TRANSFORM:  K:$L(X)>8!($L(X)<1) X
              LAST EDITED:      NOV 15, 2004
              HELP-PROMPT:      What function should this process call to do
                                its work?  Enter the routine name only in this
                                field.
              DESCRIPTION:      The routine in which the process's DO WORK
                                function is located.


779.3,.12     MAX TRIES FINDING WORK 0;12 NUMBER

              INPUT TRANSFORM:  K:+X'=X!(X>9999)!(X<0)!(X?.E1"."1.N) X
              LAST EDITED:      NOV 15, 2004
              HELP-PROMPT:      How many times should the process look for work
                                before giving up?
              DESCRIPTION:      How many times should the process look for work
                                and then quit if it cannot find anything to do?
                                It'll hang between attempts the specified
                                length of time.


779.3,.13     PERSISTENT            0;13 SET

                                    '0' FOR NO;
                                    '1' FOR YES;
              LAST EDITED:      NOV 15, 2004
              HELP-PROMPT:      Should processes of this type be restarted
                                automatically if they die?
              DESCRIPTION:      Setting this field to YES results in the
                                process being made persistent via the Taskman
                                persistent parameter.


779.3,.14     DEDICATED LINK        0;14 FREE TEXT

              INPUT TRANSFORM:  K:$L(X)>10!($L(X)<3) X
              LAST EDITED:      JUL 06, 2005
              HELP-PROMPT:      If this process is a listener, you must enter
```

```
                        the name of an HL LOGICAL LINK that is a
                        listener and whose TCP/IP PORT (OPTIMIZED)
                        field contains the correct port number.
              DESCRIPTION:      The primary use of this field is for TCP/IP
                        listener processes, and indicates which port
                        (via the HL Logical Link) that the process
                        should be listening on.  However, it could be
                        used to dedicate a client link process to a
                        particular link.

              NOTES:            XXXX--CAN'T BE ALTERED EXCEPT BY PROGRAMMER


779.3,.15     VMS TCP SERVICE        0;15 SET

                        '1' FOR YES;
                        '0' FOR NO;
              LAST EDITED:      SEP 23, 2004
              HELP-PROMPT:      If this process is a listener, enter YES if it
                        is a VMS TCP service rather than a Taskman
                        process.
              DESCRIPTION:      VMS services are not started or stopped via the
                        HL7 Process Manager.  However, on a VMS system,
                        these services are an important part of the HL7
                        system, and so an entry in the HL7 Process
                        Registry should be created for them.  The
                        Process Manager will use the PING functionality
                        to verify that the listener is running.


INPUT TEMPLATE(S):

PRINT TEMPLATE(S):

SORT TEMPLATE(S):

FORM(S)/BLOCK(S):
```

HLO SUBSCRIPTION REGISTRY File (#779.4)

STANDARD DATA DICTIONARY #779.4 -- HLO SUBSCRIPTION REGISTRY FILE

.

STORED IN ^HLD(779.4,

This file is used to store static routing lists for messages.

Static routing lists are lists of recipients that an application may create in advance for its messages. The alternate routing method is dynamic routing, whereby the recipient list is created by the application at the time the message is created.

```
            DD ACCESS: @
            RD ACCESS: @
            WR ACCESS: @
           DEL ACCESS: @
         LAYGO ACCESS: @
         AUDIT ACCESS: @
```

CROSS REFERENCED BY: OWNER(AH), NAME(B)

779.4,.01	NAME	0;1 FREE TEXT (Required)
	INPUT TRANSFORM:	K:+X'=X!(X>99999999)!(X<1)!(X?.E1"."1N.N) X S:$ D(X) DINUM=X
	LAST EDITED:	MAR 08, 2005
	HELP-PROMPT:	NAME MUST BE 3-30 CHARACTERS, NOT NUMERIC OR STARTING WITH PUNCTUATION
	DESCRIPTION:	This file should not be used to edit this file. Instead, a set of developer APIs that was released with this file should be used.
	NOTES:	XXXX--CAN'T BE ALTERED EXCEPT BY PROGRAMMER
	CROSS-REFERENCE:	779.4^B 1)= S ^HLD(779.4,"B",$E(X,1,30),DA)="" 2)= K ^HLD(779.4,"B",$E(X,1,30),DA)
779.4,.02	OWNER	0;2 FREE TEXT
	INPUT TRANSFORM:	K:$L(X)>40!($L(X)<1) X
	LAST EDITED:	JUN 01, 2004
	HELP-PROMPT:	Answer must be 1-40 characters in length.
	DESCRIPTION:	This is the application or package responsible for creating this subscription.
	CROSS-REFERENCE:	779.4^AH^MUMPS 1)= Q 2)= D KILLAH^HLOASUB1(DA) The AH x-ref is maintained by the owner of the subscription. It represents a private index by which the application my perform a lookup to find a subscription. The use of a private index is optional, the alternative being that the owning application may store the ien of the subscription entry with the application's data. The format of the index is as follows:

```
                              ^HLS(779.4,"AH",<owner>,<list of lookup
                              values....>,<DA>)=""


779.4,.03     DESCRIPTION            1;1 FREE TEXT

              INPUT TRANSFORM:  K:$L(X)>75!($L(X)<1) X
              LAST EDITED:      DEC 07, 1999
              HELP-PROMPT:      Answer must be 1-75 characters in length

779.4,20      RECIPIENTS             2;0 Multiple #779.41


              INDEXED BY:       DATE/TIME ADDED & DATE/TIME TERMINATED (AC),
                                RECEIVING APPLICATON & LOGICAL LINK & RECEIVING
                                FACILITY COMPONENT 1 & RECEIVING FACILITY
                                COMPONENT 2 & RECEIVING FACILITY COMPONENT 3
                                (AD)

779.41,.01      RECEIVING APPLICATON   0;1 FREE TEXT (Required)
                                 (Multiply asked)

                INPUT TRANSFORM:  K:$L(X)>60!($L(X)<1) X
                LAST EDITED:      AUG 17, 2004
                HELP-PROMPT:      Enter the name of the application that will
                                  receive the message, up to 60 characters.
                CROSS-REFERENCE:  779.41^B
                                  1)= S ^HLS(779.4,DA(1),2,"B",$E(X,1,30),DA)="
                                  "

                                  2)= K ^HLS(779.4,DA(1),2,"B",$E(X,1,30),DA)

                RECORD INDEXES:   AD (#444)

779.41,.02      LOGICAL LINK           0;2 POINTER TO HL LOGICAL LINK FILE (#87
                                  0) (Required)

                LAST EDITED:      MAY 27, 2004
                HELP-PROMPT:      Over which communication link should the
                                  messages be sent?
                RECORD INDEXES:   AD (#444)

779.41,.03      RECEIVING FACILITY COMPONENT 1 0;3 FREE TEXT

                INPUT TRANSFORM:  K:$L(X)>50!($L(X)<1) X
                LAST EDITED:      MAY 27, 2004
                HELP-PROMPT:      Answer must be 1-50 characters in length
                DESCRIPTION:      This is the value that should be placed in
                                  component 1 of the receiving facility field
                                  of the message header.

                RECORD INDEXES:   AD (#444)

779.41,.04      RECEIVING FACILITY COMPONENT 2 0;4 FREE TEXT

                INPUT TRANSFORM:  K:$L(X)>50!($L(X)<1) X
                LAST EDITED:      MAY 27, 2004
                HELP-PROMPT:      Answer must be 1-50 characters in length
                DESCRIPTION:      This is the value that should be placed in
                                  component 2 of the receiving facility field
                                  of the message header.
```

```
                RECORD INDEXES:   AD (#444)

779.41,.05      RECEIVING FACILITY COMPONENT 3 0;5 FREE TEXT

                INPUT TRANSFORM:  K:$L(X)>10!($L(X)<1) X
                LAST EDITED:      NOV 16, 2004
                HELP-PROMPT:      Answer must be 1-10 characters in length
                DESCRIPTION:      This is the value that should be placed in
                                  component 3 of the receiving facility field
                                  of the message header.

                RECORD INDEXES:   AD (#444)

779.41,1.01     DATE/TIME ADDED        1;1 DATE (Required)

                INPUT TRANSFORM:  S %DT="ESTXR" D ^%DT S X=Y K:X<1 X
                LAST EDITED:      MAY 26, 2004
                HELP-PROMPT:      Enter the date and time that this recipient
                                  was added to the subscription list.
                RECORD INDEXES:   AC (#443)

779.41,1.02     DATE/TIME TERMINATED   1;2 DATE

                INPUT TRANSFORM:  S %DT="ESTXR" D ^%DT S X=Y K:X<1 X
                LAST EDITED:      MAY 26, 2004
                HELP-PROMPT:      Enter the date and time that this recipient
                                  was dropped from the subscription list
                RECORD INDEXES:   AC (#443)



     FILES POINTED TO                       FIELDS

HL LOGICAL LINK (#870)            RECIPIENTS:LOGICAL LINK (#.02)


Subfile #779.41

  Record Indexes:

  AC (#443)    RECORD    MUMPS    IR    SORTING ONLY
      Short Descr:  Used to find recipients who have not been terminated.
        Set Logic:  S ^HLD(779.4,DA(1),2,"AC",DA)=""
         Set Cond:  S X=$S($G(X2(2)):0,1:1)
       Kill Logic:  K ^HLD(779.4,DA(1),2,"AC",DA)
        Kill Cond:  S X=1
             X(1):  DATE/TIME ADDED  (779.41,1.01)  (Subscr 1)  (forwards)
             X(2):  DATE/TIME TERMINATED  (779.41,1.02)  (forwards)

  AD (#444)    RECORD    MUMPS    IR    SORTING ONLY
      Short Descr:  Used to determine if the recipient is already on the
                    subscription list.
        Set Logic:  S ^HLD(779.4,DA(1),2,"AD",X2(1),X2(2),X2(3)_X2(4)_X2(5),DA)
                    =""
       Kill Logic:  K ^HLD(779.4,DA(1),2,"AD",X1(1),X1(2),X1(3)_X1(4)_X1(5),DA)
             X(1):  RECEIVING APPLICATON  (779.41,.01)  (Subscr 1)  (forwards)
             X(2):  LOGICAL LINK  (779.41,.02)  (Subscr 2)  (forwards)
             X(3):  RECEIVING FACILITY COMPONENT 1  (779.41,.03)  (forwards)
             X(4):  RECEIVING FACILITY COMPONENT 2  (779.41,.04)  (forwards)
             X(5):  RECEIVING FACILITY COMPONENT 3  (779.41,.05)  (forwards)
```

```
INPUT TEMPLATE(S):

PRINT TEMPLATE(S):

SORT TEMPLATE(S):

FORM(S)/BLOCK(S):
```

HL LOGICAL LINK File (#870)

This file is shared by HL7 1.6 and HLO. Two new fields have been added for HLO and they are bolded in the listing below. These fields are DNS DOMAIN field (# .08) and TCP/IP PORT (OPTIMIZED) field (# 400.08).

STANDARD DATA DICTIONARY #870 -- HL LOGICAL LINK FILE

```
STORED IN ^HLCS(870,

This file serves two purposes. It is a fileman-compatible transmission log. The
Low Layer Protocols write and read directly from this file.  (See routines
HLCSDR1 and HLCSDR2)

This file stores parameters that govern the behaviour of the Low Layer
Protocols. Fields like: READ TIMEOUT, ACK TIMEOUT, LLP START BLOCK, and LLP END
BLOCK, are fields that govern how long the finite state machine waits for data
to come down the line (READ TIMEOUT), how long it waits for a lower level
acknowledgement (ACK TIMEOUT), and which control characters define the
beginning and end of a message (LLP START BLOCK and LLP END BLOCK).

This file also stores information that drives the SYSTEMS LINK MONITOR display
option. Fields like, IN QUEUE FRONT POINTER, IN QUEUE BACK POINTER are used to
manage the data flow in the queues but they are also displayed on the SYSTEMS
LINK MONITOR under the alias's MESSAGES PROCESSED and MESSAGES RECEIVED. Fields
like STATE and DEVICE TYPE are also used to drive the SYSTEMS LINK MONITOR.
These fields are updated by the lower layer protocols in order to give
real-time feedback as to what is ocurring on a link. For example, when a
message is received (see HLCSDR1) the state transitions from "IDLE" to
"READING".


POINTED TO BY: LOGICAL LINK field (#2005) of the OUTPATIENT SITE File (#59)
               LOGICAL LINK field (#1) of the CPRS ORDERING INSTITUTION
                   sub-field (#59.08) of the OUTPATIENT SITE File (#59)
               LOGICAL LINK field (#770.7) of the PROTOCOL File (#101)
               LOGICAL LINK field (#11) of the HL7 MESSAGE TEXT File (#772)
               LOGICAL LINK field (#7) of the HL7 MESSAGE ADMINISTRATION File
                   (#773)
               LOGICAL LINK - IN QUEUE field (#17) of the HL7 MESSAGE
                   ADMINISTRATION File (#773)
               LOGICAL LINK field (#3) of the *DESTINATION sub-field (#774.01)
                   of the HL7 SUBSCRIPTION REGISTRY File (#774)
               LOGICAL LINK field (#.02) of the RECIPIENTS sub-field (#774.02)
                   of the HL7 SUBSCRIPTION REGISTRY File (#774)
               HLO STANDARD LISTENER field (#.1) of the HLO SYSTEM PARAMETERS
                   File (#779.1)
               APPLICATION SPECIFIC LISTENER field (#.09) of the HLO
                   APPLICATION REGISTRY File (#779.2)
               LOGICAL LINK field (#.02) of the RECIPIENTS sub-field (#779.41)
                   of the HLO SUBSCRIPTION REGISTRY File (#779.4)
               LOGICAL LINK field (#.01) of the LOGICAL LINK sub-field
                   (#869.311) of the LINK MONITOR VIEWS sub-field (#869.31) of
                   the HL COMMUNICATION SERVER PARAMETERS File (#869.3)


CROSS REFERENCED BY: STATUS(AISTAT), LLP TYPE(ALLP), NODE(B), INSTITUTION(C),
                     MAILMAN DOMAIN(D), TCP/IP SERVICE TYPE(E)

INDEXED BY:    INSTITUTION & NODE (AC), NODE & LLP TYPE & MAILMAN DOMAIN &
               TCP/IP SERVICE TYPE (AD), NODE & LLP TYPE & DNS DOMAIN & TCP/IP
```

```
              SERVICE TYPE (AD1), NODE & LLP TYPE & HLLP DEVICE (AD2), DNS
              DOMAIN (DNS), DNS DOMAIN & NODE (DNS2), IEN772 InQ-Non-TCP
              (IEN772IN), IEN772 OutQ-Non-TCP (IEN772OUT), TCP/IP ADDRESS (IP)


870,.01       NODE                    0;1 FREE TEXT (Required)

              INPUT TRANSFORM:  K:$L(X)>10!($L(X)<3)!'(X'?1P.E) X
              LAST EDITED:      MAR 08, 2005
              HELP-PROMPT:      Enter the name of the logical link (3-10
                                characters)
              DESCRIPTION:      This is the name of the logical link that data
                                will be communicated on.  It is this name that
                                will be displayed in the "NODE" column of the
                                SYSTEMS LINK MONITOR display option.  It is
                                suggested that this name be the same name as
                                the commercial application e.g.
                                KURZWEIL1,KURZWEIL2,COPATH, or SUNQUEST1.

              CROSS-REFERENCE:  870^B
                                1)= S ^HLCS(870,"B",$E(X,1,30),DA)=""
                                2)= K ^HLCS(870,"B",$E(X,1,30),DA)

              RECORD INDEXES:   AC (#433), AD (#434), AD1 (#435), AD2 (#436),
                                DNS2 (#438)

870,.02       INSTITUTION             0;2 POINTER TO INSTITUTION FILE (#4)

              INPUT TRANSFORM:  K:$O(^HLCS(870,"C",X,0)) X
              LAST EDITED:      MAY 28, 2004
              HELP-PROMPT:      Select an institution that has not been
                                associated with another link.
              DESCRIPTION:      It may be necessary for an application to
                                determine the appropriate link to use when the
                                only information it has is the institution.
                                This field is used to associate a link with
                                that institution. There is a 1:1 correspondence
                                between institution and logical link. An
                                institution cannot be associated with more than
                                one logical link.

              NOTES:            XXXX--CAN'T BE ALTERED EXCEPT BY PROGRAMMER

              CROSS-REFERENCE:  870^C
                                1)= S ^HLCS(870,"C",$E(X,1,30),DA)=""
                                2)= K ^HLCS(870,"C",$E(X,1,30),DA)

              RECORD INDEXES:   AC (#433)

870,.03       MAILMAN DOMAIN          0;7 POINTER TO DOMAIN FILE (#4.2)

              INPUT TRANSFORM:  K:$O(^HLCS(870,"D",X,0)) X
              LAST EDITED:      MAR 08, 2005
              HELP-PROMPT:      Enter the Mailman domain corresponding to this
                                link.
              DESCRIPTION:      Either this field or the OTHER DOMAIN field
                                should have a value.  The domain is used to
                                formulate the RECEIVING FACILITY field of the
                                message header.

              NOTES:            XXXX--CAN'T BE ALTERED EXCEPT BY PROGRAMMER

              CROSS-REFERENCE:  870^D
```

```
                          1)= S ^HLCS(870,"D",$E(X,1,30),DA)=""
                          2)= K ^HLCS(870,"D",$E(X,1,30),DA)

               RECORD INDEXES:   AD (#434)

870,.08        DNS DOMAIN             0;8 FREE TEXT

               INPUT TRANSFORM:  K:$L(X)>70!($L(X)<4) X S HLIP=$$ADDRESS^XLFNSLK
                                 (X) K:('HLIP)&($P($G(^HLCS(869.3,1,0)),"^",3)="
                                 P")&($P($$SITE^VASITE,"^",3)) X I HLIP S HLIP=$
                                 $IP^HLMA3(DA,HLIP)
               LAST EDITED:      MAR 09, 2005
               HELP-PROMPT:      Enter the full domain name as registered with
                                 DNS.
               DESCRIPTION:
                                 The domain name as registered with DNS.

               NOTES:            XXXX--CAN'T BE ALTERED EXCEPT BY PROGRAMMER

               FIELD INDEX:      DNS (#437)    REGULAR    IR    LOOKUP & SORTING
                   Short Descr:  This is a regular index of new-style
                                 cross-reference on the DNS Domain field
                     Set Logic:  S ^HLCS(870,"DNS",$E(X,1,70),DA)=""
                    Kill Logic:  K ^HLCS(870,"DNS",$E(X,1,70),DA)
                    Whole Kill:  K ^HLCS(870,"DNS")
                          X(1):  DNS DOMAIN  (870,.08)  (Subscr 1)  (Len 70)
                                 (forwards)

               RECORD INDEXES:   AD1 (#435), DNS2 (#438)

870,2          LLP TYPE               0;3 POINTER TO HL LOWER LEVEL PROTOCOL TYP
                                      E FILE (#869.1) (Required)

               LAST EDITED:      MAR 08, 2005
               HELP-PROMPT:      Enter the  LLP used for this logical link.
               DESCRIPTION:      Enter the type of Lower Level Protocol for this
                                 logical link. Selection of TCP implies that the
                                 MLLP protocol will be used. Each of the
                                 supported LLP's are described in detail in
                                 Appendix C of the HL7 Implementation Guide.

               CROSS-REFERENCE:  870^ALLP
                                 1)= S ^HLCS(870,"ALLP",$E(X,1,30),DA)=""
                                 2)= K ^HLCS(870,"ALLP",$E(X,1,30),DA)
                                 This cross-reference is used to link the HL
                                 Lower Layer Parameter file (#869.2) with the HL
                                 Logical Link file. Using this x-ref you can
                                 locate the parameter associated with this link.


               RECORD INDEXES:   AD (#434), AD1 (#435), AD2 (#436)

870,3          DEVICE TYPE            0;4 SET

                                 'PC' FOR Persistent Client;
                                 'NC' FOR Non-Persistent Client;
                                 'SS' FOR Single-threaded Server;
                                 'MS' FOR Multi-threaded Server;
                                 'SH' FOR Serial HLLP;
                                 'SX' FOR Serial X3.28;
                                 'MM' FOR MailMan;
               LAST EDITED:      JUL 10, 2003
```

```
             HELP-PROMPT:      Not editable from any user option. See field
                               description.
             DESCRIPTION:      This field is not editable from any user
                               option. It is used purely for display purposes
                               in the DEVICE TYPE column of the SYSTEMS LINK
                               MONITOR display option.


870,4        STATE                  0;5 FREE TEXT

             INPUT TRANSFORM:  K:$L(X)>10!($L(X)<1) X
             LAST EDITED:      OCT 06, 1994
             HELP-PROMPT:      Not editable from any user option. See field
                               description.
             DESCRIPTION:      This field is not editable from any user
                               option. It is used purely for display purposes
                               in the STATE column of the SYSTEMS LINK MONITOR
                               display option.


870,4.5      AUTOSTART              0;6 SET

                               '0' FOR Disabled;
                               '1' FOR Enabled;
             LAST EDITED:      NOV 20, 1996
             DESCRIPTION:      This field permits the link to be restarted by
                               Taskman after a system reboot. If disabled, the
                               link will not be restarted.


870,5        IN QUEUE FRONT POINTER IN QUEUE FRONT POINTER;1 NUMBER

             INPUT TRANSFORM:  K:+X'=X!(X>99999999999999)!(X<0)!(X?.E1"."1N.N)
                                X
             LAST EDITED:      DEC 21, 2004
             HELP-PROMPT:      This field is not editable from any user
                               option. See Field Description.
             DESCRIPTION:      The queue structure relies on two queues,
                               INcoming and OUTgoing.  This pointer is used to
                               point to the first message in the FIFO queue.


870,6        IN QUEUE BACK POINTER  IN QUEUE BACK POINTER;1 NUMBER

             INPUT TRANSFORM:  K:+X'=X!(X>99999999999999)!(X<0)!(X?.E1"."1N.N)
                                X
             LAST EDITED:      DEC 21, 2004
             HELP-PROMPT:      Type a Number between 0 and 99999999999999, 0
                               Decimal Digits
             DESCRIPTION:      The queue structure relies on two
                               queues-INcoming and OUTgoing.  This pointer
                               points to the last message in the FIFO queue.


870,7        OUT QUEUE FRONT POINTER OUT QUEUE FRONT POINTER;1 NUMBER

             INPUT TRANSFORM:  K:+X'=X!(X>99999999999999)!(X<0)!(X?.E1"."1N.N)
                                X
             LAST EDITED:      DEC 21, 2004
             HELP-PROMPT:      This field is not editable from any user
                               option. See Field Description.
             DESCRIPTION:      The queue structure relies on two
                               queues-INcoming and OUTgoing.  This pointer
```

```
                              points to the first message in the FIFO queue.

870,8          OUT QUEUE BACK POINTER OUT QUEUE BACK POINTER;1 NUMBER

               INPUT TRANSFORM:  K:+X'=X!(X>99999999999999)!(X<0)!(X?.E1"."1N.N)
                                  X
               LAST EDITED:      DEC 21, 2004
               HELP-PROMPT:      This field is not editable from any user
                                 option. See field description.
               DESCRIPTION:      The queue structure relies on two queues,
                                 INcoming and OUTgoing.  This pointer points to
                                 the last message in the FIFO queue.


870,9          TIME STARTED           0;10 DATE

               INPUT TRANSFORM:  S %DT="EST" D ^%DT S X=Y K:Y<1 X
               LAST EDITED:      OCT 06, 1994
               HELP-PROMPT:      This field is not editable from any user
                                 option. See Field Description.
               DESCRIPTION:      This field contains the Date/Time the Lower
                                 Layer Protocol was last launched for the
                                 Logical Link. Only applicable to Lower Layer
                                 Protocols, not Mailman.


870,10         TIME STOPPED           0;11 DATE

               INPUT TRANSFORM:  S %DT="EST" D ^%DT S X=Y K:Y<1 X
               LAST EDITED:      OCT 06, 1994
               HELP-PROMPT:      This field is not editable from any user
                                 option. See Field Description.
               DESCRIPTION:      This field contains the Date/Time the Lower
                                 Layer Protocol was last shutdown for the
                                 Logical Link. Only applicable to Lower Layer
                                 Protocols, not Mailman.


870,11         TASK NUMBER            0;12 NUMBER

               INPUT TRANSFORM:  K:+X'=X!(X>999999999999999)!(X<1)!(X?.E1"."1N.N
                                 ) X
               LAST EDITED:      OCT 06, 1994
               HELP-PROMPT:      This field is not editable from any user
                                 option. See Field Description.
               DESCRIPTION:      This is the task number assigned by TaskMan
                                 when the Lower Layer Protocol is launched in
                                 the background.


870,14         SHUTDOWN LLP ?         0;15 SET

                                 '0' FOR NO;
                                 '1' FOR YES;
               LAST EDITED:      FEB 22, 1995
               HELP-PROMPT:      Setting this field to 'Y' through the STOP
                                 HYBRID LLP option, shuts down the lower layer
                                 protocol and closes the device.
               DESCRIPTION:      This field is editable through the STOP HYBRID
                                 LLP option. When the Lower Layer Protocol is
                                 running, and this field is set to 'Y' the Lower
                                 Layer Protocol will shutdown and close the
```

```
                              device.

870,18        GROSS COMMUNICATIONS ERROR 0;19 POINTER TO HL7 ERROR MESSAGE FILE
                                (#771.7)

              LAST EDITED:      DEC 13, 1994
              HELP-PROMPT:      This field contains the most recent
                                communications error which has occured on a
                                particular link.
              DESCRIPTION:      This field contains the most recent gross
                                communications error. It serves a two purposes.
                                It is a flag for the SYSTEMS LINK MONITOR.  In
                                other words, if this field is defined it means
                                a communications error has occured on the link.
                                This is indicated by the NODE field flashing on
                                the SYSTEMS LINK MONITOR. The error can be
                                viewed by using the SHOW COMMUNICATION ERROR
                                option.  The error can be cleared by using the
                                CLEAR COMMUNICATION ERROR option.  If set, an
                                error occurred while transmitting (timeout for
                                example).  This field can be viewed using the
                                'Show Gross LLP Error' and cleared using the
                                'Clear Gross LLP Error' options.


870,19        IN QUEUE              1;0 Multiple #870.019

              LAST EDITED:      OCT 12, 2004
              DESCRIPTION:      This multiple contains the queue for incoming
                                messages.


870.019,.01     MESSAGE NUMBER         0;1 NUMBER

                INPUT TRANSFORM:K:+X'=X!(X>999999999999999)!(X<0)!(X?.E1"."1N.N
                                ) X
                LAST EDITED:    NOV 02, 1994
                HELP-PROMPT:    Not editable from any user option. See field
                                description.
                DESCRIPTION:    This is the number assigned to the message when
                                it is enqued into the FIFO queue.

                CROSS-REFERENCE:870.019^B
                                1)= S ^HLCS(870,DA(1),1,"B",$E(X,1,30),DA)=""
                                2)= K ^HLCS(870,DA(1),1,"B",$E(X,1,30),DA)


870.019,1       STATUS              0;2 SET

                                'P' FOR PENDING;
                                'D' FOR DONE PROCESSING;
                                'S' FOR STUB RECORD;
                LAST EDITED:    MAY 08, 2000
                HELP-PROMPT:    This is the status of an individual message in
                                the queue. These statuses control the flow and
                                processing of the message.
                DESCRIPTION:    Status of a particular message. Can be Stub,
                                Pending, or Done.  These statuses control the
                                processing of a message. For example, a stub
                                record is created when a message is enqueued,
                                and cannot be processed until its status is
                                changed from 'stub' to 'pending'.  Changing
```

```
                              this status is the last thing the Low Layer
                              Protocol does after reading in a message.
                              Finally, once the message has been processed,
                              its status is changed to 'done'.

                    CROSS-REFERENCE:870^AISTAT
                              1)= S ^HLCS(870,"AISTAT",$E(X,1,30),DA(1),DA)="
                              "

                              2)= K ^HLCS(870,"AISTAT",$E(X,1,30),DA(1),DA)
                              Used to quickly identify links with messages in
                              any status, thus improving performance.


870.019,2           ERROR               0;3 SET

                              'A' FOR LLP ACKNOWLEDGED NO ERROR;
                              'C' FOR LLP CHAR COUNT WRONG;
                              'X' FOR LLP XOR WRONG;
                              'B' FOR DATA TOO LONG;
                              'G' FOR OTHER LLP ERROR;
                    LAST EDITED:    OCT 27, 1994
                    HELP-PROMPT:    This field is not editable from any user
                              option. See Field Description.
                    DESCRIPTION:    If this field contains a "C","X","B", or a "G"
                              it means an error has ocurred while reading in
                              a message. This field is not editable from any
                              user option.


870.019,3           MESSAGE TEXT        1;0   WORD-PROCESSING #870.193 (NOWRAP)

                    DESCRIPTION:
                              This field contains the text of the message.


870.019,4           tvv                 0;4 FREE TEXT

                    INPUT TRANSFORM:K:$L(X)>3!($L(X)<3) X
                    LAST EDITED:    JAN 12, 1995
                    HELP-PROMPT:    Answer must be 3 characters in length.

870.019,5           ccccc               0;5 FREE TEXT

                    INPUT TRANSFORM:K:$L(X)>5!($L(X)<1) X
                    LAST EDITED:    JAN 13, 1995
                    HELP-PROMPT:    Answer must be 1-5 characters in length.

870.019,6           xxx                 0;6 FREE TEXT

                    INPUT TRANSFORM:K:$L(X)>3!($L(X)<1) X
                    LAST EDITED:    JAN 13, 1995
                    HELP-PROMPT:    Answer must be 1-3 characters in length.

870.019,7           dhcp-longtitudinal checksum 0;7 FREE TEXT

                    INPUT TRANSFORM:K:$L(X)>5!($L(X)<1) X
                    LAST EDITED:    JAN 13, 1995
                    HELP-PROMPT:    Answer must be 1-5 characters in length.

870.019,8           dhcp-xor checksum   0;8 FREE TEXT
```

```
                INPUT TRANSFORM:K:$L(X)>3!($L(X)<1) X
                LAST EDITED:    JAN 13, 1995
                HELP-PROMPT:    Answer must be 1-3 characters in length.

870.019,9       IEN772 InQ-Non-TCP   0;9 FREE TEXT

                INPUT TRANSFORM:K:$L(X)>30!($L(X)<1)!'(X?.N) X
                LAST EDITED:    DEC 21, 2004
                HELP-PROMPT:    Answer must be 1-30 numeric digits in length.
                DESCRIPTION:    The ien of file #772, where this sub-entry is
                                copied to.

                FIELD INDEX:    IEN772IN (#439)    REGULAR    IR
                                LOOKUP & SORTING    WHOLE FILE (#870)
                  Short Descr:  IEN772 whole file cross reference
                    Set Logic:  S ^HLCS(870,"IEN772IN",X,DA(1),DA)=""
                   Kill Logic:  K ^HLCS(870,"IEN772IN",X,DA(1),DA)
                   Whole Kill:  K ^HLCS(870,"IEN772IN")
                         X(1):  IEN772 InQ-Non-TCP  (870.019,9)  (Subscr 1)
                                (forwards)



870,20          OUT QUEUE             2;0 Multiple #870.01
                                    (Add New Entry without Asking)

                LAST EDITED:        OCT 12, 2004
                DESCRIPTION:        This multiple contains the queue for outgoing
                                    messages.


870.01,.01      MESSAGE NUMBER         0;1 NUMBER

                INPUT TRANSFORM:K:+X'=X!(X>999999999999999)!(X<0)!(X?.E1"."1N.N
                                ) X
                LAST EDITED:    NOV 02, 1994
                HELP-PROMPT:    Not editable from any user option. See Field
                                Description.
                DESCRIPTION:    This is the number assigned to the message when
                                it is enque'd into the FIFO queue.

                CROSS-REFERENCE:870.01^B
                                1)= S ^HLCS(870,DA(1),2,"B",$E(X,1,30),DA)=""
                                2)= K ^HLCS(870,DA(1),2,"B",$E(X,1,30),DA)


870.01,1        STATUS               0;2 SET

                                'P' FOR PENDING;
                                'D' FOR DONE PROCESSING;
                                'S' FOR STUB RECORD;
                LAST EDITED:    MAY 08, 2000
                HELP-PROMPT:    This is the status of an individual message in
                                the queue. These statuses control the flow and
                                processing of the message.
                DESCRIPTION:    Status of a particular message. Can be Stub,
                                Pending, or Done. These statuses control the
                                processing of a message. For example, a stub
                                record is created when a message is enque'd, it
                                cannot be processed until it's status is
                                changed from a 'stub' to 'pending', this is the
                                last thing HLCS1 does when dequeueing a message
```

```
                              from file #772 and enqueueing it into file
                              #870.


870.01,2        ERROR                 0;3 SET

                              'A' FOR LLP ACKNOWLEDGED NO ERROR;
                              'C' FOR LLP CHAR COUNT WRONG;
                              'X' FOR LLP XOR WRONG;
                              'B' FOR DATA TOO LONG;
                              'G' FOR OTHER LLP ERROR;
                LAST EDITED:  NOV 17, 1994
                HELP-PROMPT:  This field is not editable from any user
                              option. See field description.
                DESCRIPTION:  If this field contains a "C","X","B", or a "G"
                              it means an error has occured while attempting
                              to send a message to another application.  This
                              field is not editable from any user option.


870.01,3        MESSAGE TEXT          1;0   WORD-PROCESSING #870.13 (NOWRAP)

                DESCRIPTION:
                              This field contains the text of the message.


870.01,4        dhcp-longtitudinal checksum 0;4 FREE TEXT

                INPUT TRANSFORM:K:$L(X)>5!($L(X)<1) X
                LAST EDITED:  JAN 13, 1995
                HELP-PROMPT:  Answer must be 1-5 characters in length.

870.01,5        dhcp-xor checksum   0;5 FREE TEXT

                INPUT TRANSFORM:K:$L(X)>3!($L(X)<1) X
                LAST EDITED:  JAN 13, 1995
                HELP-PROMPT:  Answer must be 1-3 characters in length.

870.01,6        IEN772 OutQ-Non-TCP  0;6 FREE TEXT

                INPUT TRANSFORM:K:$L(X)>30!($L(X)<1)!'(X?.N) X
                LAST EDITED:  DEC 21, 2004
                HELP-PROMPT:  Answer must be 1-30 numeric digits in length.
                DESCRIPTION:  The ien of file #772, where this sub-entry is
                              copied from.

                FIELD INDEX:  IEN772OUT (#440)    REGULAR    IR
                              LOOKUP & SORTING    WHOLE FILE (#870)
                  Short Descr:  IEN772 whole file cross reference
                    Set Logic:  S ^HLCS(870,"IEN772OUT",X,DA(1),DA)=""
                   Kill Logic:  K ^HLCS(870,"IEN772OUT",X,DA(1),DA)
                   Whole Kill:  K ^HLCS(870,"IEN772OUT")
                         X(1):  IEN772 OutQ-Non-TCP  (870.01,6)  (Subscr 1)
                              (forwards)



870,21          QUEUE SIZE            0;21 NUMBER

                INPUT TRANSFORM:  K:+X'=X!(X>100000)!(X<2)!(X?.E1"."1N.N) X
                LAST EDITED:      DEC 21, 2004
                HELP-PROMPT:      Type a Number between 2 and 100000, 0 Decimal
```

```
                        Digits
              DESCRIPTION:      This is the steady-state size of the queue. The
                                queue may dynamically grow beyond this size,
                                under certain conditions. See Users Manual, for
                                more information.


870,100.01    MAIL GROUP              100;1 POINTER TO MAIL GROUP FILE (#3.8)

              LAST EDITED:      JUL 07, 1999
              HELP-PROMPT:      Enter the mail group that messages should be
                                sent to.
              DESCRIPTION:      If you are building a logical link that will
                                use Mailman as a transport, you must define a
                                mail group that contains the remote member,

                                S.HL V16 SERVER@your target domain

                                The HL7 package will place outbound messages in
                                file 870's outque. The link must be running for
                                messages to be handed off to Mailman.

                                Inbound messages that are received by the
                                server option are placed directly in file 772.


870,200.01    HLLP DEVICE             200;1 POINTER TO DEVICE FILE (#3.5)

              LAST EDITED:      MAR 08, 2005
              HELP-PROMPT:      Enter the device to be used for the HLLP
                                protocol.
              DESCRIPTION:      This is an entry in the Device file (#3.5). It
                                is opened when this logical link is started up
                                and remains open until the link is shut down.
                                Normally, Vista will initiate and the
                                connection with this serial device.

              RECORD INDEXES:   AD2 (#436)

870,200.02    RE-TRANSMISSION ATTEMPTS 200;2 NUMBER

              INPUT TRANSFORM:  K:+X'=X!(X>999)!(X<0)!(X?.E1"."1N.N) X
              LAST EDITED:      JUL 07, 1999
              HELP-PROMPT:      Type a Number between 0 and 999, 0 Decimal
                                Digits
              DESCRIPTION:      Enter the number of times to re-try sending a
                                message.  The default is 5 tries if this field
                                is left blank. If a single message exceeds this
                                value an Alert is sent to the HL7 mail group
                                and the link is shutdown.


870,200.021   EXCEED RE-TRANSMIT ACTION 200;10 SET

                                'I' FOR ignore;
                                'R' FOR restart;
                                'S' FOR shutdown;
              LAST EDITED:      AUG 20, 1999
              DESCRIPTION:      This field determines what to do when a message
                                exceeds the number of
                                 retry attempts for this Logical Link.  Actions
                                are:
                                  Ignore   = send alert once and keep trying to
```

```
                              resend
                                Restart  = send alert once and shutdown link
                              then start link
                                Shutdown = send alert once and shutdown link


870,200.03    BLOCK SIZE             200;3 NUMBER

              INPUT TRANSFORM:  K:+X'=X!(X>512)!(X<9)!(X?.E1"."1N.N) X
              LAST EDITED:      JUL 07, 1999
              HELP-PROMPT:      Type a Number between 9 and 512, 0 Decimal
                                Digits245 is the recommended default.

870,200.04    READ TIMEOUT           200;4 NUMBER

              INPUT TRANSFORM:  K:+X'=X!(X>600)!(X<1)!(X?.E1"."1N.N) X
              LAST EDITED:      JUL 08, 1999
              HELP-PROMPT:      Type a Number between 1 and 600, 0 Decimal
                                Digits
              DESCRIPTION:      Enter the number of seconds the Lower Layer
                                Protocol remains in a read state for data to
                                come in on the link. The default is 10 seconds
                                if this field is left blank.


870,200.05    ACK TIMEOUT            200;5 NUMBER

              INPUT TRANSFORM:  K:+X'=X!(X>600)!(X<0)!(X?.E1"."1N.N) X
              LAST EDITED:      APR 18, 2000
              HELP-PROMPT:      Type a Number between 0 and 600, 0 Decimal
                                Digits
              DESCRIPTION:      The number of seconds the Lower Layer Protocol
                                waits for an acknowledgement from the receiving
                                application. The default is 60.  If this field
                                is less than the READ TIMEOUT field, the READ
                                TIMEOUT value will be used.


870,200.06    LLP START BLOCK        200;6 NUMBER

              INPUT TRANSFORM:  K:+X'=X!(X>300)!(X<1)!(X?.E1"."1N.N) X
              LAST EDITED:      JUL 09, 1999
              HELP-PROMPT:      Type a Number between 1 and 300, 0 Decimal
                                Digits
              DESCRIPTION:      Enter the numeric value of the control
                                character used by the HLLP communications
                                protocol as a START BLOCK CHARACTER. While this
                                character is negotiable, the HL7 implementation
                                guide recommends the use of the 'VT' character.
                                If this field is left blank, the default value
                                of 11 will be used.


870,200.07    LLP END BLOCK          200;7 NUMBER

              INPUT TRANSFORM:  K:+X'=X!(X>300)!(X<1)!(X?.E1"."1N.N) X
              LAST EDITED:      JUL 09, 1999
              HELP-PROMPT:      Type a Number between 1 and 300, 0 Decimal
                                Digits
              DESCRIPTION:      Enter the numeric value of the control
                                character defined in the HLLP specification as
                                the END BLOCK CHARACTER. The recommended value
                                is 28. If this field is left blank, the default
```

```
                          value will be 28 for the 'FS' character.

870,200.08    PROTOCOL ID VERSION    200;8 SET

                          '21' FOR 2.1;
                          '22' FOR 2.2;
                          '23' FOR 2.3;
              LAST EDITED:       AUG 05, 1999
              HELP-PROMPT:       Enter the version of HLLP as specified by the
                                 lates HL7 Implementation Guide
              DESCRIPTION:       The latest specification for the HLLP
                                 communications protocol is identified by a
                                 version number, which may be different from the
                                 HL7 standard itself.

870,200.09    UNI-DIRECTIONAL WAIT   200;9 NUMBER

              INPUT TRANSFORM:  K:+X'=X!(X>10)!(X<0)!(X?.E1"."1N.N) X
              LAST EDITED:       JUL 09, 1999
              HELP-PROMPT:       Type a Number between 0 and 10, 0 Decimal
                                 Digits
              DESCRIPTION:       The number of seconds the Lower Layer Protocol
                                 waits after sending a messages.  This is only
                                 used if the message doesn't want an
                                 acknowledgment.

870,300.01    X3.28 DEVICE           300;1 POINTER TO DEVICE FILE (#3.5)

              LAST EDITED:       JUL 09, 1999
              HELP-PROMPT:       Enter the device from the Device File.
              DESCRIPTION:       X3.28 is a serial protocol like HLLP. It
                                 requires the remote system be configured as a
                                 device (bi-directional) in the Device and
                                 terminal type files. The Vista system will
                                 normally open and maintain the connection.  See
                                 the HL7 Implementation Guide for details of
                                 this protocol.

870,300.02    MAXIMUM MESSAGE SIZE   300;2 NUMBER

              INPUT TRANSFORM:  K:+X'=X!(X>9999999)!(X<1)!(X?.E1"."1N.N) X
              LAST EDITED:       JUL 09, 1999
              HELP-PROMPT:       Type a Number between 1 and 9999999, 0 Decimal
                                 Digits
              DESCRIPTION:       See the HL7 Implementation Guide for details of
                                 the x3.28 protcol.

870,300.03    MAXIMUM BLOCK SIZE     300;3 NUMBER

              INPUT TRANSFORM:  K:+X'=X!(X>245)!(X<9)!(X?.E1"."1N.N) X
              LAST EDITED:       JUL 09, 1999
              HELP-PROMPT:       Type a Number between 9 and 245, 0 Decimal
                                 Digits
              DESCRIPTION:       See the HL7 Implementation Guide for details of
                                 the x3.28 protcol.

870,300.04    TIMER A                300;4 NUMBER
```

```
                INPUT TRANSFORM:  K:+X'=X!(X>60)!(X<1)!(X?.E1"."1N.N) X
                LAST EDITED:      JUL 09, 1999
                HELP-PROMPT:      Type a Number between 1 and 60, 0 Decimal
                                  Digits
                DESCRIPTION:      Enter the time in seconds for the Response
                                  Timer. The default is 6 seconds if no time is
                                  entered.


870,300.05      TIMER B                300;5 NUMBER

                INPUT TRANSFORM:  K:+X'=X!(X>60)!(X<1)!(X?.E1"."1N.N) X
                LAST EDITED:      JUL 09, 1999
                HELP-PROMPT:      Type a Number between 1 and 60, 0 Decimal
                                  Digits
                DESCRIPTION:      Enter the time in seconds for the Receive
                                  Timer. The default is 3 seconds if no time is
                                  entered.


870,300.06      TIMER D                300;6 NUMBER

                INPUT TRANSFORM:  K:+X'=X!(X>60)!(X<1)!(X?.E1"."1N.N) X
                LAST EDITED:      JUL 09, 1999
                HELP-PROMPT:      Type a Number between 1 and 60, 0 Decimal
                                  Digits
                DESCRIPTION:      Enter the time for the Inter-Block timer. The
                                  default is 30 seconds if no time is specified.


870,300.07      TIMER E                300;7 NUMBER

                INPUT TRANSFORM:  K:+X'=X!(X>600)!(X<1)!(X?.E1"."1N.N) X
                LAST EDITED:      JUL 09, 1999
                HELP-PROMPT:      Type a Number between 1 and 600, 0 Decimal
                                  Digits
                DESCRIPTION:      Enter the time for the Line Check Timer. The
                                  default is 180 seconds is no time is specified.



870,400.01      TCP/IP ADDRESS         400;1 FREE TEXT

                INPUT TRANSFORM:  K:$L(X)>40!($L(X)<7) X
                LAST EDITED:      OCT 12, 2004
                HELP-PROMPT:      Answer must be 7-40 characters in length.
                DESCRIPTION:      Enter the numeric address of the remote site
                                  using the syntax, nn.nn.nn.nn

                FIELD INDEX:      IP (#441)    REGULAR    IR    LOOKUP & SORTING
                    Short Descr:  IP Address cross reference
                      Set Logic:  S ^HLCS(870,"IP",$E(X,1,40),DA)=""
                     Kill Logic:  K ^HLCS(870,"IP",$E(X,1,40),DA)
                     Whole Kill:  K ^HLCS(870,"IP")
                           X(1):  TCP/IP ADDRESS  (870,400.01)  (Subscr 1)
                                  (Len 40)  (forwards)


870,400.02      TCP/IP PORT            400;2 NUMBER

                INPUT TRANSFORM:  K:+X'=X!(X>65535)!(X<1)!(X?.E1"."1N.N) X
                LAST EDITED:      JUL 09, 1999
```

```
                HELP-PROMPT:      Type a Number between 1 and 65535, 0 Decimal
                                  Digits
                DESCRIPTION:      This is the port over which the HL7 service
                                  will create a socket for message exchange. In
                                  VA, this port will be 5000 between production
                                  systems.


870,400.03      TCP/IP SERVICE TYPE    400;3 SET

                                  'C' FOR CLIENT (SENDER);
                                  'S' FOR SINGLE LISTENER;
                                  'M' FOR MULTI LISTENER;
                LAST EDITED:      MAR 08, 2005
                HELP-PROMPT:      Does this link send a connection request or
                                  receive connection requests?
                DESCRIPTION:      This field determines if the Logical Link is
                                  the client (sender) or a listener (server) of a
                                  message.  Choose from:


                                   CLIENT (SENDER): Indicates that this Logical
                                  Link connects to a target system, with the
                                  current system acting as the sender.

                                   SINGLE LISTENER: Designates that the current
                                  system is a server (listener), using a single M
                                  process to do the listening.

                                   MULTI LISTENER: Designates that the current
                                  system is a server (listener), creating
                                  multiple background processes.

                CROSS-REFERENCE:  870^E
                                  1)= S ^HLCS(870,"E",$E(X,1,30),DA)=""
                                  2)= K ^HLCS(870,"E",$E(X,1,30),DA)

                RECORD INDEXES:   AD (#434), AD1 (#435)

870,400.04      PERSISTENT             400;4 SET

                                  'Y' FOR YES;
                                  'N' FOR NO;
                LAST EDITED:      JUL 09, 1999
                DESCRIPTION:      Enter 'YES' if this connection needs to remain
                                  open even if there are no messages to send. The
                                  connection will remain open until it is
                                  disconnected by either side via shutting-down
                                  the logical link.  A setting of 'YES' is
                                  appropriate for connecting to a COTS device
                                  with a high volume of messages.


870,400.05      RETENTION              400;5 NUMBER

                INPUT TRANSFORM:  K:+X'=X!(X>999999)!(X<0)!(X?.E1"."1N.N) X
                LAST EDITED:      JUL 09, 1999
                HELP-PROMPT:      Type a Number between 0 and 999999, 0 Decimal
                                  Digits
                DESCRIPTION:      Retention is the maximum time in seconds in
                                  which a non-persistent LLP will wait after the
                                  associated queue has been emptied.  If further
                                  messages arrive in the queue before the
```

retention time has expired, the LLP will continue to dequeue and send messages. Otherwise, the LLP will become inactive and will remain inactive until further messages are queued.

870,400.06 STARTUP NODE 400;6 POINTER TO TASKMAN SITE PARAMETERS FILE (#14.7)

LAST EDITED: JUL 09, 1999
HELP-PROMPT: Enter the Taskman node to start this LLP on.
DESCRIPTION: This field is ONLY for VMS sites running Dual Taskman in DCL. This field is used to specify what Taskman node you want to job the Lower Level Protocol. It should only be used if you have two Taskmans running and only want the LLP to run on a particular node. It will only work if you are running the dual Taskmans in DCL context on a VMS system.

870,400.07 SAY HELO 400;7 SET

'Y' FOR YES;
'N' FOR NO;
LAST EDITED: NOV 02, 2000
HELP-PROMPT: Send initial HELO for Cache/NT TCP links?
DESCRIPTION: This field is applicable only to Cache/NT sites with TCP links. If you are not a Cache/NT site, or this is not a TCP link, this field does not apply, AND will be ignored.

If this TCP link is for a VA site, answer YES. If this TCP link is for a COTS system, answer NO. This is the default.

TECHNICAL DESCR: When sending HL7 transactions to other VA sites over TCP links, Cache/NT sites must send an initial HELO to overcome buffering issues. However, the initial HELO is not part of the HL7 standard, and may cause a reject error when sent to a COTS system.

870,400.08 TCP/IP PORT (OPTIMIZED) 400;8 NUMBER

INPUT TRANSFORM: K:+X'=X!(X>65535)!(X<1)!(X?.E1"."1N.N) X
LAST EDITED: FEB 16, 2005
HELP-PROMPT: Enter the port to use for the new HL7 'optimized' server, a number between 1 and 65535. 5001 is the default.
DESCRIPTION: The new HL7 'optimized' server will operate concurrently with the old HL7 1.6 server. To enable that, the two servers are assigned different ports to listen on. The default port for the old HL7 server is 5000, whereas the new optimized HL7 server uses port 5001.

FILES POINTED TO FIELDS

```
DEVICE (#3.5)                     HLLP DEVICE (#200.01)
                                  X3.28 DEVICE (#300.01)

DOMAIN (#4.2)                     MAILMAN DOMAIN (#.03)

HL LOWER LEVEL PROTOCOL TYPE
                   (#869.1)       LLP TYPE (#2)

HL7 ERROR MESSAGE (#771.7)        GROSS COMMUNICATIONS ERROR (#18)

INSTITUTION (#4)                  INSTITUTION (#.02)

MAIL GROUP (#3.8)                 MAIL GROUP (#100.01)

TASKMAN SITE PARAMETERS (#14.7)   STARTUP NODE (#400.06)


File #870

  Record Indexes:

  AC (#433)    RECORD    MUMPS    IR    SORTING ONLY
      Short Descr:  Sorts entries by <station number>,<link name>,<ien>
        Set Logic:  S ^HLCS(870,"AC",$S($L($P($G(^DIC(4,+X2(1),99)),"^")):$P($G
                    (^DIC(4,+X2(1),99)),"^"),1:" "),X2(2),DA)=""
       Kill Logic:  K ^HLCS(870,"AC",$S($L($P($G(^DIC(4,+X1(1),99)),"^")):$P($G
                    (^DIC(4,+X1(1),99)),"^"),1:" "),X1(2),DA)
       Whole Kill:  K ^HLCS(870,"AC")
             X(1):  INSTITUTION  (870,.02)  (Subscr 1)  (forwards)
             X(2):  NODE  (870,.01)  (forwards)

  AD (#434)    RECORD    MUMPS    IR    ACTION
      Short Descr:  Used to find the link over which to send the application
                    acknowledgment.
        Set Logic:  D SET1^HLOTLNK(X(1),X(3))
         Set Cond:  S X=0 I X(4)="C",X(2),$P($G(^HLCS(869.1,X(2),0)),"^")="TCP"
                     S X=1
       Kill Logic:  D KILL1^HLOTLNK(X(1),X(3))
        Kill Cond:  S X=0 I X(4)="C",X(2),$P($G(^HLCS(869.1,X(2),0)),"^")="TCP"
                     S X=1
             X(1):  NODE  (870,.01)  (Subscr 1)  (Len 30)  (forwards)
             X(2):  LLP TYPE  (870,2)  (Subscr 2)  (forwards)
             X(3):  MAILMAN DOMAIN  (870,.03)  (Subscr 3)  (forwards)
             X(4):  TCP/IP SERVICE TYPE  (870,400.03)  (Subscr 4)  (forwards)

  AD1 (#435)    RECORD    MUMPS    IR    ACTION
      Short Descr:  Used to find the link overwhich to send the application
                    acknowledgment.
        Set Logic:  D SET2^HLOTLNK(X(1),X(3))
         Set Cond:  S X=0 I X(4)="C",X(2),$P($G(^HLCS(869.1,X(2),0)),"^")="TCP"
                     S X=1
       Kill Logic:  D KILL2^HLOTLNK(X(1),X(3))
        Kill Cond:  S X=0 I X(4)="C",X(2),$P($G(^HLCS(869.1,X(2),0)),"^")="TCP"
                     S X=1
             X(1):  NODE  (870,.01)  (Subscr 1)  (forwards)
             X(2):  LLP TYPE  (870,2)  (Subscr 2)  (forwards)
             X(3):  DNS DOMAIN  (870,.08)  (Subscr 3)  (forwards)
             X(4):  TCP/IP SERVICE TYPE  (870,400.03)  (Subscr 4)  (forwards)

  AD2 (#436)    RECORD    MUMPS    IR    ACTION
      Short Descr:  Used to find the link overwhich to send an application
                    acknowledgment.
        Set Logic:  D SET3^HLOTLNK(X(1),X(3))
```

```
         Set Cond:  S X=0 I X(2),$P($G(^HLCS(869.1,X(2),0)),"^")="HLLP" S X=1
       Kill Logic:  D KILL3^HLOTLNK(X(1),X(3))
        Kill Cond:  S X=0 I X(2),$P($G(^HLCS(869.1,X(2),0)),"^")="HLLP" S X=1
       Whole Kill:  K ^HLCS(870,"AD","HLLP")
             X(1):  NODE  (870,.01)  (Subscr 1)  (forwards)
             X(2):  LLP TYPE  (870,2)  (Subscr 2)  (forwards)
             X(3):  HLLP DEVICE  (870,200.01)  (Subscr 3)  (forwards)

  DNS2 (#438)    RECORD    REGULAR    IR    LOOKUP & SORTING
      Short Descr:  Regular index of new-style x-ref on the DNS Domain and Node
                    fields
        Set Logic:  S ^HLCS(870,"DNS2",$E(X(1),1,70),$E(X(2),1,30),DA)=""
       Kill Logic:  K ^HLCS(870,"DNS2",$E(X(1),1,70),$E(X(2),1,30),DA)
       Whole Kill:  K ^HLCS(870,"DNS2")
             X(1):  DNS DOMAIN  (870,.08)  (Subscr 1)  (Len 70)  (forwards)
             X(2):  NODE  (870,.01)  (Subscr 2)  (Len 30)  (forwards)


INPUT TEMPLATE(S):

PRINT TEMPLATE(S):
CAPTIONED                                       USER #0

SORT TEMPLATE(S):

FORM(S)/BLOCK(S):
HL7 LOGICAL LINK              JUL 06, 1999@12:07  USER #0
  HL7 LL HEADER1                DD #870
  HL7 LL BLK1                   DD #870
  HL7 LLP HLLP                  DD #870
  HL7 LLP TCP                   DD #870
  HL7 LLP MAIL                  DD #870
  HL7 LLP X3.28                 DD #870
```

Appendix B – Developer APIs

Below is a complete listing of all HLO APIs, grouped into functional categories. Following the API table are more detailed descriptions and specifications. Each API that is documented for developer use also includes input and output parameters. If a parameter is not specifically documented as being passed by reference, then it is to be passed by value.

HLO API	**Brief Description**
Building Messages	
$$NEWMSG^HLOAPI	Begins a new message.
$$NEWBATCH^HLOAPI	Begins a new batch of messages.
$$ADDMSG^HLOAPI	Begins a new message within a batch.
SET^HLOAPI	Builds a message segment.
$$ADDSEG^HLOAPI	Adds a segment to a message.
$$MOVEMSG^HLOAPI	Moves a message built by a traditional pre-HLO message builder to HLO.
Inserting Data Types	
SETTS^HLOAPI4	Sets a timestamp into a segment.
SETDT^HLOAPI4	Sets a date into a segment.
SETCE^HLOAPI4	Sets a coded element into a segment.
SETHD^HLOAPI4	Sets an HL7 hierarchic designator into a segment.
SETCNE^HLOAPI4	Sets a coded value with no exceptions into a segment.
SETCWE^HLOAPI4	Sets a coded value with exceptions into a segment.
SETAD^HLOAPI4	Sets an address into a segment.
Sending Messages	
$$SENDONE^HLOAPI1	Sends messages to a receiving application.
$$SENDMANY^HLOAPI1	Sends messages to multiple receiving applications.
$$SENDSUB^HLOAPI1	Sends messages to subscribers.
Receiving Messages	
$$STARTMSG^HLOPRS	Initiates message parsing, returns header values.
$$NEXTSEG^HLOPRS	Advances to and parses next segment.
$$NEXTMSG^HLOPRS	Advances to next message, returns header.
$$PARSE^HLOPRS1	Called by NEXTSEG, returns parsed values.
$$GET^HLOPRS	Returns specific segment components.
Parsing Data Types	
GETTS^HLOPRS2	Gets a timestamp from a segment.
GETDT^HLOPRS2	Gets a date from a segment.
GETCE^HLOPRS2	Gets a coded element from a segment.

GETHD^HLOPRS2	Gets an HL7 hierarchic designator from a segment.
GETCNE^HLOPRS2	Gets a coded value with no exceptions from a segment.
GETCWE^HLOPRS2	Gets a coded value with exceptions from a segment.
GETAD^HLOPRS2	Gets an address from a segment.

Application Acknowledgements

$$ACK^HLOAPI2	Initiates an application acknowledgement.
$$SENDACK^HLOAPI2	Sends an application acknowledgement.
$$BATCHACK^HLOAPI3	Begins creation of batch acknowledgement.
$$ADDACK^HLOAPI3	Adds an application acknowledgement to a batch.

Subscription Registries

$$CREATE^HLOASUB	Creates a subscription registry entry.
$$ADD^HLOASUB	Adds a new recipient to a subscription registry.
$$END^HLOASUB	Terminates a recipient from a subscription registry.
$$ONLIST^HLOASUB	Determines if a recipient is on a subscription list.
$$NEXT^HLOASUB	Goes to next recipient on subscription list.
$$INDEX^HLOASUB1	Builds an index of subscription list recipients.
$$FIND^HLOASUB1	Finds a specific subscription list.

HL 1.6 to HLO Message Conversion

$$EN^HLOCNRT	Converts HL 1.6 message to HLO and send.
APAR^HLOCVU	Converts HL 1.6 parameters to HLO.
$$HLNEXT^HLOMSG	Returns segment as a set of lines stored in SEG.

Queue Management

STARTQUE^HLOQUE	Set a “Stop” flag on a specific HLO queue.
STOPQUE^HLOQUE	Remove a “Stop” flag on a specific HLO queue.
$$STOPPED^HLOQUE	Check the status of a queue (started or stopped).

Miscellaneous

$$RESEND^HLOAPI3	Retransmits an outgoing message
$$REPROC^HLOAPI3	Reprocesses an incoming message
$$SETPURGE^HLOAPI3	Resets purge date of a currently stored message

Create Messages

Applications create messages to send using these APIs. These APIs are substantially different from the old APIs in a number of respects:

- Minimal setup is required. Instead, necessary information is passed in as input parameters.
- The messages are built in memory, and then the entire message is written to the message files. If the message is too big, the partially built message is stored to clear the buffer.
- Support for creating batch messages is much improved.
- Support for building segments is provided. Rather than being expected to pass in the complete message via an array, the developer is provided a set of tools for building segments and adding them to the messages. **However, to accommodate existing message building routines, the complete message array may be passed in using the MOVEMSG API.**
- HL7 encoding characters, if found within a data field, are automatically replaced with the proper escape sequence.

Create a New Single Message

Routine: $$NEWMSG^HLOAPI(.PARMS,.HLMSTATE,.ERROR)

Description: This API is used by applications that need to send an HL7 message via HLO. It starts the messaging process.

Input:

PARMS(“COUNTRY”)	Optional	Pass-by-Reference	A three-character country code.
PARMS(“CONTINUATION POINTER”)	Optional	Pass-by-Reference	Indicates a fragmented message.
PARMS(“EVENT”)	Required	Pass-by Reference	A three-character event type.
PARMS(“FIELD SEPARATOR”)	Optional	Pass-by-Reference	Field separator that defaults to "|".
PARMS(“ENCODING CHARACTERS”)	Optional	Pass-by-Reference	Four HL7 encoding characters that defaults to "^~\&".
PARMS(“MESSAGE STRUCTURE”)	Optional	Pass-by Reference	MSH-9, component 3 - a code from the standard HL7 table.
PARMS(“MESSAGE TYPE”)	Required	Pass-by Reference	A three-character message type (required).
PARMS(“PROCESSING MODE”)	Optional	Pass-by Reference	MSH-11, component 2 – A one character code.
PARMS(“VERSION”)	Optional	Pass-by Reference	The HL7 Version ID, for example, "2.4", defaults to 2.4.

Output: (Function call - Returns 1 on success, 0 on failure).

HLMSTATE	Required	Pass-by-Reference	Used by the HL7 package to track the progress of the message. The application MUST NOT directly modify any values in this array.
ERROR	Optional	Pass-by-Reference	Returns an error message on failure.

Create a New Batch Message

Routine: $$NEWBATCH^HLOAPI(.PARMS,.HLMSTATE,.ERROR)

Description: This API is used by applications that need to send a batch of HL7 messages via HLO. It starts the batch building process.

Input:

PARMS("COUNTRY")	Optional	Pass-by-Reference	A three-character country code.
PARMS("FIELD SEPARATOR")	Optional	Pass-by-Reference	Field separator that defaults to "|".
PARMS("ENCODING CHARACTERS")	Optional	Pass-by-Reference	Four HL7 encoding characters that defaults to "^~\&".
PARMS("VERSION")	Optional	Pass-by-Reference	HL7 Version ID, for example, "2.4" that defaults to 2.4.

Output: (Function call - returns 1 on success, 0 on failure).

HLMSTATE	Required	Pass-by-Reference	Used by the HL7 package to track the progress of the message. The application MUST NOT directly modify any values in this array.
ERROR	Optional	Pass-by-Reference	Returns an error message on failure.

Add a New Message to a Batch

Routine: $$ADDMSG^HLOAPI(.HLMSTATE,.PARMS,.ERROR)

Description: This API is used by applications to add a message to a batch that is in the process of being built.

Input:

HLMSTATE	Required	Pass-by-Reference	Array used by the HL7 package to track the progress of the message. The application MUST NOT directly modify any values in this array.
PARMS("EVENT")	Required	Pass-by-Reference	A three-character event type.
PARMS("MESSAGE TYPE")	Required	Pass-by-Reference	A three-character message type.

Output: (Function Call - Returns 1 on success, 0 on failure).

HLMSTATE	Required	Pass-by-Reference	Used by the HL7 package to track the progress of the message. The application MUST NOT directly modify

			any values in this array.
ERROR	Optional	Pass-by-Reference	Returns an error message on failure.

Build Segments

Routine: SET^HLOAPI(SEG,VALUE,FIELD,COMP,SUBCOMP,REP)

Description: This API is used to set a value into a segment that is in the process of being built.

Input:

SEG	Required	Pass-by-Reference	The array where the segment is being built.
VALUE	Required	Pass-by-Value	The individual value to be set into the segment.
FIELD	Optional	Pass-by-Value	The sequence # of the field (optional, defaults to 0). **Note:** FIELD=0 is used to denote the segment type.
COMP	Optional	Pass-by-Value	The # of the component (optional, defaults to 1).
SUBCOMP	Optional	Pass-by-Value	The # of the subcomponent that defaults to 1.
REP	Optional	Pass-by-Value	The occurrence# (optional, defaults to 1). For a non-repeating field, the occurrence # need not be provided, because it would be 1.

Output:

SEG	Required	Pass-by-Reference	Array that is used to build segments.

Example:

D SET(.SEG,"MSA",0) Creates an MSA segment
D SET(.SEG,"AE",1) Will place the value into the array position reserved for the 1st field, 1st occurrence, 1st component, 1st sub-component

Implementation Note This format is used for the segment array built by calls to SET: SEGMENT(<SEQ #>,<occurrence #>,<component #>,<sub-component #>)=<sub-component value>

Routine: SETTS^HLOAPI4(.SEG,.VALUE,FIELD,COMP,REP)

Description: This API sets a value that is a time stamp in FM format into the segment in HL7 format. The degree of precision may be optionally specified. The inserted value will include the timezone if the input included the time. IF the component is specified, then the data type is 'demoted' to a component, and its components are 'demoted' to subcomponents.

Input:

SEG	Required	Pass-by-Reference	The segment that is being built.
VALUE	Required	Pass-by- Reference to also pass the	The date/time in FileMan format. You can optionally specify that the value is to

		"PRECISION"subscript	be rounded down to a particular precision by specifying this subscript: "PRECISION" - Allowed values are: • "S" - second • "M" - minute • "H" - hour • "D" - day • "L" - month • "Y" - year • "" - precision not specified
FIELD	Required	Pass-by-Value	The sequence # of the field.
COMP	Optional	Pass-by-Value	If specified, the data type is 'demoted' to a component value.
REP	Optional, defaults to 1	Pass-by-Value	The occurrence #. For non-repeating fields, this parameter is not necessary.

Output:

SEG	Required	Pass-by-Reference	The segment being built.

Routine: SETDT^HLOAPI4(.SEG,.VALUE,FIELD,COMP,REP)

Description: This API sets a value that is a date in FM format into the segment in HL7 format. The degree of precision may be optionally specified. IF the component is specified, then the data type is 'demoted' to a component, and its components are 'demoted' to subcomponents.

Input:

SEG	Required	Pass-by-Reference	The segment that is being built.
VALUE	Required	Pass-by-Reference	The date to be set into the segment. Optionally, you may specify that the value should be rounded down to a particular precision by specifying this subscript: "PRECISION" (If needed, VALUE must be passed by reference.) Allowed values are: • "D" - day (default value) • "L" - month • "Y" - year
FIELD	Required	Pass-by-Value	The sequence # of the field.
COMP	Optional	Pass-by-Value	If specified, the data type is 'demoted' to a component value.
REP	Optional, defaults to 1	Pass-by-Value	The occurrence #. For non-repeating fields, this parameter is not necessary.

Output:

SEG	Required	Pass-by-Reference	The segment being built.

Routine: SETCE^HLOAPI4(.SEG,.VALUE,FIELD,COMP,REP)

Description: This API sets a value that is an HL7 Coded Element data type (HL7 Section Reference 2.9.3) into the segment in the specified field. IF the component is specified, then the data type is 'demoted' to a component, and its components are 'demoted' to subcomponents.

Input:

SEG	Required	Pass-by-Reference	The segment that is being built.
VALUE	Required	Pass-by-Reference	These subscripts may be passed: • "ID" - the identifier • "TEXT" - • "SYSTEM" - name of the code system • "ALTERNATE ID" - alternate identifier • "ALTERNATE TEXT" • "ALTERNATE SYSTEM" - name of the alternate coding system
FIELD	Required	Pass-by-Value	The sequence # of the field.
COMP	Optional	Pass-by-Value	If specified, the data type is 'demoted' to a component value.
REP	Optional, defaults to 1	Pass-by-Value	The occurrence #. For non-repeating fields, this parameter is not necessary.

Output:

SEG	Required	Pass-by-Reference	The segment being built.

Routine SETHD^HLOAPI4(.SEG,.VALUE,FIELD,COMP,REP)

Description: This API sets a value that is an HL7 Hierarchic Designator data type (HL7 Section Reference 2.9.21) into the segment in the specified field. IF the component is specified, then the data type is 'demoted' to a component, and its components are 'demoted' to subcomponents.

Input:

SEG	Required	Pass-by-Reference	The array where the segment is being built.
VALUE	Required	Pass-by-Reference	These subscripts may be passed: • "NAMESPACE ID" • "UNIVERSAL ID" • "UNIVERSAL ID TYPE"
FIELD	Required	Pass-by-Value	The sequence # of the field.
COMP	Optional	Pass-by-Value	If specified, the data type is 'demoted' to a component value.
REP	Optional,	Pass-by-Value	The occurrence #. For non-repeating fields, this

	defaults to 1		parameter is not necessary.

Output:

SEG	Required	Pass-by-Reference	The segment being built.

Routine: SETCNE^HLOAPI4(.SEG,.VALUE,FIELD,COMP,REP)

Description: This API sets a value that is an HL7 Coded With No Exceptions data type (HL7 Section Reference 2.9.8) into the segment in the specified field. IF the component is specified, then the data type is 'demoted' to a component, and its components are 'demoted' to subcomponents.

Input:

SEG	Required	Pass-by-Reference	The array where the segment is being built.
VALUE	Required	Pass-by-Reference	These subscripts may be passed: • "ID" - the identifier • "TEXT" - • "SYSTEM" - name of the code system • "ALTERNATE ID" - alternate identifier • "ALTERNATE TEXT" • "ALTERNATE SYSTEM" - name of the alternate coding system • "ORIGINAL TEXT"
FIELD	Required	Pass-by-Value	The sequence # of the field.
COMP	Optional	Pass-by-Value	If specified, the data type is 'demoted' to a component value.
REP	Optional, defaults to 1	Pass-by-Value	The occurrence #. For non-repeating fields, this parameter is not necessary.

Output:

SEG	Required	Pass-by-Reference	The segment being built.

Routine: SETCWE^HLOAPI4(.SEG,.VALUE,FIELD,COMP,REP) ;

Description: This API s ets a value that is an HL7 Coded With Exceptions data type (HL7 Section Reference 2.9.11) into the segment in the specified field. IF the component is specified, then the data type is 'demoted' to a component, and its components are 'demoted' to subcomponents.

Input:

SEG	Required	Pass-by-Reference	The array where the segment is being built.
VALUE	Required	Pass-by-Reference	These subscripts may be passed: • "ID" - the identifier

			• "TEXT" - • "SYSTEM" - name of the code system • "ALTERNATE ID" - alternate identifier • "ALTERNATE TEXT" • "ALTERNATE SYSTEM" - name of the alternate coding system • "ORIGINAL TEXT"
FIELD	Required	Pass-by-Value	The sequence # of the field.
COMP	Optional	Pass-by-Value	If specified, the data type is 'demoted' to a component value.
REP	Optional, defaults to 1	Pass-by-Value	The occurrence #. For non-repeating fields, this parameter is not necessary.

Output:

SEG	Required	Pass-by-Reference	The segment being built.

Routine: SETAD^HLOAPI4(.SEG,.VALUE,FIELD,COMP,REP)

Description: This API sets an AD data type (Address, HL7 Section Reference 2.9.1) into the segment in the specified field. It can also be used to set the 1st 8 components of the XAD (Extended Address) data type. IF the component is specified, then the data type is 'demoted' to a component, and its components are 'demoted' to subcomponents.
.

Input:

SEG	Required	Pass-by-Reference	The array where the segment is being built.
VALUE	Required	Pass-by-Reference	These subscripts may be passed: • "STREET1" -street address • "STREET2" - other designation • "CITY" • "STATE" - state or province • "ZIP" - zip or postal code • "COUNTRY" • "TYPE" - address type
FIELD	Required	Pass-by-Value	The sequence # of the field.
COMP	Optional	Pass-by-Value	If specified, the data type is 'demoted' to a component value.
REP	Optional, defaults to 1	Pass-by-Value	The occurrence #. For non-repeating fields, this parameter is not necessary.

Output:

SEG	Required	Pass-by-Reference	The segment being built.

Add a Segment to a Message

Routine: $$ADDSEG^HLOAPI(.HLMSTATE,.SEG,.ERROR)

Description: This API is used to add a segment that has just been built to a message that is still in the process of being built.

Input:

HLMSTATE	Required	Pass-by-Reference	Used by the HL7 package to track the progress of the message. The application MUST NOT directly modify any values in this array.
SEG	Required	Pass-by-Reference	Contains data, must be built by calls to SET prior to calling $$ADDSEG.

Note#1: The message control segments, including the MSH and BHS segments, are added automatically.
Note#2: The 0th field must be a 3 character segment type.
Note#3: SEG is killed upon successfully adding the segment.

Output: (Function call, returns 1 on success, 0 on failure).

HLMSTATE	Required	Pass-by-Reference	Used by the HL7 package to track the progress of the message. The application MUST NOT directly modify any values in this array.
ERROR	Optional	Pass-by-Reference	Returns an error message on failure.

Move Pre-Formatted HL7 Messages into HLO

Routine: $$MOVEMSG^HLOAPI(.HLMSTATE,.ARY)

Description: This API allows segment builders that were created prior to HLO to be used within HLO. If a message was built using any other method than the HLO APIs and resides in an array, it is moved into HLO.

Input:

HLMSTATE	Required	Pass-by-Reference	Created by calling $$NEWMSG^HLOAPI or $$NEWBATCH^HLOAPI The application MUST NOT directly modify any values in this array.
ARY	Required	Pass-by-Value	The name of the local or global array where the message was built. The array is referenced by indirection using ARY.

Send Messages

There are three APIs that application developers may use to send a message to a particular recipient or list of recipients. The APIs are SENDONE, SENDMANY, and SENDSUB. They are described in detail below.

In order for the APIs to send messages, the recipient or list of recipients must be supplied to the APIs. A recipient is identified by the receiving application and the receiving facility. The HLO software allows different methods for specifying the receiving facility. The application developer has several alternatives for specifying the receiving facility, for example, by station number, institution IEN, or link name. The HLO APIs uses the input to resolve and send the VHA domain name and station number, if available, as the receiving facility.

Many applications may route their messages through the local IE, but it is the final destination that should appear in the message header as receiving facility. The IE will use that to correctly route the message. To provide for this, the APIs for sending messages allow a separate “routing” link to be specified.

The three APIs for sending messages appear below.

Send Message to a Single Receiving Application

Routine: $$SENDONE^HLOAPI1(.HLMSTATE,.PARMS,.WHOTO,.ERROR)

Description: This API is used to send a message to a single recipient. The recipient is identified in the message header by the Receiving Facility and the Receiving Application. The message may optionally be routed through an interface engine.

Input:

HLMSTATE	Required	Pass-by-Reference	Used by the HL7 package to track the progress of the message. The application MUST NOT directly modify any values in this array.

PARMS("APP ACK RESPONSE")	Optional	Pass-by Reference	<tag^routine> to call in response to app ack.
PARMS("APP ACK TYPE")	Optional	Pass-by-Reference	<AL,NE> defaults to NE.
PARMS("ACCEPT ACK RESPONSE")	Optional	Pass-by-Reference	<tag^routine> to call in response to a commit ack.
PARMS("ACCEPT ACK TYPE")	Optional	Pass-by-Reference	<AL,NE> defaults to AL.
PARMS("FAILURE RESPONSE")	Optional	Pass-by Reference	<tag^routine> The sending application routine to execute when the transmission of the message fails, i.e., the message can not be sent or no commit ack is received.
PARMS("QUEUE")	Optional	Pass-by-Reference	An application can name its own private queue - just a string under 20 characters, it should be namespaced.
PARMS("SECURITY")	Optional	Pass-by-Reference	Security information to include in the header segment, SEQ-8
PARMS("SENDING APPLICATION")	Required	Pass-by-Reference	Name of sending application (60 max-length).
WHOTO("RECEIVING APPLICATION")	Required	Pass-by Reference	String, 60 char max, required. Specifies a single recipient.

One of the following four parameters is required to identify the Receiving Facility:

WHOTO("FACILITY LINK IEN")			IEN of the logical link.
WHOTO("FACILITY LINK NAME")			Name of the logical link.
WHOTO("INSTITUTION IEN")			Pointer to the INSTITUTION File (#4).
WHOTO("STATION NUMBER")			Station # with suffix.

One of the following two parameters MAY be provided, optionally, to identify the interface engine to route the message through:

WHOTO("IE LINK IEN")			Pointer to a logical link for the interface engine.
WHOTO("IE LINK NAME")			Name of the logical link for the interface engine.

Output: (Function call returns the IEN of the message in HLO MESSAGES File (#778) on success, 0 on failure)

HLMSTATE	Required	Pass-by-Reference	Used by the HL7 package to track the progress of the message. The application MUST NOT directly modify any values in this array.
ERROR	Optional	Pass-by-Reference	On failure, will contain an error message.
PARMS			Killed when the function returns.
WHOTO			Killed when the function returns.

Send Messages to More than One Receiving Application

Routine: $$SENDMANY^HLOAPI1(.HLMSTATE,.PARMS,.WHOTO)

Description: This API is used to send a message to a list of recipients.

Input: Similar to $$SENDONE^HLOAPI1. In $$SENDMANY^HLOAPI1, the WHOTO() array is a list of recipients.

PARMS	Required	Pass-by-Reference	See PARMS definition in the $$SENDONE^HLOAPI1 API.
WHOTO	Required	Pass-by-Reference	Specifies a list of recipients. Each recipient should be listed individually in array WHOTO(i), where i=a recipient. For each recipient the same subscripts may be defined as in the $$SENDONE API. For example: WHOTO(1,"FACILITY LINK NAME")="VAALB" WHOTO(1,"RECEIVING APPLICATION")="MPI" WHOTO(2,"STATION NUMBER")=500 WHOTO(2,"RECEIVING APPLICATION")="MPI"

Output: (Function call returns 1 if a message is queued to be sent to each intended recipient; 0 otherwise.)

PARMS	Required	Pass-by-Reference	Killed when the function returns.
WHOTO	Required	Pass-by-Reference	Returns the status of each message to be sent in the format: (<i>,"QUEUED") - 1 if queued to be sent, 0 otherwise. (<i>,"IEN") – IEN from HLO MESSAGES File (#778) if queued to be sent, null otherwise. (<i>,"ERROR") - Error message if an error was encountered (status=0), and null otherwise.

Send Messages to Subscription Registry Subscribers

Routine: $$SENDSUB^HLOAPI1(.HLMSTATE,.PARMS,.MESSAGES)

Description: This API is used to send messages to a list of receiving applications based on the Subscription Registry.

Input:

HLMSTATE	Required	Pass-by-Reference	See HLMSTATE definition in the $$SENDONE^HLOAPI1 API. The application MUST NOT directly modify any values in this array.
PARMS	Required	Pass-by-Reference	See PARMS definition in the $$SENDONE^HLOAPI1 API. It has one additional subscript, PARMS(“SUBSCRIPTION IEN”).
PARMS("SUBSCRIPTION	Required	Pass-by-	The IEN of an entry in the HLO.

IEN")		Reference	SUBSCRIPTION REGISTRY File (#779.4), defining the intended recipients of this message.

Output: (Function call returns 1 if a message is queued to be sent to each intended recipient, 0 otherwise.)

PARMS			Killed when the function returns.
MESSAGES	Required	Pass-by-Reference	Returns the status of each message to be sent in this format, where the sub-IEN is the IEN of the recipient in the RECIPIENTS sub-file of the HLO SUBSCRIPTION REGISTRY File (#779.4). (<subien>,"QUEUED") - 1 if queued to be sent, 0 otherwise. (<subien>,"IEN") – IEN from HLO MESSAGES File (#778) if queued to be sent, null otherwise. (<subien>,"ERROR") - Error message if an error was encountered (status=0), and null otherwise.

Parse Messages

Start the Parsing Process and Return the Message Header

Routine: $$STARTMSG^HLOPRS(.HLMSTATE,.HLMSGIEN,.HDR)

Description: This API is used to begin the parsing of the message, parse the header, and return the individual values in the array HDR().

Input:

HLMSGIEN	Required	Pass-by-Value	The IEN of the message in HLO MESSAGES File (#778).

Output: (Function returns 1 on success, 0 on failure. Failure would indicate that the message was not found.)

HLMSTATE	Required	Pass-by-Reference	This array is used by the HL7 package to track the progress of parsing the message. The application MUST NOT directly modify any values in this array.
HDR	Optional	Pass-by-Reference	This array contains the results of parsing the message header (See Appendix E).

Advance to and Parse Next Segment

Routine: $$NEXTSEG^HLOPRS(.HLMSTATE,.SEG)

Description: This API is used to advance parsing to the next segment and return the parsed values from that segment in array SEG().

Input:

HLMSTATE	Required	Pass-by-Reference	This array is used by the HL7 package to track the current position in the message. The application MUST NOT touch it.

Output: (Function returns 1 on success, 0 if there are no more segments in this message. For batch messages, a return value of 0 does not preclude the possibility that there are additional individual messages within the batch.

HLMSTATE	Required	Pass-by-Reference	The application MUST NOT directly modify any values in this array.
SEG	Required	Pass-by-Reference	The segment is returned in this parsed array. To retrieve data from the parsed array, use the $$GET^HLOPRS function.

Advance to the Next Message within a Batch

Routine: $$NEXTMSG^HLOPRS(.HLMSTATE,.MSH)

Description: This API is used to advance to the next message within a batch and return the MSH segment for that message.

Input:

HLMSTATE	Required	Pass-by-Reference	This array is used by the HL7 package to track the current position in the message. The application MUST NOT directly modify any values in this array.

Output: (Function returns 1 on success, 0 if there are no more messages.)

HLMSTATE	Required	Passed by Reference	The application MUST NOT directly modify any values in this array.
MSH	Required	Pass-by-Reference	Returns the parsed message header (See Appendix E).

Return Parsed Value

Routine: $$GET^HLOPRS(.SEG,FIELD,COMP,SUBCOMP,REP)

Description: This API is used to get a specified value from a segment that was parsed by $$NEXTSEG^HLOPRS or $$PARSE^HLOPRS1. The FIELD, COMP, SUBCOMP, and REP parameters are optional - if not specified, they default to 1.

Example:
$$GET^HLOPRS(.SEG,1) will return the value of the first field, first component, first subcomponent, in the first occurrence of field 1.

Input:

SEG	Required	Pass-by-Reference	This is the array where the parsed segment was placed by $$NEXTSEG^HLOPRS or $$PARSE^HLOPRS1.
FIELD	Optional	Pass-by-Value	The sequence number of the field which defaults to 1. If 0 (zero) is specified, then the function returns the segment type.
COMP	Optional	Pass-by-Value	The number of the component which defaults to 1.
SUBCOMP	Optional	Pass-by-Value	The number of the subcomponent which defaults to 1.
REP	Optional	Pass-by-Value	The occurrence number which defaults to 1. For a non-repeating field, the occurrence number will always be 1.

Output: (Function returns the requested value on success, null if not valued.)

Routine: GETTS^HLOPRS2(.SEG,.VALUE,FIELD,COMP,REP)

Description: This API gets a segment value that is a time stamp in HL7 format and converts it to FileMan format. IF the data type value includes the timezone, then the time is converted to local time. The degree of precision is optionally returned. IF the component is specified, then the component is parsed for data type rather than at the higher field level.

Input:

SEG	Required	Pass-by-Reference	The array returned by a call to $$NEXTSEG^HLOPRS.
FIELD	Required	Pass-by-Value	The sequence # of the field.
COMP	Optional	Pass-by-Value	If specified, the data type is parsed as a component value.
REP	Optional, defaults to 1	Pass-by-Value	The occurrence #. For non-repeating fields, this parameter is not necessary.

Output:

VALUE	Required	Pass-by-Reference IF subscripts are	The date/time in FileMan format. The PRECISION subscript is optional, if provided the time stamp's precision will be determined.

		used	
"PRECISION" - Value should be Passed-by-Reference if the precision is needed as a return value Expected values are: • "S" - second • "M" - minute • "H" - hour • "D" - day • "L" - month • "Y" - year • "" - precision not specified Note: FM does not allow greater precision than seconds, so greater precision will be rounded down to the second.			

Routine: GETDT^HLOPRS2(.SEG,.VALUE,FIELD,COMP,REP)

Description: This API gets a segment value that is a date in HL7 format and converts it to FileMan format. The degree of precision is optionally returned. IF the component is specified, then the component is parsed for data type rather than at the higher field level.

Input:

SEG	Required	Pass-by-Reference	The array returned by a call to $$NEXTSEG^HLOPRS.
FIELD	Required	Pass-by-Value	The sequence # of the field.
COMP	Optional	Pass-by-Value	If specified, the data type is parsed as a component value.
REP	Optional, defaults to 1	Pass-by-Value	The occurrence #. For non-repeating fields, this parameter is not necessary.

Output:

VALUE	Required	Pass-by-Reference Pass-by-Reference if the precision is needed	The date/time in FileMan format. The "PRECISION" subscript is also returned:
"PRECISION" – VALUE must be passed-by-value if this subscript is needed Expected values are: • "S" – second (not valid for DT) • "M" – minute (not valid for DT) • "H" – hour (not valid for DT) • "D" - day • "L" - month • "Y" - year • "" - precision not specified			

Routine: GETCE^HLOPRS2(.SEG,.VALUE,FIELD,COMP,REP)

Description: This API gets an CE data type(Coded Element, HL7 Section Reference 2.9.8)from the specified field. IF the component is specified, then the component is parsed for data type rather than at the higher field level.

Input:

SEG	Required	Pass-by-Reference	The array returned by a call to NEXTSEG^HLOPRS.
FIELD	Required	Pass-by-Value	The sequence # of the field.
COMP	Optional	Pass-by-Value	If specified, the data type is parsed as a component value.
REP	Optional, defaults to 1	Pass-by-Value	The occurrence #. For non-repeating fields, this parameter is not necessary.

Output:

VALUE	Required	Pass-by-Reference	These subscripts are returned: • "ID" - the identifier • "TEXT" - • "SYSTEM" - name of the code system • "ALTERNATE ID" - alternate identifier • "ALTERNATE TEXT" • ALTERNATE SYSTEM" - name of the alternate coding system

Routine: GETHD^HLOPRS2(.SEG,.VALUE,FIELD,COMP,REP)

Description: This API gets an HD data type (Hierarchic Designator, HL7 Section Reference 2.9.21) from the specified field. IF the component is specified, then the component is parsed for data type rather than at the higher field level.

Input:

SEG	Required	Pass-by-Reference	The array returned by a call to NEXTSEG^HLOPRS.
FIELD	Required	Pass-by-Value	The sequence # of the field.
COMP	Optional	Pass-by-Value	If specified, the data type is parsed as a component value.
REP	Optional, defaults to 1	Pass-by-Value	The occurrence #. For non-repeating fields, this parameter is not necessary.

Output:

VALUE	Required	Pass-by-Reference	These subscripts are returned: • "NAMESPACE ID" • "UNIVERSAL ID" • "UNIVERSAL ID TYPE"

Routine: GETCNE HLOPRS2(.SEG,.VALUE,FIELD,COMP,REP)

Description: This API gets an CNE data type (Coded With No Exceptions, HL7 Section Reference 2.9.8) from the specified field. IF the component is specified, then the component is parsed for data type rather than at the higher field level.

Input:

SEG	Required	Pass-by-Reference	The array returned by a call to NEXTSEG^HLOPRS.
FIELD	Required	Pass-by-Value	The sequence # of the field.
COMP	Optional	Pass-by-Value	If specified, the data type is parsed as a component value.
REP	Optional, defaults to 1	Pass-by-Value	The occurrence #. For non-repeating fields, this parameter is not necessary.

Output:

VALUE	Required	Pass-by-Reference	These subscripts are returned: • "ID" - the identifier • "TEXT" - • "SYSTEM" - name of the code system • "ALTERNATE ID" - alternate identifier • "ALTERNATE TEXT" • ALTERNATE SYSTEM" - name of the alternate coding system • "SYSTEM VERSION" - version ID of the coding system • "ALTERNATE SYSTEM VERSION" - version ID of the alternate coding system • "ORIGINAL TEXT"

Routine: GETCWE^HLOPRS2(.SEG,.VALUE,FIELD,COMP,REP) --

Description: This API gets an CWE data type (Coded With Exceptions, HL7 Section Reference 2.9.11) from the specified field. IF the component is specified, then the component is parsed for data type rather than at the higher field level.

Input:

SEG	Required	Pass-by-Reference	The array returned by a call to NEXTSEG^HLOPRS.
FIELD	Required	Pass-by-Value	The sequence # of the field.
COMP	Optional	Pass-by-Value	If specified, the data type is parsed as a component value.
REP	Optional, defaults to 1	Pass-by-Value	The occurrence #. For non-repeating fields, this parameter is not necessary.

Output:

VALUE	Required	Pass-by-Reference	These subscripts are returned: • "ID" - the identifier • "TEXT" - • "SYSTEM" - name of the code system • "ALTERNATE ID" - alternate identifier • "ALTERNATE TEXT" • ALTERNATE SYSTEM" - name of the alternate coding system • "SYSTEM VERSION" - version ID of the coding system • "ALTERNATE SYSTEM VERSION" - version ID of the alternate coding system • "ORIGINAL TEXT"

Routine: GETAD^HLOPRS2(.SEG,.VALUE,FIELD,COMP,REP)

Description: This API gets an AD data type (Address, HL7 Section Reference 2.9.1) from the specified field. It can also be used to get the 1st 8 components of the XAD (Extended Address) data type. IF the component is specified, then the component is parsed for the address rather than at the higher field level.

Input:

SEG	Required	Pass-by-Reference	The array returned by a call to NEXTSEG^HLOPRS.
FIELD	Required	Pass-by-Value	The sequence # of the field.
COMP	Optional	Pass-by-Value	If specified, the data type is parsed as a component value.
REP	Optional, defaults to 1	Pass-by-Value	The occurrence #. For non-repeating fields, this parameter is not necessary.

Output:

VALUE	Required	Pass-by-Reference	These subscripts are returned: • "STREET1" -street address • "STREET2" - other designation • "CITY" • "STATE" - state or province • "ZIP" - zip or postal code • "COUNTRY" • "TYPE" - address type • "OTHER" - other geographic designation

Create Application Acknowledgements

An application must use these APIs to create an application acknowledgement. The developer may accept the defaults in the API to generate a very simple ACK message with a single MSA segment, but may also choose to specify any other type of message containing any number of segments, though the MSA segment is required. If the developer does not build an MSA segment, the MSA segment is added automatically. The returning receiving application and receiving facility are automatically determined from the MSH or BHS segment of the original message.

The APIs for generating an application acknowledgement follow. There is a set of APIs for acknowledging a single message, and another set of APIs for acknowledging a batch message.

- To generate a single application acknowledgement:
  - Call $$ACK^HLOAPI2 to initiate an application acknowledgement.
  - Optionally, call $$ADDSEG^HLOAPI to add whatever additional segments are needed
  - Call $$SENDACK^HLOAPI2 to send an application acknowledgement.

- To generate a batch application acknowledgement in response to a batch message:
  - Call $$BATCHACK^HLOAPI3 to initiate a batch application acknowledgement.
  - Optionally, call $$ADDACK^HLOAPI3 to add each individual acknowledgement message to the batch.
  - For each call to $$ADDACK^HLOAPI3, the application may optionally call $$ADDSEG^HLOAPI to add additional segments.
  - Call $$SENDACK^HLOAPI2 to send a batch application acknowledgement.

NOTE: The HL7 standard provides a variety of methods for acknowledging a batch message. In one method, the return batch contains only negative acknowledgement messages; the absence of an individual acknowledgement message within the batch infers that the message was accepted. Another method is to return both negative and positive acknowledgement messages in the batch for every message. Depending on the interface, the developer may need to use a particular method.

Initiate the Application Acknowledgement

Routine: $$ACK^HLOAPI2(.HLMSTATE,.PARMS,.ACK,.ERROR)

Description This API is used to initiate (but does not complete) an application acknowledgement. This API should NOT be called for batch messages, instead use $$BATCHACK^HLOAPI3.

NOTE: An important parameter involving application acknowledgements is the HDR("APP ACK TYPE"). This parameter is returned from $$STARTMSG^HLOPRS and will have a value of "NE" ("never") or "AL" ("always"). If the value of HDR("APP ACK TYPE") is "NE," the process should exit before the acknowledgement is created and sent. If the value of HDR("APP ACK TYPE") is "AL", then the application acknowledgement should be initiated.

Input:

HLMSTATE	Required	Pass-by-Reference	Obtained by calling $$STARTMSG^HLOPRS when parsing the original message. The application MUST NOT directly modify any values in this array.

These subscripts may be defined:

PARMS("ACK CODE")	Optional	Pass-by-Reference	MSA-1 contains AA, AE, or AR.
PARMS("ERROR MESSAGE")	Optional	Pass-by-Reference	MSA-3, should be used only if AE or AR.
PARMS("ACCEPT ACK RESPONSE")	Optional	Pass-by-Reference	The <tag^routine> to call in response to a commit ack
PARMS("ACCEPT ACK TYPE")	Optional	Pass-by-Reference	{AL,NE} which defaults to AL.
PARMS("CONTINUATION POINTER")	Optional	Pass-by-Reference	Indicates a fragmented message.
PARMS("COUNTRY")	Optional	Pass-by-Reference	The three-character country code
PARMS("EVENT")	Optional	Pass-by-Reference	The three-character event type which defaults to the event code of the original message.
PARMS("ENCODING CHARACTERS")	Optional	Pass-by-Reference	The four HL7 encoding characters which defaults to "^~\&".
PARMS("FAILURE RESPONSE")	Optional	Pass-by-Reference	The <tag>^<routine> that the sending application routine should execute if the transmission of the message fails, i.e., the message can not be sent or a requested commit ack is not received.
PARMS("FIELD SEPARATOR")	Optional	Pass-by-Reference	Field separator which defaults to "|".
PARMS("MESSAGE TYPE")	Optional	Pass-by-Reference	If not defined, ACK is used.

PARMS("MESSAGE STRUCTURE CODE")	Optional	Pass-by-Reference	
PARMS("QUEUE")	Optional	Pass-by-Reference	An application can name its own private queue (a string under 20 characters, it should be namespaced). The default is the name of the queue of the original message
PARMS(“SECURITY")	Optional	Pass-by-Reference	Security information to include in the header segment, SEQ-8.
PARMS("VERSION")	Optional	Pass-by-Reference	The HL7 Version ID which defaults to 2.4.

Output: (Function call returns 1 on success, 0 on failure.).

PARMS			Killed when the function returns.
ACK	Required	Pass-by-Reference	The acknowledgement message being built.
ERROR	Optional	Pass-by-Reference	An error message.

Send the Application Acknowledgement.

Routine: $$SENDACK^HLOAPI2(.ACK,.ERROR)

Description: This API is used to send the acknowledgement message that was initiated by a call to $$ACK^HLAPI2 or a batch of acknowledgement messages that was initiated by a call to $$BATCHACK^HLOAPI3.

Input:

ACK	Required	Pass-by-Reference	An array that contains the acknowledgement message.

Output: (Function call returns 1 on success, 0 on failure)

ERROR	Optional	Pass-by-Reference	An error message, returned if the function fails.

Start Batch Application Acknowledgement

Routine: $$BATCHACK^HLOAPI3(.HLMSTATE,.PARMS,.ACK,.ERROR)

Description: This API is used to initiate a batch application acknowledgement. It will contain individual application acknowledgements for each message in the original batch message that was received via HLO. Individual acknowledgements are placed in this batch by calling $$ADDACK^HLOAPI3, then the batch of acknowledgements is actually sent by calling $$SENDACK^HLOAPI2.

Input:

HLMSTATE	Required	Pass-by-Reference	Obtained by calling $$STARTMSG^HLOPRS when parsing the original message. The application MUST NOT directly modify any values in this array.

These subscripts may be defined:

PARMS("ACCEPT ACK RESPONSE")	Optional	Pass-by-Reference	<tag^routine> to call in response to a commit ack.
PARMS("ACCEPT ACK TYPE")	Optional	Pass-by-Reference	<AL,NE> which defaults to AL.
PARMS("COUNTRY")	Optional	Pass-by-Reference	A three-character country code from the HL7 standard table.
PARMS("ENCODING CHARACTERS")	Optional	Pass-by-Reference	The four HL7 encoding characters; optional, defaults to "^~\&".
PARMS("FAILURE RESPONSE")	Optional	Pass-by-Reference	The <tag>^<routine> that the sending application routine should execute if the transmission of the message fails, i.e., the message can not be sent or a requested commit ack is not received.
PARMS("FIELD SEPARATOR")	Optional	Pass-by-Reference	Field separator; optional, defaults to "|".
PARMS("QUEUE")	Optional	Pass-by-Reference	An application can name a private queue (a string under 20 characters, it should be namespaced). The default is the name of the queue of the original message.
PARMS("SECURITY")	Optional	Pass-by-Reference	Security information to include in the header segment, SEQ-8.
PARMS("VERSION")	Optional	Pass-by-Reference	The HL7 Version ID which defaults to 2.4.

Output: (Function call returns 1 on success, 0 on failure.)

PARMS			Killed when the function returns.
ACK	Required	Pass-by-Reference	The acknowledgement message being built.
ERROR	Optional	Pass-by-Reference	An error message.

Add an Application Acknowledgement to a Batch Acknowledgement Message

Routine: $$ADDACK^HLOAPI3(.ACK,.PARMS,.ERROR)

Description: This API is used to add an application acknowledgement to a batch acknowledgement message that was started by calling $$BATCHACK^HLOAPI3.

Input:

ACK	Required	Pass-by-Reference	The batch of acknowledgements that is being built.
PARMS("ACK CODE")	Required	Pass-by-Reference	Allowed values are AA, AE, or AR and is used to populate MSA-1.
PARMS("ERROR MESSAGE")	Optional	Pass-by-Reference	If the "ACK CODE" is AE or AR then the value passed in this parameter is used to populate MSA-3.
PARMS("EVENT")	Optional	Pass-by-Reference	A three-character event type (optional, defaults to the event type of the original message).
PARMS("MESSAGE CONTROL ID")	Required	Pass-by-Reference	The message control ID of the original individual message within the batch that is being acknowledged.
PARMS("MESSAGE STRUCTURE CODE")	Optional	Pass-by-Reference	
PARMS("MESSAGE TYPE")	Optional	Pass-by-Reference	A three-character message type that defaults to ACK.
PARMS("SECURITY")	Optional	Pass-by-Reference	Security information to include in the header segment SEQ-8.

Output: (Function call returns 1 on success, 0 on failure.)

PARMS			Killed when the function returns.
ACK	Required	Pass-by-Reference	The acknowledgement message being built.
ERROR	Optional	Pass-by-Reference	An error message.

Manage Subscription Registry

An entry in the HLO Subscription Registry is similar to a mailing list in that it can be used to send HL7 messages to multiple recipients. The application that creates the entry is the owner and is responsible for maintaining it.

Create an Entry in the Subscription Registry

Routine: $$CREATE^HLOASUB(OWNER,DESCRIPTION,.ERROR)

Description: This API is used to create a new entry in the HLO SUBSCRIPTION REGISTRY File (#779.4).

Input:

OWNER	Required	Pass-by-Value	The name of the owning application. It should be prefixed with the package namespace to ensure uniqueness.
DESCRIPTION	Required	Pass-by-Value	A brief (1 line) description.

Output: (Function call returns the new IEN in the HLO SUBSCRIPTION REGISTRY File (#779.4) if successful, 0 if error.)

ERROR	Optional	Pass-by-Reference	An error message, returned if the function fails.

Add a New Recipient to a Subscription Registry Entry

Routine: $$ADD^HLOASUB(IEN,.WHO,.ERROR)

Description: This API is used to add a new recipient to the subscription list.

Input:

IEN	Required	Pass-by-Value	The IEN of the entry in the HLO SUBSCRIPTION REGISTRY File (#779.4).
WHO	Required	Pass-by-Reference	An array containing the information for a single new recipient to be added to the list. These subscripts are allowed: WHO("RECEIVING APPLICATION") - String, 60 char max, required.

One of the following four parameters must be provided to identify the Receiving Facility:

WHO("FACILITY LINK IEN")			IEN of the logical link.
WHO("FACILITY LINK NAME")			Name of the logical link.
WHO("INSTITUTION IEN")			Pointer to the INSTITUTION File (#4).
WHO("STATION NUMBER")			Station # with suffix.

ONE of the following two parameters MAY be provided - optionally - to identify the interface engine to route the message through:

WHO("IE LINK IEN")			Pointer to a logical link for the interface engine.
WHO("IE LINK NAME")			Name of the logical link for the interface engine.

Output: (Function call returns the IEN of the recipient from the RECIPIENTS multiple, 0 on failure.)

WHO			Killed when the function returns.
ERROR	Optional	Pass-by-Reference	On an error, one of the following error messages may be returned: "SUBSCRIPITON REGISTRY ENTRY NOT FOUND" "RECEIVING FACILTY LOGICAL LINK NOT FOUND" "RECEIVING APPLICATION NOT FOUND" "INTERFACE ENGINE LOGICAL LINK PROVIDED BUT NOT FOUND" "FAILED TO ACTIVATE SUBSCRIBER"

Terminate a Recipient from a Subscription Registry Entry

Routine: $$END^HLOASUB(IEN,.WHO)

Description: This API is used to terminate a recipient from the subscriber list. The recipient is not deleted, but the DATE/TIME TERMINATED field is entered with the current date/time.

Input:

IEN	Required	Pass-by-Value	The IEN of the HLO SUBSCRIPTION REGISTRY File (#779.4) entry.
WHO	Required	Pass-by-Reference	If WHO("SUBIEN") is defined, then it should be the IEN of the sub-record to be terminated. Otherwise, set the parameters as per $$ADD^HLOASUB.

Output: (Function call returns 1 on success or if subscriber was not found, 0 on failure to update termination date)

WHO			Killed when the function returns.

Check Subscription Registry entry for a recipient

Routine: $$ONLIST^HLOASUB(IEN,LINKIEN,APPNAME,FAC1,FAC2,FAC3)

Description: This API is used to locate an individual recipient within a subscription registry entry

Input:

IEN	Required	Pass-by-Value	The IEN of the HLO SUBSCRIPTION REGISTRY File (#779.4) entry.
LINKIEN	Required	Pass-by-Value	IEN of the logical link.
APPNAME	Required	Pass-by-Value	Name of the receiving application.
FAC1	Required	Pass-by-Value	Component 1 of the receiving facility.
FAC2	Required	Pass-by-Value	Component 2 of the receiving facility.
FAC3	Required	Pass-by-Value	Component 3 of the receiving facility.

Output: Function call returns the IEN of the recipient from the RECIPIENTS multiple, 0 on failure.

Retrieve Next Recipient from a Subscription Registry Entry

Routine: $$NEXT^HLOASUB(IEN,.RECIP)

Description: This API is used to loop through a subscription list. It ignores recipients that have been terminated from the specified subscription registry entry.

Input:

IEN	Required	Pass-by-Value	The IEN of the HLO SUBSCRIPTION REGISTRY File (#779.4) entry.
RECIP	Required	Pass-by-Reference	If empty, the function finds the first recipient in the specified subscription registry entry, else it uses the value of RECIP("SUBIEN") to find the next recipient.

Output: Function call returns the IEN of the recipient from the RECIPIENTS multiple, 0 on failure. If no more recipients are found, then SUBIEN=-1 and all other subscripts are set to null.

RECIP			Returns the next recipient in the specified subscription registry entry. These subscripts are returned: RECIP("LINK IEN") - IEN of the logical link RECIP("LINK NAME") - Name of the logical link RECIP("RECEIVING APPLICATION") - Name of receiving application RECIP("RECEIVING FACILITY",1) - Component 1 of receiving facility RECIP("RECEIVING FACILITY",2) - Component 2 of receiving facility RECIP("RECEIVING FACILITY",3) - Component 3 of receiving facility RECIP("SUBIEN") - The IEN in the multiple, used to find the next recipient in the specified subscription registry entry.

Build a Subscription Registry Index

Routine: $$INDEX^HLOASUB1(IEN,.PARMARY)

Description: This API is used by an application to build an index of its subscriptions. This is optional, but using this function allows the application to easily find subscriptions without storing the IEN.

Input:

IEN	Required	Pass-by-Value	The IEN of the HLO SUBSCRIPTION REGISTRY File (#779.4) entry.
PARMARY	Required	Pass-by-Reference	An array of parameters with which to build the index. The format is: PARMARY(1)=<first parameter>, PARMARY(2)=<second parameter> If PARMARY(i)=null, the parameter is translated to a single space.

Output: Function returns 1 on success, 0 on failure.

PARMARY			Killed when the function returns.

Find a Subscription Registry Entry

Routine: $$FIND^HLOASUB1(OWNER,.PARMARY)

Description: This API is used by an application to find a subscription registry entry. The application must maintain a private index in order to utilize this function, via $$INDEX^HLOASUB1.

Input:

OWNER	Required	Pass-by-Value	The name of the owning application, as entered in $$CREATE^HLOSUB.
PARMARY	Required	Pass-by-Reference	An array of parameters with which to build the index. The format is: PARMARY(1)=<first parameter>, PARMARY(2)=<second parameter> If PARMARY(i)=null, the parameter is translated to a single space.

Output: Function call returns the IEN of the subscription list if found, 0 otherwise.

PARMARY			Killed when the function returns.

Miscellaneous

Resend a Message

Routine: $$RESEND^HLOAPI3(MSGIEN,.ERROR)

Description: This API is used to retransmit a message by making a copy of the message, reusing all the original parameters. Then the message is placed in the same outgoing queue as the original message.

Input:

MSGIEN	Required	Pass-by-Value	The IEN of the HLO MESSAGES File (#778) entry that is to be resent.

Output: (Function returns the IEN of the new message in HLO MESSAGES File (#778) on success, 0 on failure).

ERROR	Optional	Pass–by-Reference	on failure, will contain an error message.

Reprocess a Message

Routine: $$REPROC^HLOAPI3(MSGIEN,.ERROR)

Description: This API is used to place a message back on an incoming queue for reprocessing.

Input:

MSGIEN	Required	Pass-by-Value	The IEN of the HLO MESSAGES File (#778) entry that is to be reprocessed.

Output: (Function returns 1 on success, 0 on failure)

ERROR	Optional	Pass-by-Reference	On failure, will contain an error message.

Reset the Purge Date and Time for a Message

Routine: $$SETPURGE^HLOAPI3(MSGIEN,TIME)

Description: This API is used to reset the scheduled purge date/time of a message.

Input:

MSGIEN	Required	Pass-by/Value	The IEN of the HLO MESSAGES File (#778) entry that is to be modified.
TIME	Optional	Pass-by-Reference	The new scheduled purge date/time. If not defined, defaults to NOW.

Output: (Function returns 1 on success, 0 on failure).

Conversion APIs - HL 1.6 to HLO

Advance to and Return Next Segment for Incoming Message

The new function $$HLNEXT is used to step through the segments of a message stored in the new HLO data structures. It is meant to replicate the current HL 1.6 'X HLNEXT' to ease conversion.

However, there are two differences between 'X HLNEXT' and the new HLO API, $$HLNEXT:

1. For batch messages, $$HLNEXT does not traverse to the next message in the batch as 'X HLNEXT' does. Instead, a new HLO API is available to transverse to the next message, $$NEXTMSG^HLOPRS.
2. $$HLNEXT always returns the data in an array. Whereas, 'X HLNEXT' sets the initial value into a non-subscripted variable (SEG) and only returns an array if the segment is greater than 245 characters.

Routine: $$HLNEXT^HLOMSG(.HLMSTATE,.SEG)

Description: This API is used by messaging applications that were developed prior to HLO. This is the HLO equivalent of executing HLNEXT and is used to step through the segments in an HL7 formatted message.

Input:

HLMSTATE	Required	Pass-by-Reference	This array is used by the HL7 package to track the current position in the message. The application MUST NOT directly modify any values in this array.

Output: (Function returns 1 on success, 0 if there are no more segments in this message. For batch messages, a return value of 0 does not preclude the possibility that there are additional individual messages within the batch.)

HLMSTATE	Required	Pass-by-Reference	This array is used by the HL7 package to track the current position in the message. The application MUST NOT directly modify any values in this array.
SEG	Required	Pass-by-Reference	The segment is returned in this array.

Convert HL1.6 Outgoing Message to HLO Outgoing Message

Routine: $$EN^HLOCNRT(HLOPRTCL,.ARYTYP,.HLP)

Description: This API is used to take a current HL 1.6 message (Pre-HLO) that follows the standard HL 1.6 methodology, convert it to use the HLO engine, and place the message onto an HLO message queue. The function call EN^HLOCNRT is the HLO equivalent of the HL 1.6 function call GENERATE^HLMA.

Input:

HLOPRTCL	Required	Pass-by-Value	Protocol IEN or Protocol Name
ARYTYP	Required	Pass-by-Value	An array type: “GM” is used for a global array and “LM” is used for a local array.
HLP	Optional	Pass-by-Reference	Additional MSH parameters. For example: HLP(“SECURITY”) - Current HL 1.6 parameter HLP(“CONTPTR”) - Current HL 1.6 parameter HLP(“QUEUE”) - This is not actually a current HL 1.6 parameter but can be added to the HLP array to allow a converted application to define HLO private queues.

Output: Function returns 1 on success and 0^error code^error description on failure.

Convert HL 1.6 Parameters to HLO Parameters for Outgoing Message

Routine: APAR^HLOCVU(HLOEID,.APPARMS,.WHO,.WHOTO)

Description: This API is used to retrieve HL 1.6 parameters from the existing HL 1.6 protocol and translate them to the HLO parameter format.

Input:

HLOEID	Required	Pass-by-Reference	The Event Protocol IEN.

Output: (see table below)

APPARMS	Required	Pass-by-Reference	An array containing the necessary HLO message parameters.

Will return one of the following:

WHO		Pass-by-Reference	For single HLO message recipients, receiving application parameters.
WHOTO		Pass-by-Reference	For multiple HLO message recipients, receiving application parameters.

WARNING – HLOCVU expects the values of HLFS and HLECH as set from INIT^HLFNC2. Changing these variables before the call to HLOCNRT or HLOCVU may negatively impact the conversion of the HL7 message.

Specific Translation From HL 1.6 Parameters to HLO Parameters (as follows):

HL 1.6 APPLICATION PROTOCOL	=>	HLO APPARMS ARRAY PARAMETERS
COUNTRY CODE	=>	APPARMS(“COUNTRY”)
APPLICATION ACK TYPE	=>	APPARMS(“APP ACK TYPE”)
EVENT TYPE	=>	APPARMS(“EVENT”)
SENDING APPLICATION	=>	APPARMS(“SENDING APPLICATION”)
TRANSACTION MESSAGE TYPE	=>	APPARMS(“MESSAGE TYPE”)
VERSION ID	=>	APPARMS(“VERSION”)
HL7 FIELD SEPARATOR	=>	APPARMS(“FIELD SEPARATOR”)
HL7 ENCODING CHARACTERS	=>	APPARMS(“ENCODING CHARACTERS”)

HL 1.6 Passed Parameters	=>	HLO APPARMS ARRAY PARAMETERS
HLP("SECURITY")	=>	APPARMS("SECURITY")
HLP("CONTPTR")	=>	APPARMS("CONTINUATION POINTER")
HLP(“QUEUE”)*	=>	APPARMS(“QUEUE”)

NOTE: HLP(“QUEUE”) is not actually a current HL 1.6 parameter, but can be added to the HLP array to allow a converted application to define HLO private queues.

For Sending Messages To One Application

RECEIVING APPLICATION	=>	WHO(“RECEIVING APPLICATION”)
LOGICAL LINK	=>	WHO(“FACILITY LINK NAME”)

For Sending Messages To Multiple Applications (where “n” is a numeric index (0,1,2,…))

RECEIVING APPLICATION	=>	WHOTO(n,”RECEIVING APPLICATION”)
LOGICAL LINK	=>	WHOTO(n,”FACILITY LINK NAME”)

Queue Management

NOTE: The three queue APIs are intended for use in a KIDS install when a VistA module uses a specific HLO queue. Pre and post-build processes that use these APIs should be developed to stop and restart the queue. This will prevent errors from occurring during the installation process.

When stopping a queue, the queue processor only checks for the flag after processing the current message. A hang may need to be included after setting a "Stop" flag on a named queue depending on the size of the messages being processed.

Stop a Queue

Routine: STOPQUE^HLOQUE(DIR,QUEUE)

Description: This API is used to set a "Stop" flag on a named queue.

Input:

DIR	Required	Pass-by-Value	Direction of queue. Values are "IN" or "OUT".
QUEUE	Required	Pass-by-Value	The name of the queue to be stopped.

Start a Queue

Routine: STARTQUE^HLOQUE(DIR,QUEUE)

Description: This API is used to remove a "Stop" flag on a named queue.

Input:

DIR	Required	Pass-by-Value	Direction of queue. Values are "IN" or "OUT".
QUEUE	Required	Pass-by-Value	The name of the queue to be started.

Check the status of a queue

Routine: $$STOPPED^HLOQUE(DIR,QUEUE)

Description: This API is used to check if a "Stop" flag is currently set on a named queue.

Input:

DIR	Required	Pass-by-Value	Direction of queue. Values are "IN" or "OUT".
QUEUE	Required	Pass-by-Value	The name of the queue to be checked..

Output: (Function returns 1 is the queue is stopped, 0 otherwise.)

Appendix C – HLO SACC Exemptions

NOTE: These SACC exemptions are still awaiting approval…..

1. ^HLTMP
2. $INCREMENT
3. OPEN => for HLO routines
4. CLOSE => for HLO routines
5. *READ => this is covered by existing SACC exemption
6. $ZA
7. string lengths greater than 245
   a. Cache 1024
   b. DSM 512
   c. All others – 245

Exemptions that have been approved by SACC:

NOTE: THIS SECTION IS INCOMPLETE……

HEALTH LEVEL SEVEN

1 STANDARD SECTION: 1 ANSI
DATE GRANTED: JUL 26,1995
Permanent exemption to use the following 1994 M standard language features:

Set $Extract
Merge
Two Argument $Order (reverse $o)
Use of routines with routine size greater than 5K

2 STANDARD SECTION: 2D2 * & # READs
DATE GRANTED: JUL 26,1995
HI7 V1.6 has been granted an exemption to use the *READ command for user input, thus allowing a user input read to terminate with some other character than the carriage return.

3 DATE GRANTED: NOV 29,1996
Section 2.3.1.6 referring to setting the variable IO for routine HLCSTCP1;

Section 2.3.3.2 regarding the use $ZC intrinsic system variable for routine HLCSTC11;

Section 2.4.3 CLOSE Command for routines HLCSTCP1, HLCSTCP2, HLCSTC11;

Section 2.4.9 OPEN Command for routines HLCSTCP1, HLCSTC11;

Section 2.4.12 USE Command with parameters for routines HLCSTCP1, HLCSTCP2, HLCSTC11.

Appendix D - The HLO Process Registry

DANGER: This section is for information only. Do not use the information in this section to modify the HLO Process Registry. Only the two fields, ACTIVE and DEDICATED LINK should ever be modified by IRMs or application developers.

However, this section may be useful for troubleshooting problems with the HLO system.

HLO Processes

The processes involved with the HLO system are:

- PROCESS MANAGER – The HLO Process Manager is the primary process of the HLO system. All other processes except for the VMS TCP LISTENER process are started, monitored, and stopped by this process.
- SINGLE LISTENER – The server process for the listener that is designed to only accept a single connection, receive messages, and put them on designated inbound queues.
- TASKMAN MULTI-LISTENER – The server process for the listener that is started from TaskMan and can accept multiple connections, receive messages, and put them on designated inbound queues. Used for sites which are not running under OpenVMS.
- VMS TCP LISTENER – The server process for the TCP/IP Services for Open VMS systems only. It accepts messages and puts them on designated inbound queues. This process is started by the OpenVMS TCP/IP service.
- OUTGOING CLIENT LINK – These processes initiate the sending of messages that are pending on outbound queues.
- INCOMING QUEUES – These processes take received messages from inbound queues and invoke the appropriate applications' logic.
- CHECK PROCESS COUNTS – This process intermittently recounts the running and scheduled processes.
- CLIENT MESSAGE UPDATES – This process updates message status information as requested by other processes.
- PURGE OLD MESSAGES – This process removes messages that are older than the defined purge date/time.
- REMOVE BAD MESSAGES – This process intermittently checks for queues that have been down for a substantial length of time. If the message at the head of the queue has repeatedly failed transmission, then the process removes that message and changes the message status to Transmission Failure.
- TOTAL MESSAGE COUNTS – This process totals message counts by message type for hourly, daily, and monthly intervals.

The HLO Process Manager

The HLO Process Manager is the HLO component that manages all of the processes that make up the HLO system. It uses the HLO PROCESS REGISTRY File (#779.3) which contains information about each type of process under the Process Manager's control. Since only the HLO development team should require access to the HLO PROCESS REGISTRY File (#779.3), it does not require a special user input screen.

Processes running under the Process Manager are dynamic; they start and stop in response to changing workload.

HLO PROCESS REGISTRY File (#779.3)

This file contains these fields:

PROCESS NAME	FREE TEXT	A unique name for the type of process.
ACTIVE	SET (YES or NO)	A flag that indicates whether or not this type of process is active under the HLO Process Manager. Some processes may not apply to some systems; for example, a particular site may not use the TaskMan multi-listener.
MINIMUM ACTIVE PROCESSES	NUMERIC	This field indicates the minimum number of concurrent processes of this type. The actual number of processes changes as the HLO Process Manager starts and stops processes in response to changes in workload, but there should always be at least as many as this field indicates.
MAXIMUM ACTIVE PROCESSES	NUMERIC	This field indicates the maximum number of concurrent processes of this type. The actual number of processes changes as the HLO Process Manager starts and stops processes in response to changes in workload, but it should never exceed the number specified in this field.
SCHEDULING FREQUENCY (minutes)	NUMERIC	This is how long the Process Manager should wait between checks to see if another process of this type should be started.
DT/TM LAST STARTED OR STOPPED	DATE/TIME	The date and time when a process of this type was last started or stopped.
HANG TIME (seconds)	NUMERIC	This is how long a process should wait between attempts to find work to do.
GET WORK FUNCTION (TAG)	FREE TEXT	The M entry point for the process's GET WORK function.
GET WORK FUNCTION (ROUTINE)	FREE TEXT	The routine in which the process's GET WORK function is located.
DO WORK FUNCTION (TAG)	FREE TEXT	The M entry point for the process's DO WORK function.
DO WORK FUNCTION (ROUTINE)	FREE TEXT	The routine in which the process's DO WORK function is located.
MAX TRIES FINDING WORK	NUMERIC	The number of times the process looks for work before quitting. It will hang between attempts the specified length of time.

PERSISTENT	SET (YES or NO)	Setting this field to YES results in the process being made persistent via the TaskMan persistent parameter.
DEDICATED LINK	FREE TEXT	The primary use of this field is for TCP/IP listener processes, and indicates on which port (via the HL Logical Link) the process should be listening. However, it could be used to dedicate a client link process to a particular link.
VMS TCP SERVICE	SET (YES or NO)	VMS services are not started or stopped via the HLO Process Manager. However, on a VMS system, these services are an important part of the HLO system, and an entry in the HLO PROCESS REGISTRY File (#779.3) should be created for them. The Process Manager will verify that the listener is running.

Process Manager Operation

The Process Manager runs continuously after the HLO system is started. A scheduled option ensures that the HLO system is started at system startup. There are also actions in the HLO System Monitor that allow the HLO system to be started or stopped. Starting the HLO system starts a new instance of the process manager, which in turn starts all of the other necessary processes found in the process registry. A lock is used to ensure that only one instance of the process manager is running.

The Process Manager itself also has an entry in the HLO PROCESS REGISTRY File (#779.3), and executes within the same framework as all the other processes.

Generic Framework Process

All processes (including the process manager) execute the same framework process, starting at the same entry point. The processes are started via TaskMan knowing only the process name. The process then looks up its entry in the HLO Process Registry and tailors its behavior according to the parameters defined there; the primary ones are the GET WORK function and the DO WORK function. The framework process is as follows:

- Get the process parameters from the process registry, using the process name to do a lookup.
- The process manager checks the minimum and maximum parameters of the process and determines whether or not another process needs to be started. The information on the current number of processes running is periodically updated by the CHECK PROCESS COUNTS process.

Trouble-shooting Note: If the process counts seem to be out of sync, a little time may be needed for re-synchronization.

- The process manager calls the GET WORK function of the process that is executing. If work is found, then the DO WORK function is called. This procedure is repeated until either the HLO system is shut down or GET WORK fails to find work. Once GET WORK fails to find work it will try the 'MAX TRIES FINDING WORK' to find work waiting the 'HANG TIME (seconds)' between attempts. If no work found the process exits after rescheduling itself to try later.

Appendix E - Message Header Arrays

Message Header, MSH Segment

Sequence	Header	Description/Comment
	HEADER("SEGMENT TYPE")= "MSH"	
SEQ-1	HEADER("FIELD SEPARATOR")	The 1st character field separator.
SEQ-2	HEADER("COMPONENT SEPARATOR") HEADER("SUBCOMPONENT SEPARATOR") HEADER("REPETITION SEPARATOR") HEADER("ESCAPE CHARACTER")	The four encoding characters.
SEQ-3	HEADER("SENDING APPLICATION")	
SEQ-4	HEADER("SENDING FACILITY",1) HEADER("SENDING FACILITY",2) HEADER("SENDING FACILITY",3)	First Component Second Component Third Component
SEQ-5	HEADER("RECEIVING APPLICATION")	
SEQ-6	HEADER("RECEIVING FACILITY",1) HEADER("RECEIVING FACILITY",2) HEADER("RECEIVING FACILITY",3)	First Component Second Component Third Component
SEQ-7	HEADER("DT/TM OF MESSAGE")	Converted to FileMan format.
SEQ-8	HEADER("SECURITY")	
SEQ-9	HEADER ("MESSAGE TYPE") HEADER("EVENT") HEADER("MESSAGE STRUCTURE")	First Component Second Component Third Component
SEQ-10	HEADER("MESSAGE CONTROL ID")	Message control ID.
SEQ-11	HEADER("PROCESSING ID") HEADER("PROCESSING MODE")	First Component Second Component
SEQ-12	HEADER("VERSION")	
SEQ-14	HEADER("CONTINUATION POINTER")	MESSAGE CONTROL ID of the message that this one continues.
SEQ-15	HEADER("ACCEPT ACK TYPE")	ACCEPT ACKNOWLEDGMENT TYPE, <AL or NE>
SEQ-16	HEADER("APP ACK TYPE")	APPLICATION ACKNOWLEDGMENT TYPE, <AL or NE>
SEQ-17	HEADER("COUNTRY")	COUNTRY CODE

Message Header, BHS Segment

Sequence	Header	Description/Comment
	HEADER("SEGMENT TYPE")= "BHS"	
SEQ-1	HEADER("FIELD SEPARATOR")	First character field separator
SEQ-2	HEADER("COMPONENT SEPARATOR") HEADER("SUBCOMPONENT SEPARATOR") HEADER("REPETITION SEPARATOR")	The four encoding characters,

	HEADER("ESCAPE CHARACTER")	
SEQ-3	HEADER("SENDING APPLICATION")	
SEQ-4 (See Note)	HEADER("SENDING FACILITY",1) ER("SENDING FACILITY",2) HEADER("SENDING FACILITY",3)	First Component Second Component Third Component
SEQ-5	HEADER("RECEIVING APPLICATION")	
SEQ-6 (See Note)	HEADER("RECEIVING FACILITY",1) ER("RECEIVING FACILITY",2) HEADER("RECEIVING FACILITY",3)	First Component Second Component Third Component
SEQ-7	HEADER("DT/TM OF MESSAGE")	Converted to FileMan Format.
SEQ-8	HEADER("SECURITY")	
SEQ-9	HEADER("BATCH NAME/ID/TYPE") The below fields are not defined by the standard within the BHS segment, but they are needed and are encoded in SEQ 9 by the VistA HL7 package: HEADER("PROCESSING ID") HEADER("ACCEPT ACK TYPE") HEADER("APP ACK TYPE")	
SEQ-10	HEADER("BATCH COMMENT") The VistA HL7 package, Version 1.6, Pre-HLO, requires that an application designate a batch message as being of a particular message type, event type, and version, and this information is encoded as components of SEQ10. This practice will be made obsolete with HLO, but for backwards compatibility, if SEQ10 contains components, the below three additional subscripts will be provided: HEADER("MESSAGE TYPE") HEADER("EVENT") HEADER("VERSION")	
SEQ-11	HEADER("BATCH CONTROL ID")	
SEQ-12	HEADER("REFERENCE BATCH CONTROL ID")	
Note: *The HL7 Version 2.4 Standards Manual does not document this, but components 2 & 3 were added in an erratum to be compatible with the MSH segment. It is documented in version 2.5 of the standard.*		

Appendix F – Creating TCP/IP Services for Open VMS with DSM

Note for Multi-Node Cluster Sites:

For sites configured with a multi-node cluster, more than one node may be advertised under the domain name HL7.SITENAME.MED.VA.GOV and the TCP/IP service may be running on multiple nodes.

In addition, the impersonator VMS feature allows for the possibility of all nodes in the cluster to become the surrogate. This allows for the listening process to remain uninterrupted if the TCP/IP service is enabled on all nodes in the cluster.

If this is the case for your site, be sure to enable the service on all these nodes after setting up the TCP/IP service and COM file on one of these nodes.

1. Create OpenVMS User Account

To create an OpenVMS User Account:

- **If an account already exists for HL7 1.6, use the same user account.** Review the settings for that user account to insure conformance to the screen below, then skip to Section 3 below to "Create a DCL Command Procedure."
- Determine an unused User Identification Code (UIC), typically in the same group as other DSM for OpenVMS accounts.
- Using the OpenVMS Authorize utility, add the new HLSEVEN account with the unused UIC. You must have SYSPRV to do this.
- Verify that the account settings for the new HLSEVEN account are the same as they appear in the example that follows; or, if they are different, verify that the impact of the different settings is acceptable for your system. For security, make sure that the DisCtlY, Restricted, and Captive flags are set.

There are two different ways to set up a new user account, and you are free to choose the one you prefer. The following two examples illustrate two different ways to set up an OpenVMS User account:

One way to set up an OpenVMS User account is to copy your existing XMINET (TCP/IP MailMan) account to a new account with an unused UIC. For example:

```
$ MC AUTHORIZE
UAF> COPY /ADD XMINET HLSEVEN/UIC=[51,45]/DIR=[HLSEVEN]
%UAF-I-COPMSG, user record copied
%UAF-W-DEFPWD, copied or renamed records must receive new password
%UAF-I-RDBADDMSGU, identifier HLSEVEN value [000051,000045] added to rights
database
UAF>
```

The other way to set up an Open VMS User account is to add the new HLSEVEN OpenVMS account directly. For example:

```
$ MC AUTHORIZE
 UAF> ADD HLSEVEN /UIC=[100,45]/OWNER="HLSEVEN" - (must use continuation
character "-")
_UAF> /DEVICE=USER$/DIRECTORY=[HLSEVEN] -
_UAF> /NOACCESS/NETWORK/FLAGS=(DISCTLY,RESTRICTED,NODISUSER) -
_UAF> /PRIV=(NETMBX,TMPMBX) -
_UAF> /DEF=(NETMBX,TMPMBX)/LGICMD=NL:
%UAF-I-ADDMSG, user record successfully added
%UAF-I-RDBADDMSGU, identifier HLSEVEN value [000100,000045] added to rights
data
base
UAF>

UAF> SHOW HLSEVEN

Username: HLSEVEN                          Owner:  HLSEVEN
Account:                                   UIC:    [100,45] ([HLSEVEN])
CLI:      DCL                              Tables: DCLTABLES
Default:  USER$:[HLSEVEN]
LGICMD:   NL:
Flags:  DisCtlY Restricted
Primary days:   Mon Tue Wed Thu Fri
Secondary days:                     Sat Sun
Primary   000000000011111111112222  Secondary 000000000011111111112222
Day Hours 012345678901234567890123  Day Hours 012345678901234567890123
Network:  ##### Full access ######            ##### Full access ######
Batch:    -----  No access  ------             -----  No access  ------
Local:    -----  No access  ------             -----  No access  ------
Dialup:   -----  No access  ------             -----  No access  ------
Remote:   -----  No access  ------             -----  No access  ------
Expiration:            (none)    Pwdminimum:  6   Login Fails:     0
Pwdlifetime:         90 00:00    Pwdchange:      (pre-expired)
Last Login:            (none) (interactive),            (none) (non-
interactive)
Maxjobs:          0  Fillm:       100  Bytlm:        64000
Maxacctjobs:      0  Shrfillm:      0  Pbytlm:           0
Maxdetach:        0  BIOlm:       150  JTquota:       4096
Prclm:            8  DIOlm:       150  WSdef:         2000
Prio:             4  ASTlm:       250  WSquo:         4000
Queprio:          4  TQElm:        10  WSextent:     16384
CPU:        (none)  Enqlm:      2000  Pgflquo:      50000
Authorized Privileges:
  NETMBX       TMPMBX
Default Privileges:
  NETMBX       TMPMBX
UAF>  Exit
%UAF-I-DONEMSG, system authorization file modified
%UAF-I-RDBDONEMSG, rights database modified
$
```

2. Create OpenVMS Home Directory

If a home directory already exists for HL7 1.6, then use the same home directory. Skip to Section 3 below to "Create a DCL Command Procedure."

This directory will house the DCL command procedure which is executed whenever a client connects. A log file is created for every instance of a connection for that listener. Make sure that the owner of the directory is the HLSEVEN account.

For example, to create a home directory named [HLSEVEN] with ownership of HLSEVEN:

```
$ CREATE/DIR [HLSEVEN]/OWNER=HLSEVEN
```

3. Create a DCL Command Procedure

Create a DCL command procedure (shown below) in the home directory for the HLSEVEN account and name it according to the recommended convention. Make sure the command procedure file is owned by the HLSEVEN account.

1. To create a DCL command procedure that will use port 5001, name your command procedure file as HLS5001DSM.COM.
2. Adjust the DSM command line (environment, UCI, and volume set) for your system.
3. If access control is enabled, ensure that the HLSEVEN account has access to this UCI, volume set, and routine (see "Access Control List (ACL) Issues" later in this chapter).
4. Ensure that the name of the DCL command file, as described in step 1, matches the port assignment. For example, if you changed the port number from 5001 to 6788, rename your HLS5001DSM.COM file to HLS6788DSM.COM.

WARNING: All VistA sites must use Port #5001 for the HLO Standard Listener for production accounts. For test accounts, Port #5026 must be used.

For your convenience, you can cut and paste the following DCL command procedure file into your OpenVMS HLSEVEN device and directory.

Sample DCL Command Procedure file:

```
$! HLS5001DSM.COM - for incoming tcp connect requests with port=5001 and
$! using "DSM" command line to enter the M environment
$! File name HLS5001DSM.COM is recommended to be changed to reflect the
$! change of the TCPIP port number.  For example, file name could be
$! changed to HLS6788DSM.COM if port=6788.
$!
$!this file is copied and modifed from HLSEVEN.COM
$! Revision History:
$! Patch HL*1.6*19 & HL*1.6*56--Documentation only
$! Patch HL*1.6*70--HL71_6P70.COM
$! Patch HL*1.6*84--HLS5001CACHE.COM and HLS5001DSM.COM
$!-------------------------------------------------------------
$ set noon        !Don't stop
$ set noverify    !change as needed
$! set verify     !change as needed
```

```
$ purge/keep=5 sys$login:*.log !Purge log files only
$ set proc/priv=(share)  !Required for MBX device
$ x=f$trnlnm("sys$net")  !This is our MBX device
$!
$ write sys$output "Opening "+x !This can be viewed in the log file
$! Check status of the BG device before going to either DSM or Cache'
$ cnt=0
$ CHECK:
$ stat=f$getdvi("''x'","STS")
$ if cnt .eq. 10
$ then
$ write sys$output "Could not open ''x' - exiting"
$ goto EXIT
$ else
$    if stat .ne. 16
$    then
$    cnt=cnt+1
$    write sys$output "''cnt'> ''x' not ready!"
$    wait 00:00:01 !Wait one second to assure connection
$    goto CHECK
$    else
$    write sys$output "''x' is now ready for use - entering DSM or Cache"
$!---------------------------------------------------------------
$! **Be sure the command line(s) in the COMMAND LINE SECTION
$! **below is correct for your system and if access control is
$! **enabled, that this account has access to this uci,vol & routine.
$! **An entry in file 870 for this logical link with the specified
$! **unique port number and its device type as "MS"(Multi-threaded
$! **server) must be existed.
$!
$! **Also, comment or uncomment the appropriate lines for your system.
$!
$!---------------------------------------------------------------
$! COMMAND LINE SECTION:
$! =====================
$!---------------------------------------------------------------
$! for DSM
$  dsm/env=dsmmgr/uci=vah/vol=tou VMS^HLOSRVR
$!---------------------------------------------------------------
$! for Cache
$!assign 'f$trnlnm("SYS$NET")' SYS$NET
$!csession cache "-U" "VAH" "VMS^HLOSRVR"
$!---------------------------------------------------------------
$ endif
$ exit:
$ logout/brief
```

4. Set up the TCP/IP Service

To create the TCP/IP service to listen for connections:

- Choose the OpenVMS node where you want to run the TCP/IP service listener. This is also the node whose IP address will be advertised to other systems as the location of your HL7 listener.
- Use TCP/IP port number 5001 on production systems and 5026 on test systems.

  WARNING – All VistA sites must use Port #5001 for the HLO Standard Listener for production accounts. For test accounts, Port #5026 must be used.

- Use user account HLSEVEN.
- Use DCL command file name HLS5001DSM.COM.

NOTE: Since TCP/IP Services is node specific, make sure you are on the same node that you want the listener to run on.

Ensure that your new TCP/IP service uses the recommended naming convention. For example, set up a service that will be listening on port 5001 and use a corresponding DCL command file HLS5001DSM.COM.

Set the service name as HLS5001DSM as follows:

```
$ TCPIP          (must use continuation character "-" at end of long lines)
TCPIP> SET SERVICE HLS5001DSM/USER=HLSEVEN/PROC=HLS5001DSM /PORT=5001-
_TCPIP> /PROTOCOL=TCP/REJECT=MESSAGE="All channels busy" -
_TCPIP> /LIMIT=50/FILE=USER$:[HLSEVEN]HLS5001DSM.COM
```

In this command, **/LIMIT=50/** specifies the maximum number of TCP/IP connections that can be made at any time. The limit of 50 is appropriate for most local sites, but for systems that serve as national databases the limit should be set initially to 500. The system manager is responsible for monitoring the peak number of connections made, and if the peak approaches the limit, the limit should be increased.

If you get an error because you mistyped any of the above lines or forgot to use the continuation character "-", we suggest you do the following to remove the corrupted service and repeat the above commands.

```
TCPIP> SET CONFIG ENABLE NOSERVICE HLS5001DSM
TCPIP> SET NOSERVICE HLS5001DSM
```

```
TCPIP> SHO SERVICE HLS5001DSM/FULL

Service: HLS5001DSM
                           State:     Disabled
Port:              5001    Protocol:  TCP             Address:  0.0.0.0
User_name: not defined     Process:  HLS5001DSM
```

5. Enable and Save the TCP/IP Service

Since TCP/IP Services is node specific, make sure you are on the same node on which the listener will run.

```
TCPIP> ENABLE SERVICE HLS5001DSM          (enable service immediately)
TCPIP> SET CONFIG ENABLE SERVICE HLS5001DSM  (save service for reboot)
TCPIP> SHO SERVICE/FULL HLS5001DSM

Service: HLS5001DSM
                          State:     Enabled
Port:              5001   Protocol:  TCP            Address:  0.0.0.0
Inactivity:           5   User_name: HLSEVEN        Process:  HLS5001DSM
Limit:               50   Active:      0            Peak:       0

File:          USER$:[HLSEVEN] HLS5001DSM.COM
Flags:         Listen

Socket Opts:   Rcheck Scheck
 Receive:             0     Send:               0

Log Opts:      None
 File:         not defined

Security
 Reject msg:   All channels busy

 Accept host: 0.0.0.0
 Accept netw: 0.0.0.0

TCPIP> SHO CONFIG ENABLE SERVICE

Enable service
     FTP, FTP_CLIENT, HLS5001DSM, MPI, TELNET, XMINETMM
TCPIP> EXIT
```

6. Access Control List (ACL) Issues

Some sites use DSM's ACL feature, which controls access explicitly to each OpenVMS account that needs to enter that DSM environment. If your site is using ACL, you should set up the HLSEVEN account with PROGRAMMER access, and then specify the Volume set and UCI name that the HLSEVEN user account has authorization to access. Ensure that the OpenVMS HLSEVEN account prohibits Batch, Local, Dialup, and Remote logins, allowing only Network logins.

An example of setting this level of access for an HLSEVEN account is provided below:

```
$ DSM /MAN ^ACL

Environment Access Utilities

    1.  ADD/MODIFY USER                 (ADD^ACL)
    2.  DELETE USER                     (DELETE^ACL)
    3.  MODIFY ACTIVE AUTHORIZATIONS    (^ACLSET)
    4.  PRINT AUTHORIZED USERS          (PRINT^ACL)

Select Option > 1  <Enter>  ADD/MODIFY USER

OpenVMS User Name:   >  HLSEVEN

ACCESS MODE    VOL       UCI       ROUTINE
-----------    ---       ---       -------

No access rights for this user.

Access Mode ([M]ANAGER, [P]ROGRAMMER, or [A]PPLICATION):   >  P
Volume set name:   >   VAH
UCI:   >  ROU
UCI:   >  <RET>
Volume set name:   >  <RET>
Access Mode ([M]ANAGER, [P]ROGRAMMER, or [A]PPLICATION):   >  <RET>

USER           ACCESS MODE    VOL       UCI       ROUTINE
----           -----------    ---       ---       -------

HLSEVEN        PROGRAMMER     ROU       VAH

OK to file?   <Y>   <RET>

OpenVMS User Name:   >    <RET>

OK to activate changes now?   <Y>    <RET>

Creating access authorization file:  USER$:[DSMMGR]DSM$ACCESS.DAT.
```

7. Control the Number of Log Files Created by TCP/IP Services

The HLS5001DSM TCP/IP service automatically creates log files (TCP/IP services does this and it cannot be prevented) in the HLSEVEN directory named HLS5001DSM.LOG;xxx where 'xxx' is a file version number. New versions of this file will be created until that file version number reaches the maximum number of 32767. In order to minimize the number of log files created, you may want to initially rename this log file to the highest version number with the command:

```
$ RENAME USER$:[HLSEVEN]HLS5001DSM.LOG; USER$:[HLSEVEN]HLS5001DSM.LOG;32767
```

Alternatively, you can set a limit on the number of versions of the log file that can concurrently exist in the HLSEVEN directory:

```
$ SET FILE /VERSION_LIMIT=10 USER$:[HLSEVEN]HLS5001DSM.LOG;
```

NOTE: This cannot be done until the first log file has actually been created.

You probably should not limit the number of versions of the log file until you know that your HLS5001DSM service is working correctly; keeping the log files can help when diagnosing problems with the service/account.

8. Other TCP/IP Service Commands

WARNING: If HLO is stopped or disabled for two hours or more, the VMS Multi-Listener service should be disabled and then re-enabled before restarting HLO.

The definition of a link is required for the multi-threaded listener for Open VMS systems. This link never needs to be started or stopped through the VistA HL 1.6 or HLO options. Instead, it is normally started and stopped via TCP/IP services. For example:

```
TCPIP> DISABLE SERVICE HLS5001DSM        (Stop TCP/IP service)
```

```
TCPIP> ENABLE SERVICE HLS5001DSM         (Start TCP/IP service)
```

Any questions about configuring TCP/IP Service for OpenVMS should be directed to EVS for assistance.

Appendix G – HLO Integration Agreements (DBIAs)

This appendix lists the DBIAs created for the HLO APIs. Please refer to FORUM for the most up to date listing of active DBIAs.

INTEGRATION REFERENCE INQUIRY #4716
NAME: HLO BUILD MESSAGE APIS
CUSTODIAL PACKAGE: HEALTH LEVEL SEVEN
SUBSCRIBING PACKAGE:
USAGE: Supported ENTERED: JUL 6,2005
STATUS: EXPIRES:
DURATION: Till Otherwise Agr VERSION:
DESCRIPTION: TYPE: Routine

These APIs are used to build HLO messages.

ROUTINE: HLOAPI
COMPONENT: $$NEWMSG(.PARMS,.HLMSTATE,.ERROR)
This API is to be used by applications that need to send an
HL7 message via HLO. It starts the message building process.

VARIABLES: Input PARMS
("COUNTRY") - A three-character country code (optional).
("CONTINUATION POINTER") - Indicates a fragmented message. ("EVENT") – A three-character event type (required).
("FIELD SEPARATOR") - Field separator (optional, defaults to "|").
("ENCODING CHARACTERS") - Four HL7 encoding characters (optional, defaults to "(^~\&").
("MESSAGE STRUCTURE") - MSH 9, component 3 - a code from the standard HL7table ((optional).
("MESSAGE TYPE")- A three-character message type (required).
("PROCESSING MODE") - MSH 11, component 2 - A one character code (optional).
("VERSION") - The HL7 Version ID, for example, "2.4" (optional, defaults to 2.4).
VARIABLES: Output HLMSTATE
Used by the HL7 package to track the progress of the message.
VARIABLES: Output ERROR
Optional. Returns an error message on failure.
VARIABLES: Output $$NEWMSG
Returns 1 on success, 0 on failure.
COMPONENT: $$NEWBATCH(.PARMS,.HLMSTATE,.ERROR)
This API is to be used by applications that need to send a batch of HL7 message via HLO. It starts the batch building process.
VARIABLES: Input PARMS
("COUNTRY") - A three-character country code (optional).
("FIELD SEPARATOR") - Field separator (optional, defaults to "|").|
("ENCODING CHARACTERS") - Four HL7 encoding characters

(optional, defaults to "^~\&"). ("VERSION") - HL7 Version ID, for example, "2.4" (optional, defaults to 2.4).

VARIABLES: Output HLMSTATE
Used by the HL7 package to track the progress of the message.
VARIABLES: Output ERROR
Returns an error message on failure.
VARIABLES: Output $$NEWBATCH
Function returns 1 on success, 0 on failure.
COMPONENT: $$ADDMSG(.HLMSTATE,.PARMS,.ERROR)
Used by applications to add a message to a batch that is in the process of being built.
VARIABLES: Output $$ADDMSG
The function returns 1 on success, 0 on failure.
VARIABLES: Both HLMSTATE
An array used by the HL7 package to track the progress of the message as it is being built.
VARIABLES: Input PARMS
("EVENT") - A three-character event type (required).
("MESSAGE TYPE") - A three-character message type (required).
VARIABLES: Output ERROR
Optional - returns an error message on failure.
COMPONENT: SET(.SEG,VALUE,FIELD,COMP,SUBCOMP,REP)
Used to set a value into a segment that is in the process of being built.
VARIABLES: Both SEG
The array where the segment is being built.
VARIABLES: Input VALUE
The individual value to be set into the segment.
VARIABLES: Input FIELD
The sequence # of the field (optional, defaults to 0) Note: FIELD=0 is used to denote the segment type.
VARIABLES: Input COMP
The # of the component (optional, defaults to 1).
VARIABLES: Input SUBCOMP
The # of the subcomponent (optional, defaults to 1).
VARIABLES: Input REP
The occurrence# (optional, defaults to 1) For a non-repeating field, the occurrence # need not be provided, because it would be 1.
COMPONENT: $$ADDSEG(.HLMSTATE,.SEG,.ERROR)
Used to add a segment that has just been built to a message that is still in the process of being built.
VARIABLES: Output $$ADDSEG
Function returns 1 on success, 0 on failure.
VARIABLES: Both HLMSTATE
Used by the HL7 package to track the progress of the message as it is being built.
VARIABLES: Input SEG
Required. Contains the segment built by calls to SET prior to calling $ADDSEG.

Note#1: The message control segments, including the MSH and BHS segments, are added automatically.
Note#2: The 0th field must be a 3 character segment type. Note#3: SEG is killed upon

successfully adding the segment.

VARIABLES: Output ERROR
Returns an error message on failure.
COMPONENT: MOVEMSG(.HLMSTATE,.ARY)
If a message was built using any other method than the HLO APIs and resides in an array, it will be moved into HLO. This API allows segment builders that were created prior to HLO to be used within HLO.
VARIABLES: Both HLMSTATE
Created by calling $$NEWMSG^HLOAPI or $$NEWBATCH^HLOAPI. It tracks the progress of the message as its is being built.
VARIABLES: Input ARY
The name of the local or global variable where the message was built. It is accessed via indirection to move the message into HLO.
KEYWORDS:

INTEGRATION REFERENCE INQUIRY #4717
NAME: HLO SEND MESSAGE APIS
CUSTODIAL PACKAGE: HEALTH LEVEL SEVEN
SUBSCRIBING PACKAGE:
USAGE: Supported ENTERED: JUL 6,2005
STATUS: Pending EXPIRES:
DURATION: Till Otherwise Agr VERSION:
DESCRIPTION: TYPE: Routine
These APIs provide the ability to address a message that has already been built and put it on an out-going queue for transmission.

ROUTINE: HLOAPI1
COMPONENT: $$SENDONE(.HLMSTATE,.PARMS,.WHOTO,.ERROR)
Sends the message to a single recipient. The recipient is identified in the message header by the Receiving Facility and the Receiving Application. The message may optionally be routed through an interface engine.
VARIABLES: Output $$SENDONE
Function call returns the IEN of the message in file 778 on success, 0 on failure.
VARIABLES: Both HLMSTATE
Used by the HL7 package to track the progress of the message.
VARIABLES: Input PARMS
("APP ACK RESPONSE")=<tag^routine> to call in response to app ack (i.e., not received). (Optional. This parameter is ignored if the ACK TO parameter is present.)
("APP ACK TYPE") =<AL,NE> (Optional, defaults to NE).
("ACCEPT ACK RESPONSE")=<tag^routine> to call in response to a commit ack(optional).

("ACCEPT ACK TYPE") =<AL,NE> (Optional, defaults to AL).
("FAILURE RESPONSE") - <tag>^<routine> (Optional) The sending application routine to execute when the transmission of the message fails, i.e., the message can not be sent or no commit ack is received.
("QUEUE") - Optional. An application can name its own private queue - just a string under 20 characters, it should be namespaced.
("SECURITY")=Security information to include in the header segment, SEQ 8 (Optional).
("SENDING APPLICATION")=name of sending application (required, 60 max-length).

VARIABLES: Input WHOTO

Required. Specifies a single recipient.
("RECEIVING APPLICATION") - String, 60 char max, required.

One of the following four parameters is required to identify the Receiving Facility:

("FACILITY LINK IEN") - IEN of the logical link.
("FACILITY LINK NAME") - Name of the logical link.
("INSTITUTION IEN") - Pointer to the INSTITUTION file.
("STATION NUMBER") - Station # with suffix.

One of the following two parameters MAY be provided, optionally, to identify the interface engine to route the message through:

("IE LINK IEN") - Pointer to a logical link for the interface engine.
("IE LINK NAME") - Name of the logical link for the interface engine.

VARIABLES: Output ERROR

Returns an message on error.

COMPONENT: $$SENDMANY(.HLMSTATE,.PARMS,.WHOTO)

Sends the message that has already been built to a list of recipients.

VARIABLES: Output $$SENDMANY

Returns 1 on success, 0 on failure.

VARIABLES: Both HLMSTATE

This array is used to track the progress of the message.

VARIABLES: Input PARMS

("APP ACK RESPONSE")=<tag^routine> to call in response to app ack (i.e., not received). (Optional. This parameter is ignored if the ACK TO parameter is present.)
("APP ACK TYPE") =<AL,NE> (Optional, defaults to NE).
("ACCEPT ACK RESPONSE")=<tag^routine> to call in response to a commit ack (optional).
("ACCEPT ACK TYPE") =<AL,NE> (Optional, defaults to AL).
("FAILURE RESPONSE") - <tag>^<routine> (Optional) The sending application routine to execute when the transmission of the message fails, i.e., the message can not be sent or no commit ack is received.
("QUEUE") - Optional. An application can name its own private queue - just a string under 20 characters, it should be namespaced.
("SECURITY")=Security information to include in the header segment, SEQ 8 (Optional).

("SENDING APPLICATION")=name of sending application (required, 60 max-length).
VARIABLES: Both WHOTO
For Input: Specifies a list of recipients. Each recipient should be listed individually in array
WHOTO(i), where i=a recipient. For each recipient the same subscripts may be defined as in the $$SENDONE API. For example:

WHOTO(1,"LINK NAME")="VAALB"
WHOTO(1,"RECEIVING APPLICATION")="MPI"
WHOTO(2,"STATION NUMBER")=500
WHOTO(2,"RECEIVING APPLICATION")="MPI"

For Output: Returns the status of each message to be sent in the format:
(<i>,"QUEUED") - 1 if queued to be sent, 0 otherwise.
(<i>,"IEN") -IEN, file 778.
(<i>,"ERROR") - Error message if an error was encountered (status=0),not defined otherwise.

COMPONENT: $$SENDSUB(.HLMSTATE,.PARMS,.MESSAGES)
Send Messages to Subscription Registry Subscribers
VARIABLES: Output $$SENDSUB
Function call returns 1 if a message is queued to
be sent to each intended recipient, 0 otherwise.

VARIABLES: Both HLMSTATE
Used by HLO internally to track the progress of the message.
VARIABLES: Input PARMS
Required. Same as $$SENDMANY^HLOAPI1, with one additional subscript:
("SUBSCRIPTION IEN") -The IEN of an entry in the HLO SUBSCRIPTION REGISTRY file (#779.4), defining the intended recipients of this message

VARIABLES: Output MESSAGES
Returns the status of each message to be sent in this format, where the sub-IEN is the IEN of the recipient in the RECIPIENTS sub-file of the HLO SUBSCRIPTION REGISTRY file (#779.4).
(<subien>,"QUEUED") - 1 if queued to be sent, 0 otherwise.
(<subien>,"IEN") - IEN, HLO MESSAGES file (#778).
(<subien>,"ERROR") - Error message if an error was encountered (status=0),;not defined otherwise.

KEYWORDS:

INTEGRATION REFERENCE INQUIRY #4718
NAME: HLO PARSING APIS
CUSTODIAL PACKAGE: HEALTH LEVEL SEVEN
SUBSCRIBING PACKAGE:
USAGE: Supported ENTERED: JUL 6,2005
STATUS: Pending EXPIRES:

DURATION: Till Otherwise Agr VERSION:
DESCRIPTION: TYPE: Routine
These APIs are to be used by applications that receive HL7 messages via HLO. They provide the means of stepping through batches of messages, the message segments, and fetching data values from within segments.

ROUTINE: HLOPRS
COMPONENT: $$STARTMSG(.HLMSTATE,IEN,.HDR)
This function begins the parsing of the message, parsing the header and returning the individual values in the array HDR().
VARIABLES: Output $$STARTMSG
Returns 1 on success, 0 on failure.
VARIABLES: Input IEN
The internal entry number of the message in file 778.
VARIABLES: Output HLMSTATE
Required. This array is used by the HL7 package to track the progress of parsing the message. The application MUST NOT touch it.
VARIABLES: Output HDR Optional. This array contains the results of parsing the message header.
COMPONENT: $$NEXTSEG(.HLMSTATE,.SEG)
Advances parsing to the next segment within the message.
VARIABLES: Output $$NEXTSEG
Function returns 1 on success, 0 if there are no more segments in this message. For batch messages, a return value of 0 does not preclude the possibility that there are additional individual messages within the batch.
VARIABLES: Both HLMSTATE
HLMSTATE is an array used internally by HLO to track the progress of parsing.
VARIABLES: Output SEG
The segment is returned in this array.
COMPONENT: $$NEXTMSG(.HLMSTATE,.MSH)
Advances to the next message within the batch, with the MSH segment returned.
VARIABLES: Output $$NEXTMSG
Function returns 1 on success, 0 if there are no more messages.
VARIABLES: Both HLMSTATE
This array is used by HLO to track the current position in the message.
VARIABLES: Output MSH
Returns the MSH segment, parsed into its individual values.
COMPONENT: $$GET(.SEG,FIELD,COMP,SUBCOMP,REP)
This function gets a specified value from a segment that was parsed by $$NEXTSEG^HLOPRS. The FIELD,COMP,SUBCOMP,REP parameters are optional - if not specified, they default to 1.
Example: $$GET^HLOPRS(.SEG,1) will return the value of the first field, first component, first subcomponent, in the first occurrence of field #1.
VARIABLES: Output $$GET
This function returns a specified value from a segment.
VARIABLES: Input SEG
This is the array where the parsed segment was placed by $$NEXTSEG^HLOPRS.
VARIABLES: Input FIELD
The sequence # of the field (optional, defaults to 1). If 0 (zero) is specified, then

the function returns the segment type.
VARIABLES: Input COMP
The number of the component (optional, defaults to 1).
VARIABLES: Input SUBCOMP
The number of the subcomponent (optional, defaults to 1).
VARIABLES: Input REP
The occurrence number (optional, defaults to 1) For a non-repeating field, the occurrence number need not be provided, because it would be 1.
KEYWORDS:

INTEGRATION REFERENCE INQUIRY #4722
NAME: HLO APPLICATION ACKNOWLEDGEMENT APIS
CUSTODIAL PACKAGE: HEALTH LEVEL SEVEN
SUBSCRIBING PACKAGE:
USAGE: Supported ENTERED: JUL 7,2005
STATUS: EXPIRES:
DURATION: Till Otherwise Agr VERSION:
DESCRIPTION: TYPE: Routine
These APIs are used by applications to return application acknowledgments to messages received via HLO.

ROUTINE: HLOAPI2
COMPONENT: $$ACK(.HLMSTATE,.PARMS,.ACK,.ERROR)
This API initiates (but doesn't complete) an application acknowledgment. This API should NOT be called for batch messages, use $$BATCHACK^HLOAPI3 instead.
VARIABLES: Output $$ACK
Function call returns 1 on success, 0 on failure.
VARIABLES: Both HLMSTATE
Obtained by calling $$STARTMSG^HLOPRS when parsing the original message. It is used internally by HLO.
VARIABLES: Input PARMS
Optional. These subscripts may be defined:
("ACK CODE") - MSA1 contains AA, AE, or AR.
("ERROR MESSAGE") - MSA3, should be used only if AE or AR.
("ACCEPT ACK RESPONSE") - Optional. The <tag^routine> to call in response to a commit ack.
("ACCEPT ACK TYPE") - {AL,NE} (Optional, defaults to AL).
("CONTINUATION POINTER") -Indicates a fragmented messages.
("COUNTRY") - The three-character country code (optional).
("EVENT") - The three-character event type (optional, defaults to the event code of the original message).
("ENCODING CHARACTERS") - The four HL7 encoding characters (optional, defaults to "^~\&".
("FAILURE RESPONSE") - Optional. The <tag>^<routine> that the sending application routine should execute if the transmission of the message fails, i.e., the message can not be sent or a requested commit ack is not received.
("FIELD SEPARATOR") - Field separator (optional, defaults to "|").|
("MESSAGE TYPE") - If not defined, ACK is used.
("MESSAGE STRUCTURE CODE") - Optional. ("QUEUE")- Optional. An

application can name its own private queue (a string under 20 characters, it should be namespaced). The default is the name of the queue of the original message
("SECURITY") - Optional. Security information to include in the header segment, SEQ 8.
("VERSION") - The HL7 Version ID (optional, defaults to 2.4).

VARIABLES: Output ACK
The acknowledgement message being built.
VARIABLES: Output ERROR
On failure, an error message is returned.
COMPONENT: $$SENDACK(.ACK,.ERROR)
Sends the acknowledgment message that was begun by a call to $$ACK^HLAPI2 or a batch of acknowledgement messages that was begun by a call to $$$$BATCHACK^HLOAPI3.
VARIABLES: Output $$SENDACK
Function call returns 1 on success, 0 on failure.
VARIABLES: Input ACK
An array that contains the acknowledgment message that was built by calling the other APIs.
VARIABLES: Output ERROR
If the function fails, an error message is returned.
KEYWORDS:

****

INTEGRATION REFERENCE INQUIRY #4723
NAME: HLO APPLICATION ACKNOWLEDGEMENT APIS (CONTINUED)
CUSTODIAL PACKAGE: HEALTH LEVEL SEVEN
SUBSCRIBING PACKAGE:
USAGE: Supported ENTERED: JUL 7,2005
STATUS: EXPIRES:
DURATION: Till Otherwise Agr VERSION:
DESCRIPTION: TYPE: Routine
These APIs are part of the set of tools that an application uses to return an application acknowledgement to a message that was received via HLO. See integration agreement # 4722 for the related APIs.

ROUTINE: HLOAPI3
COMPONENT: $$BATCHACK(.HLMSTATE,PARMS,.ACK,.ERROR)
This routine is used to initiate a batch message containing individual application acknowledgments to a batch of messages that was received via HLO. Individual acks are placed in this batch by calling $$ADDACK^HLOAPI3, then the batch of acks is actually sent by calling $$SENDACK^HLOAPI2.
VARIABLES: Output $$BATCHACK
The function returns 1 on success, 0 on failure.
VARIABLES: Both HLMSTATE
The HLMSTATE array is used internally by HLO to track the processing of the message. It is created by the application's earlier call to

$$STARTMSG^HLOPRS when parsing the original message.

VARIABLES: Input PARMS
These subscripts may be defined:

("ACCEPT ACK RESPONSE") - <tag^routine> to call in response to a commit ack (optional).
("ACCEPT ACK TYPE") - <AL,NE> (Optional, defaults to AL).
("COUNTRY") - A three-character country code from the HL7 standard table (optional).
("ENCODING CHARACTERS") - The four HL7 encoding characters; optional, defaults to "^~\&".
("FAILURE RESPONSE") - Optional. The <tag>^<routine> that the sending application routine should execute if the transmission of the message fails, i.e., the message can not be sent or a requested commit ack is not received.
("FIELD SEPARATOR") - Field separator; optional, defaults to "|".|
("QUEUE") - Optional. An application can name a private queue (a string under 20 characters, it should be namespaced). The default is the name of the queue of the original message.
("SECURITY") - Security information to include in the header segment, SEQ 8 (optional).
("VERSION") - The HL7 Version ID (optional, defaults to 2.4)

VARIABLES: Output ACK
The acknowledgement being built.

VARIABLES: Output ERROR
On failure, the function returns an error message.

COMPONENT: $$ADDACK(.ACK,.PARMS,.ERROR)
This API adds an application acknowledgement to a batch acknowledgement message that was started by calling $$BATCHACK^HLOAPI3.

VARIABLES: Output $$ADDACK
The function returns 1 on success, 0 on failure.

VARIABLES: Both ACK
The batch of acknowledgements that is being built.

VARIABLES: Input PARMS
These subscripts may be defined:

("ACK CODE") - Required. MSA1 contains AA, AE, or AR.
("ERROR MESSAGE") - Optional. MSA3 should be used only if AE or AR.
("EVENT") - A three-character event type (optional, defaults to the event code of the original message).
("MESSAGE CONTROL ID") - Required. The message control ID of the original individual message within the batch that is being acknowledged.
("MESSAGE STRUCTURE CODE") - Optional.
("MESSAGE TYPE") - Optional, defaults to ACK.
("SECURITY") - Optional. Security information to include in the header segment SEQ.

VARIABLES:

VARIABLES: Output ERROR
On failure, the function also returns an error message.
KEYWORDS:

INTEGRATION REFERENCE INQUIRY #4724
NAME: HLO MISCELANEOUS APIS
CUSTODIAL PACKAGE: HEALTH LEVEL SEVEN
SUBSCRIBING PACKAGE:
USAGE: Supported ENTERED: JUL 7,2005
STATUS: EXPIRES:
DURATION: Till Otherwise Agr VERSION:
DESCRIPTION: TYPE: Routine
These are APIs provided by HLO that don't fit into any of the other categories.

ROUTINE: HLOAPI3
COMPONENT: $$RESEND(MSGIEN,.ERROR)
This routine re-transmits a message. It does this by making a copy of the message, reusing all the original parameters. Then the message is placed in the same outgoing queue.
VARIABLES: Output $$RESEND
The function returns 1 on success, 0 on failure.
VARIABLES: Input MSGIEN
The IEN of the message that is to be sent from HLO MESSAGES file(#778).
VARIABLES: Output ERROR
On failure, the function also returns an error message.
COMPONENT: $$REPROC(MSGIEN,.ERROR)
This routine reprocesses a message by placing it on the appropriate incoming queue.
VARIABLES: Output $$REPROC
This function returns 1 on success, 0 on failure.
VARIABLES: Input MSGIEN
The IEN of the message that is to be sent from HLO MESSAGES file(#778).
VARIABLES: Output ERROR
On failure, this function also returns an error message.
COMPONENT: $$SETPURGE(MSGIEN,TIME)
This API can be used to reset the scheduled purge date/time.
VARIABLES: Output $$SETPURGE
This function returns 1 on success, 0 on failure.
VARIABLES: Input MSGIEN
The IEN of the message that is to be reprocessed from HLO MESSAGES file (#778).
VARIABLES: Input TIME
Optional, date/time to which to set the purge time. If not defined, defaults to NOW.
KEYWORDS:

INTEGRATION REFERENCE INQUIRY #4725
NAME: HLO SUBSCRIPTION REGISTRY APIS
CUSTODIAL PACKAGE: HEALTH LEVEL SEVEN
SUBSCRIBING PACKAGE:
USAGE: Supported ENTERED: JUL 7,2005
STATUS: EXPIRES:
DURATION: Till Otherwise Agr VERSION:
DESCRIPTION: TYPE: Routine
These APIs allow applications to create and manage entries in the HLO Subscription Registry. Each entry is basically a list of recipients that can be used and reused to address HL7 messages. Its similar to a mailing list. See also IA# 4726.

ROUTINE: HLOASUB
COMPONENT: $$CREATE(OWNER,DESCRIPTION,.ERROR)
This API is used to create a new entry in the HLO Subscription Registry.
VARIABLES: Output $$CREATE
Function call returns the new IEN in the HLO SUBSCRIPTION REGISTRY file (#779.4) if successful, 0 if error.
VARIABLES: Input OWNER
The name of the owning application. It should be pre-fixed with the owning application's namespace.
VARIABLES: Input DESCRIPTION
Optional. A short description of the subscription registry entry.
VARIABLES: Output ERROR
The function also returns an error message if it fails.
COMPONENT: $$ADD(IEN,.WHO,.ERROR)
Add a new recipient to an existing subscription list.
VARIABLES: Output $$ADD
The function returns the subien of the recipient from the RECIPIENTS multiple, 0 on failure.
VARIABLES: Input IEN
The IEN of the entry in the HLO SUBSCRIPTION REGISTRY file (#779.4).
VARIABLES: Input WHO
An array containing the information for a single new recipient to be added to the list. These subscripts are allowed:
("RECEIVING APPLICATION") - String, 60 char max, required.
One of the following four parameters must be provided to identify the Receiving Facility:
("FACILITY LINK IEN") - IEN of the logical link.
("FACILITY LINK NAME") - Name of the logical link.
("INSTITUTION IEN") - Pointer to the INSTITUTION file.
("STATION NUMBER") - Station # with suffix. ONE of the following two parameters MAY be provided - optionally - to identify the interface engine to route the message through:
("IE LINK IEN") - Pointer to a logical link for the interface engine.
("IE LINK NAME") - Name of the logical link for the interface engine.
VARIABLES: Output ERROR
On failure, one of these messages will be returned:
"SUBSCRIPITON REGISTRY ENTRY NOT FOUND"

"RECEIVING FACILTY LOGICAL LINK NOT FOUND" "RECEIVING APPLICATION NOT FOUND"
"INTERFACE ENGINE LOGICAL LINK PROVIDED BUT NOT FOUND"
"FAILED TO ACTIVATE SUBSCRIBER"

COMPONENT: $$END(IEN,.WHO)
To terminate a recipient from the subscriber list. The recipient isn't deleted, but the DATE/TIME TERMINATED field is entered with the current date/time.
VARIABLES: Output $$END
The function returns 1 on success, 0 on failure.
VARIABLES: Input IEN
The IEN of the HLO SUBSCRIPTION REGISTRY file (#779.4) entry.
VARIABLES: Input WHO
If WHO("SUBIEN") is defined, then it should be the IEN of the sub-record to be terminated. Otherwise, set the set the parameter as per $$ADD^HLOASUB
COMPONENT: $$ONLIST(IEN,LINKIEN,APPNAME,FAC1,FAC2,FAC3)
This function is used to check whether or not a potential recipient is already on a particular subscription list.
VARIABLES: Output $$ONLIST
Function call returns the IEN of the recipient from the RECIPIENTS multiple, 0 on failure.
VARIABLES: Input IEN
The IEN of the HLO SUBSCRIPTION REGISTRY file (#779.4) entry.
VARIABLES: Input LINKIEN
IEN of the logical link.
VARIABLES: Input APPNAME
The name of the receiving application.
VARIABLES: Input FAC1
Component 1 of the receiving facility.
VARIABLES: Input FAC2
Component 2 of the Receiving Facility.
VARIABLES: Input FAC3
Component 3 of the Receiving Facility.
COMPONENT: $$NEXT(IEN,.RECIP)
This API is used to loop through a subscription list. It ignores recipients that have been terminated from the list.
VARIABLES: Output $$NEXT
Function call returns the IEN of the recipient from the RECIPIENTS multiple, 0 on failure.
VARIABLES: Input IEN
The IEN of the HLO SUBSCRIPTION REGISTRY file (#779.4) entry.
VARIABLES: Both RECIP
Input: If NULL, it gets the first recipient on the list, else it uses the value of RECIP("SUBIEN") to find the next recipient.

Output: RECIP - Required. Returns the next recipient on the list. These subscripts are returned:

("LINK IEN") ("LINK NAME") ("RECEIVING APPLICATION") ("RECEIVING FACILITY",1) -

Component 1 ("RECEIVING FACILITY",2) - Component 2 ("RECEIVING FACILITY",3) - Component 3 ("SUBIEN") - The IEN in the multiple, used to find the next on the list.

KEYWORDS:

****

INTEGRATION REFERENCE INQUIRY #4727
NAME: HLO CONVERSION APIS
CUSTODIAL PACKAGE: HEALTH LEVEL SEVEN
SUBSCRIBING PACKAGE:
USAGE: Supported ENTERED: JUL 7,2005
STATUS: EXPIRES:
DURATION: Till Otherwise Agr VERSION:
DESCRIPTION: TYPE: Routine
These utilities provide help to applications that were developed before HLO convert to HLO. See also IA# 4728 and IA#4731.

ROUTINE: HLOCNRT
COMPONENT: $$EN(HLOPRTCL,ARYTYP,.HLP)
Takes a current HL 1.6 message that follows the standard HL 1.6 methodology, converts it to use the HLO engine, and places the message into the HLO message queue. A function call to EN^HLOCNRT replaces the HL 1.6 call to GENERATE^HLMA.
VARIABLES: Output $$EN
The function returns a string that is one to three piece variable consisting of message id^error code^error description. Only the message id will be returned if there is no error.
VARIABLES: Input HLOPRTCL
Event Protocol IEN
VARIABLES: Input ARYTYP
Array Type ("GM" is the standard usage, used for a global array containing a single message.
VARIABLES: Input HLP
Additional MSH parameters. For example: - HLP("SECURITY") - HLP("CONTPTR") - HLP("QUEUE")

KEYWORDS:

****

INTEGRATION REFERENCE INQUIRY #4278
NAME: HLO CONVERSION APIS (2)
CUSTODIAL PACKAGE: HEALTH LEVEL SEVEN
SUBSCRIBING PACKAGE:

USAGE: Supported ENTERED: JUL 7,2005
STATUS: EXPIRES:
DURATION: Till Otherwise Agr VERSION:
DESCRIPTION: TYPE: Routine
These utilities provide help to applications that were developed before HLO convert to HLO. See also IA# 4727 and IA#4731.

ROUTINE: HLOCVU
COMPONENT: APAR(HLOEID,.APPARMS,.WHO,.WHOTO)
Designed to retrieve pre-HLO HL 1.6 parameters from the existing HL 1.6 protocol and translate to HLO format.
VARIABLES: Input HLOEID
Event Protocol IEN.
VARIABLES: Output APPARMS
Array containing HLO message parameters.

Specific translation from HL 1.6 parameters to HLO is as follows:

HL 1.6 APPLICATION PROTOCOL => HLO APPARMS ARRAY PARAMETERS COUNTRY CODE => APPARMS("COUNTRY") APPLICATION ACK TYPE => APPARMS("APP ACK TYPE") EVENT TYPE => APPARMS("EVENT") SENDING APPLICATION => APPARMS("SENDING APPLICATION") TRANSACTION MESSAGE TYPE => APPARMS("MESSAGE TYPE") VERSION ID => APPARMS("VERSION") HL7 FIELD SEPARATOR => APPARMS("FIELD SEPARATOR") HL7 ENCODING CHARACTERS => APPARMS("ENCODING CHARACTERS")

HL 1.6 Passed Parameters => HLO APPARMS ARRAY PARAMETERS HLP("SECURITY") => APPARMS("SECURITY") HLP("CONTPTR") => APPARMS("CONTINUATION POINTER") HLP("QUEUE")* => APPARMS("QUEUE")

*NOTE: HLP("QUEUE") is not actually a current HL 1.6 parameter but can be added to the HLP array to allow a converted application to define HLO private queues.

For Sending Messages To One Application

–

RECEIVING APPLICATION =>
WHO("RECEIVING APPLICATION") LOGICAL LINK
=> WHO("FACILITY
LINK NAME")

For Sending Messages To Multiple Applications
(where "n" is a numeric index (0,1,2,))
_ RECEIVING APPLICATION =>
WHOTO(n,"RECEIVING APPLICATION") LOGICAL LINK
=> WHOTO(n,"FACILITY
LINK NAME")

VARIABLES: Output WHO
For single HLO message recipients, receiving
application parameters.
VARIABLES: Output WHOTO
For multiple HLO message recipients, receiving
application parameters.
KEYWORDS:

****

INTEGRATION REFERENCE INQUIRY #4730
NAME: HLO QUEUE MANAGEMENT APIS
CUSTODIAL PACKAGE: HEALTH LEVEL SEVEN
SUBSCRIBING PACKAGE:
USAGE: Supported ENTERED: JUL 8,2005
STATUS: EXPIRES:
DURATION: Till Otherwise Agr VERSION:
DESCRIPTION: TYPE: Routine
These APIs are for applications to use in KIDS distributions of messaging applications. They allow the application to turn on and off individual queues during the installation of a patch.

ROUTINE: HLOQUE
COMPONENT: STOPQUE(DIR,QUEUE)
Routine designed to set a "Stop" flag on a named queue. Flag
to set is ^HLTMP("STOPPED QUEUES",DIR,QUEUE).

VARIABLES: Input DIR
Direction of queue. Values are "IN" or "OUT".
VARIABLES: Input QUEUE
The name of the queue to be stopped.
COMPONENT: STARTQUE(DIR,QUEUE)
Routine designed to remove a "Stop" flag on a named queue.
Flag to remove is ^HLTMP("STOPPED QUEUES",DIR,QUEUE).
VARIABLES: Input DIR
Direction of queue. Values are "IN" or "OUT".
VARIABLES: Input QUEUE
The name of the queue to be started.

COMPONENT: $$STOPPED(DIR,QUEUE)
Function designed to check the status of a queue by determining if a "Stop" flag has been set on a named queue. Flag to check is ^HLTMP("STOPPED QUEUES",DIR,QUEUE).
VARIABLES: Input DIR
Direction of queue. Values are "IN" or "OUT".
VARIABLES: Input QUEUE
The name of the queue to be checked.
VARIABLES: Output $$STOPPED
The function returns 1 if the named queue is stopped, 0 otherwise.
KEYWORDS:

****

INTEGRATION REFERENCE INQUIRY #4731
NAME: HLO CONVERSOIN APIS (3)
CUSTODIAL PACKAGE: HEALTH LEVEL SEVEN
SUBSCRIBING PACKAGE:
USAGE: Supported ENTERED: JUL 8,2005
STATUS: EXPIRES:
DURATION: Till Otherwise Agr VERSION:
DESCRIPTION: TYPE: Routine
These utilities provide help to applications that were developed before HLO convert to HLO. See also IA# 4727 and IA#4728.

ROUTINE: HLOMSG
COMPONENT: $$HLNEXT(.HLMSTATE,.SEG)
This API is NOT to be used in the development of a new messaging application. It is provided for messaging applications that were developed prior to HLO where stepping through a message was accomplished by executing HLNEXT.

The new function $$HLNEXT is used to step through the segments of a message stored in the new HLO data structures. However, for batch messages, it does not transverse from one message to the next as executing HLNEXT does.
VARIABLES: Both HLMSTATE
This array is used by the HL7 package to track the current position in the message. The application MUST NOT touch it.
VARIABLES: Output SEG
The segment is returned in this array.
VARIABLES: Output $$HLNEXT
The function returns 1 on success, 0 if there are no more segments in this message. For batch messages, a return value of 0 does not preclude the possibility that there are additional individual messages within the batch.
KEYWORDS:

INTEGRATION REFERENCE INQUIRY #4852

NAME: HLO DATA TYPE PARSERS
CUSTODIAL PACKAGE: HEALTH LEVEL SEVEN
SUBSCRIBING PACKAGE:
USAGE: Supported ENTERED: APR 26,2006
STATUS: Pending EXPIRES:
DURATION: Till Otherwise Agr VERSION:
DESCRIPTION: TYPE: Routine
This provides specialized APIs for parsing HL7 data types from a segment. It applies only to HL7 messages received via the HLO software that was released in patch HL*1.6*126.

ROUTINE: HLOPRS2
COMPONENT: GETTS(.SEG,.VALUE,FIELD,COMP,REP)
Gets a segment value that is a time stamp in HL7 format and converts it to FileMan format. IF the data type value includes the time zone then the time is converted to local time. The degree of precision is optionally returned. IF the component is specified, then the component is parsed for data type rather than at the higher field level.

VARIABLES: Input SEG
(required, pass by reference) The array returned by a call to $$NEXTSEG^HLOPRS.

VARIABLES: Input FIELD
(required) The sequence # of the field.

VARIABLES: Input COMP
(optional) If specified, the data type is parsed as a component value.

VARIABLES: Input REP
The occurrence # (optional, defaults to 1). For a non-repeating field, this parameter is not necessary.

VARIABLES: Output VALUE
(required, pass-by-reference IF subscripts are used) The date/time in FileMan format. The PRECISION subscript is optional, if provided the time stamp's precision will be determined.
"PRECISION" - (optional) Expected values are:
"S" - second
"M" - minute
"H" - hour
"D" - day
"L" - month

"Y" - year
"" - precision not specified
Note: FM does not allow greater precision than seconds, so greater
precision will be rounded down to the second.

COMPONENT: GETDT(.SEG,.VALUE,FIELD,COMP,REP)
Gets a segment value that is a date in HL7 format and converts it to FileMan format. The degree of precision is optionally returned. IF the component is specified, then the component is parsed for data type rather than at the higher field level.

VARIABLES: Input SEG
(required, pass by reference) The array returned by a call to $$NEXTSEG^HLOPRS.

VARIABLES: Input FIELD
(required) The sequence # of the field.

VARIABLES: Input COMP
(optional) If specified, the data type is parsed as a component value.

VARIABLES: Input REP
(optional, defaults to 1) The occurrence#. For a non-repeating fields, this parameter is not necessary.

VARIABLES: Output VALUE
(required, pass-by-reference if the precisoin is needed) The date/time in FileMan format. The "PRECISION" subscript is also returned:
"PRECISION" Expected values are:
"S" - second (not valid for DT)
"M" - minute (not valid for DT)
"H" - hour (not valid for DT)
"D" - day
"L" - month
"Y" - year
"" - precision not specified

COMPONENT: GETCE(.SEG,.VALUE,FIELD,COMP,REP)
Gets an CE data type(Coded Element, HL7 Section Reference 2.9.8)from the
specified field. IF the component is specified, then the component is
parsed for data type rather than at the higher field level.

VARIABLES: Input SEG
(required, pass-by-reference) The array returned by a call to NEXTSEG^HLOPRS.

VARIABLES: Input COMP
(optional) If specified, the data type is parsed as a component value.

VARIABLES: Input FIELD
(required) The sequence # of the field.

VARIABLES: Input REP
The occurrence # (optional, defaults to 1). For a non-repeating fields, this parameter is not necessary.

VARIABLES: Output VALUE
(required, pass-by-reference) These subscripts are returned:
"ID" - the identifier
"TEXT" -
"SYSTEM" - name of the code system
"ALTERNATE ID" - alternate identifier
"ALTERNATE TEXT"
"ALTERNATE SYSTEM" - name of the alternate coding system

COMPONENT: GETHD(.SEG,.VALUE,FIELD,COMP,REP)
Gets an HD data type (Hierarchic Designator, HL7 Section Reference
2.9.21) from the specified field. IF the component is specified, then
the component is parsed for data type rather than at the higher field
level.

VARIABLES: Input SEG
(required, pass-by-reference) The array returned by a call to NEXTSEG^HLOPRS.

VARIABLES: Input FIELD
(required) The sequence # of the field.

VARIABLES: Input COMP
(optional) If specified, the data type is parsed as a component value.

VARIABLES: Input REP
(optional, defaults to 1) The occurrence #. For a non-repeating fields, this parameter is not necessary.

VARIABLES: Output VALUE
(required, pass-by-reference) These subscripts are returned:
"NAMESPACE ID"
"UNIVERSAL ID"
"UNIVERSAL ID TYPE"

COMPONENT: GETCNE(.SEG,.VALUE,FIELD,COMP,REP)
Gets an CNE data type (Coded With No Exceptions, HL7 Section
Reference 2.9.8) from the specified field. IF the component is
specified, then the component is parsed for data type rather than at the
higher field level.

VARIABLES: Input SEG
(required, pass-by-reference) The array returned by a call to NEXTSEG^HLOPRS.

VARIABLES: Input FIELD
(required) The sequence # of the field.

VARIABLES: Input COMP
(optional) If specified, the data type is parsed as a component value.

VARIABLES: Input REP
(optional, defaults to 1) The occurrence #. For a non-repeating fields, this parameter is not necessary.

VARIABLES: Output VALUE
(required, pass-by-reference) These subscripts are returned:
"ID" - the identifier
"TEXT" -
"SYSTEM" - name of the code system
"ALTERNATE ID" - alternate identifier
"ALTERNATE TEXT"
"ALTERNATE SYSTEM" - name of the alternate coding system
"SYSTEM VERSION" - version ID of the

coding system
"ALTERNATE SYSTEM VERSION" - version ID of the alternate
coding system
"ORIGINAL TEXT"

COMPONENT: GETCWE(.SEG,.VALUE,FIELD,COMP,REP)
Gets an CWE data type (Coded With Exceptions, HL7 Section
Reference 2.9.11) from the specified field. . IF the component is
specified, then the component is parsed for data type rather than at the
higher field level.

VARIABLES: Input SEG
(required, pass-by-reference) The array returned by a call to NEXTSEG^HLOPRS.

VARIABLES: Input FIELD
(required) The sequence # of the field.

VARIABLES: Input COMP
(optional) If specified, the data type is parsed as a component value.

VARIABLES: Input REP
(optional, defaults to 1) The occurrence #. For a non-repeating fields, this parameter is not necessary.

VARIABLES: Output VALUE
VALUE (required, pass-by-reference) These subscripts are returned:
"ID" - the identifier
"TEXT" -
"SYSTEM" - name of the code system
"ALTERNATE ID" - alternate identifier
"ALTERNATE TEXT"
"ALTERNATE SYSTEM" - name of the alternate coding system
"SYSTEM VERSION" - version ID of the coding system
"ALTERNATE SYSTEM VERSION" - version ID of the alternate
coding system
"ORIGINAL TEXT"

COMPONENT: GETAD(.SEG,.VALUE,FIELD,COMP,REP)

Gets an AD data type (Address, HL7 Section Reference 2.9.1) from the
specified field. It can also be used to get the 1st 8 components of the
XAD (Extended Address) data type. IF the component is specified, then
the component is parsed for the address rather than at the higher field
level.

VARIABLES: Input SEG
(required, pass-by-reference) The array returned by a call to NEXTSEG^HLOPRS.

VARIABLES: Input FIELD
(required) The sequence # of the field.

VARIABLES: Input COMP
(optional) If specified, the data type is parsed as a component value.

VARIABLES: Input REP
(optional, defaults to 1) The occurrence #. For a non-repeating fields, this parameter is not necessary.

VARIABLES: Output VALUE
(required, pass-by-reference) These subscripts are returned:
"STREET1" -street address
"STREET2" - other designation
"CITY"
"STATE" - state or province
"ZIP" - zip or postal code
"COUNTRY"
"TYPE" - address type
"OTHER" - other geographic designation

KEYWORDS:

****

INTEGRATION REFERENCE INQUIRY #4853

NAME: HLO BUILDING MESSAGES WITH DATA TYPES
CUSTODIAL PACKAGE: HEALTH LEVEL SEVEN
SUBSCRIBING PACKAGE:
USAGE: Supported ENTERED: APR 28,2006
STATUS: EXPIRES:

DURATION: Till Otherwise Agr VERSION:
DESCRIPTION: TYPE: Routine
This provides specialized APIs for buiding messages with HL7 data types.
It applies only to HL7 messages received via the HLO software that was
released in patch HL*1.6*126.

ROUTINE: HLOAPI4
COMPONENT: SETTS(.SEG,.VALUE,FIELD,COMP,REP)
Sets a value that is a time stamp in FM format into the segment in HL7
format. The degree of precision may be optionally specified. The
inserted value will include the timezone if the input included the time.
IF the component is specified, then the data type is 'demoted' to a
component, and its components are 'demoted' to subcomponents.

VARIABLES: Both SEG
(required, pass by reference) The segment that is being built.

VARIABLES: Input FIELD
(required) The sequence # of the field.

VARIABLES: Input COMP
(optional) If specified, the data type is 'demoted' to a component value.

VARIABLES: Input REP
(optional, defaults to 1) The occurrence #. For a non-repeating fields, this parameter is not necessary.

VARIABLES: Input VALUE
(required, pass-by-reference to also pass the the "PRECISION" subscript) The date/time in FileMan format. You can optionally specify that the value is to be rounded down to a particular precision by specifying this subscript:
"PRECISION" - Allowed values are:
"S" - second
"M" - minute
"H" - hour
"D" - day
"L" - month
"Y" - year
"" - precision not specified

COMPONENT: SETDT(.SEG,.VALUE,FIELD,COMP,REP)
Sets a value that is a date in FM format into the segment in HL7 format.
The degree of precision may be optionally specified. IF the component
is specified, then the data type is 'demoted' to a component, and its
components are 'demoted' to subcomponents.

VARIABLES: Both SEG
(required, pass by reference) The segment that is being built.

VARIABLES: Input VALUE
(required) The date to be set into the segment. Optionally, you may specify that the value should be rounded down to a particular precision by specifying this subscript:
"PRECISION" (If needed, VALUE must be passed by reference.)
Allowed values are:

"D" - day (default value)
"L" - month
"Y" - year

VARIABLES: Input FIELD
(required) The sequence # of the field.

VARIABLES: Input COMP
(optional) If specified, the data type is 'demoted' to a component value.

VARIABLES: Input REP
(optional, defaults to 1) The occurrence #. For a non-repeating fields, this parameter is not necessary.

COMPONENT: SETCE(.SEG,.VALUE,FIELD,COMP,REP)
Sets a value that is an HL7 Coded Element data type (HL7 Section
Reference 2.9.3) into the segment in the specified field. IF the
component is specified, then the data type is 'demoted' to a component,
and its components are 'demoted' to subcomponents.

VARIABLES: Both SEG

(required, pass-by-reference) The segment that is being built.

VARIABLES: Input VALUE
(required, pass-by-reference) These subscripts may be passed:
"ID" - the identifier
"TEXT" -
"SYSTEM" - name of the code system
"ALTERNATE ID" - alternate identifier
"ALTERNATE TEXT"
"ALTERNATE SYSTEM" - name of the alternate coding system

VARIABLES: Input FIELD
(required) The sequence # of the field.

VARIABLES: Input COMP
(optional) If specified, the data type is 'demoted' to a component value.

VARIABLES: Input REP
(optional, defaults to 1) The occurrence #. For a non-repeating fields, this parameter is not necessary.

COMPONENT: SETHD(.SEG,.VALUE,FIELD,COMP,REP)
Sets a value that is an HL7 Hierarchic Designator data type (HL7 Section
Reference 2.9.21) into the segment in the specified field. IF the
component is specified, then the data type is 'demoted' to a component,
and its components are 'demoted' to subcomponents.

VARIABLES: Both SEG
(required, pass-by-reference) The array where the segment is being built.

VARIABLES: Input VALUE
(required, pass-by-reference) These subscripts may be passed:
"NAMESPACE ID"
"UNIVERSAL ID"
"UNIVERSAL ID TYPE"

VARIABLES: Input FIELD
(required) The sequence # of the field.

VARIABLES: Input COMP
(optional) If specified, the data type is

'demoted' to a component value.

VARIABLES: Input REP
(optional, defaults to 1) The occurrence #. For a non-repeating fields, this parameter is not necessary.

COMPONENT: SETCNE(.SEG,.VALUE,FIELD,COMP,REP)
Sets a value that is an HL7 Coded With No Exceptions data type (HL7
Section Reference 2.9.8) into the segment in the specified field. IF the
component is specified, then the data type is 'demoted' to a component,
and its components are 'demoted' to subcomponents.

VARIABLES: Both SEG
(required, pass-by-reference) The array where the segment is being built.

VARIABLES: Input VALUE
(required, pass-by-reference) These subscripts may be passed:
"ID" - the identifier
"TEXT" -
"SYSTEM" - name of the code system
"ALTERNATE ID" - alternate identifier
"ALTERNATE TEXT"
"ALTERNATE SYSTEM" - name of the alternate coding system
"SYSTEM VERSION" - version ID of the coding system
"ALTERNATE SYSTEM VERSION" - version ID of the alternate coding
system
"ORIGINAL TEXT"

VARIABLES: Input FIELD
(required) The sequence # of the field.

VARIABLES: Input COMP
(optional) If specified, the data type is 'demoted' to a component value.

VARIABLES: Input REP
(optional, defaults to 1) The occurrence #. For a non-repeating fields, this parameter is not necessary.

COMPONENT: SETCWE(.SEG,.VALUE,FIELD,COMP,REP)
Sets a value that is an HL7 Coded With Exceptions data type (HL7 Section
Reference 2.9.11) into the segment in the specified field. IF the
component is specified, then the data type is 'demoted' to a component,
and its components are 'demoted' to subcomponents.

VARIABLES: Both SEG
(required, pass-by-reference) The array where the segment is being built.

VARIABLES: Input VALUE
(required, pass-by-reference) These subscripts may be passed:
"ID" - the identifier
"TEXT" -
"SYSTEM" - name of the code system
"ALTERNATE ID" - alternate identifier
"ALTERNATE TEXT"
"ALTERNATE SYSTEM" - name of the alternate coding system
"SYSTEM VERSION" - version ID of the coding system
"ALTERNATE SYSTEM VERSION" - version ID of the alternate coding
system
"ORIGINAL TEXT"

VARIABLES: Input FIELD
(required) The sequence # of the field.

VARIABLES: Input COMP
(optional) If specified, the data type is 'demoted' to a component value.

VARIABLES: Input REP
(optional, defaults to 1) The occurrence #. For a non-repeating fields, this parameter is not necessary.

COMPONENT: SETAD(.SEG,.VALUE,FIELD,COMP,REP)
Sets an AD data type (Address, HL7 Section Reference 2.9.1) into the
segment in the specified field. It can also be used to set the 1st 8
components of the XAD (Extended Address) data type. IF the component is

specified, then the data type is 'demoted' to a component, and its
components are 'demoted' to subcomponents.

VARIABLES: Both SEG
(required, pass-by-reference) The array where the segment is being built.

VARIABLES: Input VALUE
(required, pass-by-reference) These subscripts may be passed:
"STREET1" -street address
"STREET2" - other designation
"CITY"
"STATE" - state or province
"ZIP" - zip or postal code
"COUNTRY"
"TYPE" - address type
"OTHER" - other geographic designation

VARIABLES: Input FIELD
(required) the sequence # of the field.

VARIABLES: Input COMP
(optional) If specified, the data type is 'demoted' to a component value.

VARIABLES: Input REP
(optional, defaults to 1) The occurrence #. For a non-repeating fields, this parameter is not necessary.

KEYWORDS:

****

Appendix H – HLO Error Messages

This is a sample list of possible error messages that could be returned by HLO API function calls.

Error Message	Error Description
$$SAVE^HLOF777 FAILED!	All attempts to create a message header in HLO MESSAGE BODY File (#777) failed.
$$SAVE^HLOF778 FAILED!	All attempts to create a message body in HLO MESSAGES File (#778) failed.
BATCH ACKNOWLEDGEMENTS MUST USE $$BATCHACK^HLOAPI3	For a batch message, a single message ack entry point was used instead of batch message ack entry point.
DOMAIN NOT FOUND	Domain could not be found in DOMAIN File (#4.2).
EVENT TYPE INVALID	The Event Type is not three characters in length, as required by HL7.
INVALID ACCEPT ACKNOWLEDGEMENT TYPE	Accept Ack Type of "NE" (never) or "AL" (always) is not used.
INVALID ACK CODE	Ack Code of "AA," "AE," or "AR" was not used.
INVALID APPLICATION ACKNOWLEDGEMENT TYPE	Application Ack Type of "NE" (never) or "AL" (always) was not used.
INVALID COUNTRY CODE	Country Code used is not three characters in length.
INVALID ENCODING CHARACTERS	Encoding Characters used are not four characters long, as required.
INVALID EVENT CODE	Event Code is not three characters long, as required.
INVALID FIELD SEPARATOR	Field Separator is not one character long, as required.
INVALID MESSAGE TYPE	Message Type Code is not three characters in length, as required.
INVALID PROCESSING MODE	Processing Mode value of 'P' (Production) or 'T' (Test) is not used.
INVALID SEGMENT TYPE	Segment name is not three characters long, as required.
MESSAGE CONTROL ID MUST EXIST TO RETURN AN APPLICATION ACK	Message Control ID was not found in message parameters. Ack was not sent.
MESSAGE CONTROL ID MUST EXIST TO RETURN AN APPLICATION ACK	Message Control ID was not found in header. Ack was not sent.
MESSAGE NOT YET CREATED	SEND API call was invoked before message was created.
MESSAGE TYPE INVALID	Message Type code is not three characters in length, as required.
NO MESSAGES IN BATCH, SEGMENTS NOT ALLOWED	Attempt was made to add a segment to a batch message before creating the message.
ORIGINAL MESSAGE TO ACKNOWLEDGE IS NOT IDENTIFIED	Unable to send Ack. IEN of original message is unknown.
QUEUE PARAMETER IS MAX 20 LENGTH	Queue names cannot be longer than 20 characters.
RECEIVING APPLICATION NOT DEFINED	Receiving application not defined in the HLO APPLICATION REGISTRY File (#779.2).
SENDING APPLICATION IS REQUIRED	Sending Application parameter was not supplied.

SENDING APPLICATION NOT FOUND IN THE APPLICATION REGISTRY	Sending Application was not found in the HLO APPLICATION REGISTRY File (#779.2).
SUBSCRITPION REGISTRY IEN NOT PROVIDED	Subscription IEN was not supplied to SENDSUB API call.
TRANSMISSION LINK FOR APPLICATION ACK CANNOT BE DETERMINED	All methods used for determining application Ack link were not successful.
TRANSMISSION LINK FOR APPLICATION ACK CANNOT BE DETERMINED	All methods used for determining application Ack link were not successful.
VERSION > 20 CHARACTERS	Supplied version is over 20 characters long.

Appendix I – Daily Oversight and Troubleshooting

1. Daily Oversight

1.1 HLO System Monitor

The IRM staff should check the operational status of HLO several times daily using the HLO System Monitor. Please refer to Section 5.2 of this document for complete instructions. The Brief System Status screen provides all the information necessary. The following should be verified:

- SYSTEM STATUS is RUNNING
- PROCESS MANAGER is RUNNING
- STANDARD LISTENER is OPERATIONAL
- TASKMAN is RUNNING
- DOWN LINKS is 0 (zero)
- MESSAGES PENDING TRANSMISSION is not unusually large
- STOPPED OUTGOING QUEUES is blank
- MESSAGES PENDING ON APPLICATION is not unusually large
- STOPPED INCOMING QUEUES is blank
- IF Interface Engine is used, then INTERFACE ENGINE is operational

NOTE: At the time this was written, the status of the Interface Engine will show as 'NOT OPERATIONAL'. This will continue until a new entry in the HL LOGICAL LINK named 'VA-VIE' is distributed in a future patch.

If any of these conditions indicate that there may be a problem, then investigate further by following instructions found in Section 5.2 of this document.

1.2 HLO Message Viewer

The IRM should run three reports on a daily basis and review them for signs of problems. Refer to Section 5.3 of this document for further details on these reports. Potential problems should be fully investigated and either corrected or referred to the appropriate subject experts. For example, if a message is not processed due to an application error, the problem might need to be referred to the appropriate package expert.

The reports are:

- System Errors – these are messages that were not passed to the Receiving Application for any reason. An example might be a message whose header lacked the Receiving Application field.
- Application Errors – these are messages that were passed to the Receiving Application, but the application was unable to fully process the message. Usually errors of this type require subject matter expertise to resolve.
- Transmission Failures – these are messages that could not be transmitted after three days, despite multiple attempts. Errors of this type occur when the listener process at the receiving facility is running but for some reason it is unable to respond properly.

2. Troubleshooting

Problem #1:
After installing patch HL*1.6*126 and establishing the HLO standard listener, the existing HL7 1.6 listener stopped working.

Solution:
Were any of the field values in the HL LOGICAL LINK File (#870) used by the existing listener modified or deleted? Some fields in the HL LOGICAL LINK File (#870) are not used by HLO, but listeners running under the existing HL7 1.6 still need them. If any field was altered where the set up instructions did not explicitly state to do so, restore the old value.

Problem #2:
The HLO System Monitor shows the status of the Standard Listener as NOT OPERATIONAL even though the listener set up was completed.

Solution:
1. Double-check that the VMS TCP/IP service was enabled.
2. Double-check that the entry in the HL LOGICAL LINK File (#870) for the listener has these fields completed:
    a. TCP/IP PORT (OPTIMIZED) field (#400.08)
    b. TCP/IP ADDRESS field (#400.01)
3. Double-check that the HLO SYSTEM PARAMETERS File (#779.1) IEN=1 (there should be exactly one entry) for the HLO STANDARD LISTENER field (#.1) points to the correct listener entry in the HL LOGICAL LINK File (#870). (see 2. above)
4. Double-check the HLO PROCESS REGISTRY File (#779.3) record for the site's listener process. Most sites will be using the VMS TCP Listener process. The record should include the following:
    a. Field ACTIVE (#.02) is set to YES
    b. Field DEDICATED LINK (#.14) points to the correct listener entry in the HL LOGICAL LINK File (#870). (see 2 above)

Problem #3:
The HLO Message Monitor shows "SE" System Errors for a newly installed application.

Solution:
Check the HL Logical Link record for the application and verify the following:
1. The DNS DOMAIN field (#.08) is set to the correct domain.
2. The TCP/IP ADDRESS field (#400.01) is set to the correct address.
3. The TCP/IP PORT (OPTIMIZED) field (#400.08) is set to the correct port number (production is 5001 and test is 5026).

Problem #4:
A DOWN LINK is found when checking the HLO System Monitor.

Solution:
Check the following:

1. The link entry parameters in the HL LOGICAL LINK File (#870) are correct.
2. The listener process at the receiving facility is running.
3. Network connections between the sending and receiving facility are operating normally.

For any questions or problems related to installation issues or other errors not described in this document, please contact EVS and request technical support.

Glossary

Accept Acknowledgement
From the HL7 standard: "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."[3]

Acknowledgement
A software handshake method used by HL7. When a system receives a message, it may send back an "accept acknowledgement" message to confirm the data was received. It may also send back an "application acknowledgement" message to confirm if the data received was appropriate.

Application
In the terminology of VistA HLO, an application sends and/or receives messages through the interface between two systems. The application consists of a sending and receiving entry in the HLO APPLICATION REGISTRY File (#779.2).

Application Acknowledgement
From the HL7 standard: "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."[4]

Commit Acknowledgement
See Accept Acknowledgement

Enterprise Application Integration (EAI)
Combines the technologies and processes that enable COTS and/or in-house-developed software applications to exchange information in the formats and contexts that each understands.

Event
A healthcare event, such as an admission, discharge or bed transfer, inter-ward transfer, transfer to a new treating specialty, etc., that causes a need for information to flow between two or more applications.

Event Driver Protocol
For VistA HL 1.6, 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 VistA side of the interface, or on the other side of the interface. Protocols are used in HL 1.6 but not in HLO. HLO conversion APIs use the HL 1.6 protocol to retrieve application parameters.

Event Type
Trigger event code (one component of a message header's Message Type field) defined by *HL7 table 0003 - Event type*.

Field
A specific unit of data within an HL7 *segment*. Each field is defined by the following set of characteristics: Position in the Segment, Maximum Length, Data Type, Optionality, Repetition, Table Assignment (optional), ID Number, and Name.

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.

HL7
Health Level Seven. An ANSI standard that specifies how information exchange should occur between healthcare applications in a healthcare environment. It permits data exchange between heterogeneous applications and systems through a messaging architecture.

[3] Health Level Seven, *Health Level Seven, Version 2.3.1*, ©1999, p. E-1.
[4] Health Level Seven, *Health Level Seven, Version 2.3.1*, ©1999, p. E-2.

HLO	"HL7 Optimized" Enhanced HL7 engine for VistA HL7, Version 1.6.
HLO Standard Listener	This is the HL Logical Link entry that defines the primary listener for facility. Port number must be 5001 on production systems and 5026 on a sites main test system.
Interface	The negotiated HL7 specification between two or more systems, defining the supported transactions.
Link	An entry in the HL Logical Link File (#870) that links VistA HLO to a particular destination. The destination is a TCP/IP address and port or a DNS DOMAIN.
Logical Link	See Link.
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."[5]
Message Type	Message type code (one component of a message header's Message Type field) defined by *HL7 table 0076 - Message type*.
Protocol	For each message type (e.g., A01) sent or received by any given interface (e.g., Radiology), one PROTOCOL File (#101) entry on your system is used to represent the sending application (event driver protocol) and one PROTOCOL File (#101) entry on your system is used to represent the receiving application (subscriber protocol). Protocols are used in HL 1.6 but not in HLO. HLO conversion APIs use the HL 1.6 protocol to retrieve application parameters.
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."[6]
Subscriber	An application that subscribes to a particular event point, registering its interest so it can receive unsolicited updates. This is used in HL 1.6 but not in HLO. HLO conversion APIs use the HL 1.6 protocol to retrieve application parameters.
Subscriber Protocol	For VistA 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 VistA side of the interface, or by the other side of the interface. Protocols are used in HL 1.6 but not in HLO. HLO conversion APIs use the HL 1.6 protocol to retrieve application parameters.
Queue	Incoming and outgoing queues are used by VistA HLO to manage message delivery.
Trigger Event	An event that initiates an action. In the case of HL7, a specific event within an application might trigger the generation of a message.

[5] Health Level Seven, *Health Level Seven, Version 2.3.1*, ©1999, p. E-18.
[6] Health Level Seven, *Health Level Seven, Version 2.3.1*, ©1999, p. E-28.

Index

U

V