V*ISTA* HEALTH LEVEL SEVEN

(HL7)

SITE MANAGER & DEVELOPER MANUAL

Version 1.6*161

December 1999

Department of Veterans Affairs
VHA CIO Technical Services

P R E F A C E

Acknowledgments

The **V***IST***A** HL7 team gratefully acknowledges the following individuals for their invaluable contributions to **V***IST***A** HL7 patches HL*1.6*56 and/or HL*1.6*57:

REDACTED

We gratefully acknowledge the following participants in the VHA Messaging System JAD (Joint Application Development) Workshop for their contributions to **V***IST***A** HL7:

REDACTED

Reader's Comments

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

REDACTED

C O N T E N T S

I N T R O D U C T I O N

Welcome to the ***VISTA** HL7 Site Manager & Developer Manual*. The goal of this manual is to provide **V*IST*A** developers and site managers with all of the information you need to build **V*IST*A** HL7 interfaces and manage the **V*IST*A** HL7 software package.

For Post-HL*1.6*57 V*IST*A HL7 Environments Only

This single, combined manual supersedes the two original user manuals released with version 1.6 of the package, the *DHCP HL7 Developer Manual* and *DHCP HL7 User Manual*. Many patches have been released since that time, and the HL7 standard itself has evolved over the past few years. The new, combined ***VISTA** HL7 Site Manager & Developer Manual* brings the package's documentation up-to-date with those changes. It should be used for **V*IST*A** HL7 systems that are at a patch level of HL*1.6*57 and above only. The manuals it replaces are archived on the **V*IST*A** HL7 Package Homepage (see below).

Related Resources

V*IST*A HL7 Package Homepage

Provides the latest information on the **V*IST*A** HL7 package, including the full documentation set, latest news, and links to other sites:

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

V*IST*A Data Systems and Integration (VDSI) HL7 Homepage

The web site of the **V*IST*A** HL7 Messaging Administrator, this provides information on HL7, including the HL7 standard and the **V*IST*A** HL7 Specification Repository:

http://vista.med.va.gov/vdsi/

HL7 Standard Documentation

The best source of information about the *Health Level Seven* standard is the standard itself. It is available at the **V*IST*A** HL7 package's homepage and the VDSI homepage, both listed above.

For information about the HL7 standards body itself, please see their web site, at:

http://www.hl7.org/

Documentation Conventions

Programmer Entry Point Descriptions

For each M programmer entry point, a function prototype is provided as follows:

Usage	`D ENTRY^ROUTINE(param1, .param2, [param3])`

Conventions used for function prototypes are as follows:

- Entry point *parameters* (as opposed to input variables) are listed in lowercase. This is to convey that the listed parameter name is merely a placeholder; M allows you to pass a variable of any name as the parameter and even a string literal (if the parameter is not being passed by reference).
- A leading period indicates the parameter should be passed by reference.
- Rectangular brackets [ ] around a parameter indicate that passing the parameter is optional.

Screen Dialog

The 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>. So, for example, the following indicates that the user should enter two question marks followed by <RET> when prompted:

```
Select Primary Menu option: ?? <RET>
```

Documentation Icons

These icons placed in the left-hand margin highlight passages in the documentation as follows:

This illuminates a key point.

Warning!

Make note of this.

C H A P T E R

1. Getting Started with HL7

1.1 What is HL7?

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

From the HL7 standard itself:

> Health Level Seven (HL7) is an application protocol for electronic data exchange in health care environments. The HL7 protocol is a collection of standard formats which specify the implementation of interfaces between computer applications from different vendors. This communication protocol allows healthcare institutions to exchange key sets of data amount different application systems. Flexibility is built into the protocol to allow compatibility for specialized data sets that have facility-specific needs.[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 in question.

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

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

[1] Health Level Seven, *Health Level Seven, Version 2.3.1* copyright 1999, p. E-15.

1.2 What is an HL7 Message?

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 a field containing the message type and (HL7 versions 2.2 and above) 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/Query" 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 BHS or FHS segments).

In addition to the field separator character, four other special characters, called *encoding characters*, are used as delimiters. Encoding characters are also defined in the MSH 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 delimiters for which there are encoding characters are:

- *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 separates escape sequences from 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|^~\&|INFOLIO TEST|REDACTED|IB TEST|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 first line of the message is the message header (MSH) segment
- The message type (from the MSH segment) is Observation Result/Unsolicited (ORU)
- 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

1.3 When are HL7 Messages Sent?

1.3.1 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, the trigger event **a patient is admitted** may cause the need for data about that patient to be sent to a number of other systems. The trigger event, **an observation (e.g., a CBC result) for a patient is available,** may cause the need for that 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 **V*IST*A**, an ADT A04 message is generated and delivered to all subscriber applications interested in patient registrations.

Unsolicited updates are also used to roll up data from **V*IST*A** systems to Austin and other centralized databases.

1.3.2 Acknowledgments

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

- An *accept acknowledgment* (also called a commit acknowledgment) confirms that the receiving system has received the message that was sent.
- An *application acknowledgment* confirms that the application that received the message found the data in the message to be appropriate.

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

1.3.3 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.

---

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

1.4 What is the Role of the V*IST*A HL7 Package?

V*IST*A HL7 is an *implementation* of the HL7 standard. It is an M-based software product that assists M-based **V*IST*A** applications by providing a means for those applications to send and receive HL7 messages.

V*IST*A HL7 does not provide tools to map **V*IST*A** 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 **V*IST*A** applications can use **V*IST*A** 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 **V*IST*A** M-based applications running on a different VA facility's systems. As such, **V*IST*A** HL7 acts as an Enterprise Application Integration (EAI) solution for **V*IST*A** applications.

1.4.1 Evolution of V*IST*A HL7

The first released version of **V*IST*A** HL7, V. 1.5, supported simple point-to-point HL7 transactions between **V*IST*A** and a local COTS system using Hybrid Lower Layer Protocol (HLLP), and to other VA facilities using VA MailMan. The initial release of V. 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 V. 1.6, released through patches, including more complex message routing (dynamic addressing), and messaging using Minimal Lower Layer Protocol (MLLP) over TCP.

Since the early 1990's, 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 **V*IST*A**'s Radiology, Pharmacy, Lab, PIMS and 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 **V*IST*A** application now generates a corresponding HL7 message, which **V*IST*A** HL7 distributes to all interested subscribers.

1.4.2 HL7 Standard Support

V*IST*A HL7 supports message transactions for each of the following versions of the HL7 standard:

- 2.1 (HL7 standard publication date: 6/90)
- 2.2 (HL7 standard publication date: 12/94)
- 2.3 (HL7 standard publication date: 4/97)
- 2.3.1 (HL7 standard publication date: 5/99)

1.5 What's New in V*IST*A HL7

1.5.1 Major Functionality Changes since V. 1.6 Release

Patch #	Release	Major Functionality Change(s)
HL*1.6*10	8/30/96	Enhance purge functionality in message purging option.
HL*1.6*13	11/7/96	Add ability to purge 'Awaiting Acknowledgment' status messages.
HL*1.6*30	7/15/97	Improve performance by optimizing to reduce disk hits.
HL*1.6*36	6/8/98	Introduce new fields to allow a message to be flagged such that it will not be purged. This is so that applications can flag and process exceptions without the message in question being purged.
		Introduce new Exception Processing entry points to flag and un-flag messages' "Don't Purge" status, and to reprocess messages: $$DONTPURG^HLUTIL, $$TOPURG^HLUTIL, $$SETPURG^HLUTIL and $$REPROC^HLUTIL.
HL*1.6*38	6/8/98	Revise the $$M10^HLFNC and $$M11^HLFNC check digit functions to comply with the HL7 standard. The predecessor functions are moved to $$OLDM10^HLFNC and $$OLDM11^HLFNC.
HL*1.6*42	8/27/98	Add support to calculate check digits for alphanumeric (as well as numeric) IDs in $$M10^HLFNC and $$M11^HLFNC.
HL*1.6*14	10/20/98	Introduce dynamic addressing of messages.
		Introduce a subscription registry.
		Provide APIs to map links to institutions.
		Provide new option to shut down all links at once.
		Introduce performance and locking improvements.
HL*1.6*40	11/18/98	Provide more information in the Systems Link Monitor.
		Send general messaging alerts and notifications to a local mail group.
		Add new site parameters for Current Domain, Current Institution, type of account (Production or Test), and Mail Group for Alerts.
		New entry point (GETAPP^HLCS2) to retrieve application-specific mail group for alerts and notifications about a particular transaction.
HL*1.6*19	2/18/99	Introduce TCP/IP as a new transport layer for **V*IST*A** HL7 (message delivery over TCP/IP was not supported in the original V. 1.6 release of **V*IST*A** HL7). Also, provide a new viewer for the transmission log (messages transmitted over TCP links only), and an Exception Processing API (again for messages transmitted over TCP links only). Optimize message delivery over TCP links (over 50% reduction in disk writes).

Patch #	Release	Major Functionality Change(s)
HL*1.6*43	4/26/99	Provide support for CIRN to connect to Master Patient Index.
		Add support for auditing based on subscriber protocol (to prevent bypassing auditing when DUZ is undefined).
HL*1.6*47	7/9/99	Move purging parameters from the Task Parameters field in the Option Scheduling file into **V*IST*A** HL7's site parameters.
		Enhance "View Transmission Log" option's querying capabilities.
HL*1.6*50	7/9/99	Enhancements to the View Transmission Log option.
HL*1.6*49	9/30/99	Provide a new link manager for TCP links so that each TCP link no longer needs a continuously running process (150 links to other VA hospitals would otherwise require 150 running processes.)
HL*1.6*39	10/13/99	Provide pre-defined TCP links for **V*IST*A** facilities. This patch is being continually updated as a test patch during the deployment of CIRN-PD.
HL*1.6*56	12/99	Provide a complete refresh of the User and Technical manuals.
HL*1.6*57	12/99	Please see "Package Overhaul Patch (HL*1.6*57)" below.

1.5.2 Package Overhaul (Patch HL*1.6*57)

- **Exportable Interface Definitions.** Interfaces are now exportable to all **V*IST*A** systems with little or no modification at the site.
- **Streamlined Interface Development.** Interfaces are more intuitive to set up. Definition is now consistent, regardless of the **V*IST*A** system role as sender or receiver, and of the link type used (Mailman, TCP, HLLP, X3.28 or same-system). This consistency reduces the number of protocol definitions required for simple interfaces (usually, just two now).
- **Menu Redesign.** The menu tree is streamlined and redesigned, based on field and Joint Application Development (JAD) group feedback.
- **Full HL7 V. 2.1 Compatibility.** HL7 1.6 is now fully backwards compatible to HL7 Standard, version 2.1. In particular, event types for version 2.1 messages are no longer required. All HL7 V. 2.1 transactions currently interfaced using **V*IST*A** HL7 V. 1.5 can now be converted to **V*IST*A** HL7 V. 1.6, and should be upgraded as soon as possible.
- **Logical Link File Consolidation.** The formerly split File #870 (HL Logical Link) and File #869.2 (HL Lower Level Protocol Parameter) have been merged into a single file (File #870).
- **Response Message Type/Event Type Can Be Different from Original Message.** Message type and Event type can now be different from the initial message's message type/event type pair. Transaction Message type and Event type are no longer required on the subscriber protocol for successful message delivery. These fields are now used only to define the message type and event type of the subscriber response.
- **Report Option Fixed.** The **V*IST*A** HL7 "Report" option has been fixed to properly display non-TCP status codes and errors. It is renamed "Link Error/Status Report (non-TCP)."

- **Same-System Batch Message Delivery.** It is now possible to exchange batch messages between applications running on the same system.
- **Message Header Security Enhancements.** It is now possible to enable header-based sending facility and receiving facility message validation. When a **V*IST*A** HL7 facility receives a message, if that facility's subscriber protocol is set to "Receiving Facility Required?", the incoming message header must have a receiving facility, and it must match the domain and institution of the receiving facility. This guards against processing messages intended for another facility. In addition, if the receiving facility's subscriber protocol is set to "Sending Facility Required", the incoming message header must have a sending facility, and it must match a facility associated with an active Link on the receiving system. This helps prevent messages being processed from, and replies sent to, unauthorized systems.
- **Addressing Deferred Acknowledgments.** The enhancements to improve message header security also provide a means for addressing deferred acknowledgments. The validated domain information is now used as the return path for responses, in place of what is defined in the Logical Link field of the Protocol file. This allows more flexible addressing without requiring a separate protocol for every possible incoming message source. In the future, it may be possible to query a DNS for current IP information for TCP addressing.
- **Processing ID Field Checking (Debug, Test or Production).** Another new security feature includes a new check of the processing id field (Debug, Test, or Production) for incoming against the protocol definition or the HL7 site parameters. A mismatch results in an error, which, if properly used, stops messages from test accounts being processed in production accounts, and vice versa.
- **Startup and Shutdown Enhancements.** A new option makes it easier to shut down **V*IST*A** HL7: "Stop All Messaging Background Processes". Also, in the past, the "Restart/Start All Links and Filers" option would shut down *all* links, but only restart links with AUTOSTART enabled. Now it only shuts down and restarts links with AUTOSTART enabled.
- **Interface Workbench Replacement.** The former **V*IST*A** HL7 "Interface Workbench" option is replaced by the new "Application Edit", "Link Edit" and "Protocol Edit" options. Due to changes in **V*IST*A** HL7's architecture, the old option had become unintuitive to use and difficult to maintain. The new options eliminate 21 routines and 62 protocols.
- **Message Resynchronization Improvements for TCP Links.** When receiving a message over a TCP link, **V*IST*A** HL7 looks for duplicates of the message that have already been received. If the message is a duplicate, **V*IST*A** HL7 does not reprocess it again; instead, it re-sends the original acknowledgment again. Also, if attempts to send a message over a TCP link exceed the link's retransmission attempts setting, you can have the link automatically shut down and restart based on a new link setting, "Exceed Attempts Action". This flushes the message buffers and in many cases regains synchronization.
- **Time Zone Offset in Header.** The DATE/TIME field of the HL7 Message Header has been enhanced to include the time zone offset.
- **New Entry Points:** Several new entry points support manipulating messages sent over TCP links: $$MSGSTAT^HLUTIL (message status), $$MSGACT^HLUTIL (requeue or cancel transmission) and $$CHKLL^HLUTIL (validate TCP link).
- **New Option to Validate Interfaces.** Developers can use the new "Validate Interfaces" option to check interfaces for potential problems.

1.5.3 New Menu (Patch HL*1.6*57)

[HL MAIN MENU] has been redesigned, with the following organization of options:

- **Systems Link Monitor**
- **Filer and Link Management Options [menu]**
  - Systems Link Monitor
  - Monitor, Start, Stop Filers
  - TCP Link Manager Start/Stop
  - Stop All Messaging Background Processes
  - Restart/Start All Links and Filers
  - Default Filers Startup
  - Start/Stop Links
  - Ping (TCP Only)
  - Link Edit
  - Link Errors [menu]
    - Link Error/Status Report (non-TCP)
    - Show Communications Error
    - Clear Communications Error
    - Clear a Queue of All Entries
- **Message Management Options [menu]**
  - Purge Messages
  - View Transmission Log (TCP only)
  - Awaiting/Pending Transmission Report (non-TCP)
  - Failed Transmission Report (non-TCP only)
  - Message Requeuer (non-TCP)
  - HL7 V1.5 Options [HL Menu 1.5]
    - Non-DHCP Application Parameter Enter/Edit
    - Initiate Background Task
    - Start/Stop Log of HL7 Transmissions
    - Non-DHCP Application Parameters Print/Display
- **Interface Developer Options [menu]**
  - Application Edit
  - Protocol Edit
  - Link Edit
  - Validate Interfaces
  - Reports [menu]
    - Application Parameters Report
    - Data Type Report
    - Fields Report
    - Message Type Report
    - Segment Name Report
    - Version Report
- **Site Parameter Edit**

C H A P T E R

2. Link Setup

2.1 Introduction

Links (also known as logical links) describe the complete network path to a given system. They are similar in function to VA MailMan's Domain file and Kernel's Device file. Link entries hold the details of how to connect to the target system, such as IP address and Port. For every target system that you need to exchange HL7 messages with, a link needs to be set up so that **V*IST*A** HL7 knows how to reach the target system.

2.1.1 Link Types

V*IST*A HL7 supports the following types of links for outgoing connections:

- Hybrid Lower Layer Protocol (HLLP), for use over reliable serial lines
- TCP/IP, using Minimal Lower Layer Protocol (MLLP)
- X3.28, for use over unreliable serial lines
- VA MailMan

The HL7 Implementation Support Guide provides the following advice about choice of LLPs:[3]

> a) Many environments consist of simple RS-232 circuits where flow control and error recovery issues dominate the communication design. For these environments a protocol based on the American National Standards Institute Standard X3.28 is provided.
>
> b) Some LAN-based networking environments provide a reliable byte stream but insufficient session control to support HL7. Environments based on the TCP protocol of the Internet Suite are in this category. For these a very simple protocol is provided that simply delimits messages [MLLP].
>
> c) Some environments are hybrids where communications occur primarily over LANs but host connections to the LAN may be made over RS-232 ports on terminal servers. Environments such as this have some of the same problems as those operating entirely with RS-232 circuits, but the bit error rates are generally lower and flow control is available from the network. [HLLP is] significantly more efficient than the X3.28-based protocol for these environments.

[3] Health Level Seven, *Health Level Seven Implementation Guide for HL7 Standard Version 2.3*, copyright 1998, p. C-1.

2.1.2 Transport Layers and Lower Layer Protocols

A *transport layer* defines the physical connection between systems. Examples included TCP/IP networks, and serially cabled connections.

An HL7 *Lower Layer Protocol* (LLP) defines how the systems handshake with each other and exchange HL7 messages ***across*** a transport layer. While not defined within the HL7 standard itself, several LLPs are defined in the *Health Level Seven Implementation Support Guide*.

LLPs provide the lower layer communications functionality needed to exchange messages between systems, such as flow control and error recovery. "Lower layer" refers to a portion of the Open Systems Interconnect (OSI) model, which is divided into seven layers. The lower layers (1 through 4) include the actual physical connection between the systems, and the communications protocol used. The HL7 standard itself defines the seventh and highest *application* layer.

Note that the LLP used for a link is distinct from the underlying transport layer used to connect the two systems. **V***IST***A** HL7 supports *four* LLPs over *two* transport layers:

Lower Layer Protocol	Protocol Source	VA Transport Layer
Hybrid Lower Layer Protocol (HLLP)	HL7 Implementation Guide	Serial
X3.28	ANSI X3.28-1976, HL7 Implementation Guide	Serial
MailMan (an implementation of SMTP)	none	TCP/IP and Serial
Minimal Lower Layer Protocol (MLLP)	HL7 Implementation Guide	TCP/IP

Note: V*IST***A** HL7 can also exchange HL7 messages between **V***IST***A** M-based applications on the *same* system. This type of interface does not make use of links, as it bypasses the communications layer altogether.

2.1.3 How Links Are Stored

Each link is an entry in the HL Logical Link file (#870). Patch HL*1.6*57 combines the complete definition of a link into a single file:

- HL LOGICAL LINK (#870)

Prior to patch HL*1.6*57, each link consisted of:

- An entry in the HL LOGICAL LINK (#870)
- An entry in the HL LOWER LEVEL PROTOCOL PARAMETER (#869.2)

The same links can be shared by different applications sending messages to the same target system, and in fact, that is the intent behind the structure of links. Only one link needs to be set up to reach any given system via any given lower layer protocol.

2.1.4 Link "In" and "Out" Queues

Each link has an "In" and an "Out" queue:

- The "In" queue is used to receive incoming messages coming across the link.
- The "Out" queue is used to send outgoing messages across the link.

You can view the status of the "In" and "Out" queues using the Systems Link Monitor (please see the "Managing a **V***IST***A** HL7 System" chapter for more information).

When outgoing messages are sent over HLLP, X3.28 and MailMan links, **V***IST***A** HL7's filers place a copy of the message in the link's "Out" queue prior to transmission. For TCP links, the queue is used to maintain counters and status, but the message is transmitted directly from the **V***IST***A** HL7 message files; the filers never actually copy the message to the "Out" queue.

For incoming messages received over HLLP and X3.28 links, the message is first stored in the "In" queue, and later moved by filers to the **V***IST***A** HL7 message files. For TCP and MailMan links, the queue is used to maintain counters and status, but the message is placed directly in the **V***IST***A** HL7 message files.

2.1.5 Implementation Differences for the Lower Layer Protocol Types

Characteristic	HLLP	X3.28	MailMan	TCP
Outgoing messages stored in link's "Out" Queue?	Yes	Yes	Yes	No
Incoming messages stored in link's "In" Queue?	Yes	Yes	No	No
Means of addressing target system	Device File entry	Device File entry	Mail Group	TCP/IP address , port
Are incoming messages checked for communications errors (e.g., block checks)?	Yes	Yes	No	No
Is the link persistent (e.g., does a background process need to run continuously to keep the link up?)	Always	Always	Yes (Outgoing) No (Incoming)	Config-urable
Supports Synchronous Bi-directional Transactions[4]?	No	No	No	Always
Can both systems initiate sending a new message (not an acknowledgment) over the same open connection?	Yes	Yes	N/A	No

[4] When a message is sent over a connection, the expected response to that message is returned immediately, over the same open connection. The sending system does not initiate a new transaction until the current transaction is completed. The receiving system must respond to the original message without attempting to initiate a new transaction. This is called a synchronous bi-directional transaction.

2.2 How to Edit Links

To set up or edit 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: REDACTED
```

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

          NODE: REDACTED
   INSTITUTION: REDACTED
        DOMAIN: REDACTED
     AUTOSTART:
    QUEUE SIZE: 10
      LLP TYPE: TCP

______________________________________________________________________________


COMMAND:                                      Press <PF1>H for help    Insert
```

The six fields in the screen above are editable for every LLP type. In addition, every LLP type has LLP-specific fields. To edit those fields, tab down to the LLP Type field (in the "Link Edit" option) and press <RET>. The option then presents the fields relevant to that particular LLP type.

2.2.1 Who Needs to Create/Edit Links

Patch HL*1.6*39 provides TCP links for all VA facilities. For all interfaces requiring links between VA facilities, these links should be sufficient.

Other links needed for interfaces should be created by the developer (national or local) of the package owning the interface. Under some circumstances, sites may need to modify links provided by national packages to tailor them for local use.

Finally, site managers will need to create links for locally interfaced systems.

2.2.2 LLP-Specific Fields

Each link type has LLP-specific fields. To edit those fields for a given link, tab down to the LLP Type field (in the "Link Edit" option) and press <RET>. The option then presents a form to edit the fields specific to the LLP type of the selected Link:

```
                          HL7 LOGICAL LINK
------------------------------------------------------------------------------
   ┌─────────────────────────TCP LOWER LEVEL PARAMETERS──────────────────────┐
   │                        REDACTED
|
   │                                                                         │
   │   TCP/IP SERVICE TYPE: CLIENT (SENDER)                                  │
   │        TCP/IP ADDRESS: 152.199.199.199                                  │
   │           TCP/IP PORT: 5000                                             │
   │                                                                         │
   │                                                                         │
   │    ACK TIMEOUT: 40                    RE-TRANSMISION ATTEMPTS:          │
   │   READ TIMEOUT: 10                  EXCEED RE-TRANSMIT ACTION:          │
   │     BLOCK SIZE: 245                                                     │
   │                                                                         │
   │ STARTUP NODE:                                        PERSISTENT:        │
   │    RETENTION:                                 UNI-DIRECTIONAL WAIT:     │
   └─────────────────────────────────────────────────────────────────────────┘
_______________________________________________________________________________




COMMAND:                                        Press <PF1>H for help    Insert
```

2.2.3 Converting Links from One Type to Another

To *convert an existing link* from one type to another, follow this procedure:

1. Make sure all messages in the link that need to be delivered have been delivered. Otherwise, they will be lost.
2. Stop the link.
3. Use the "Clear a Queue of all Entries" option on the link. This resets all message counters and flags, ensuring the queue is initialized correctly. It deletes any messages in the "In" and "Out" queues of the link for HLLP, X3.28 and MailMan links, as well.
4. Now use the "Link Edit" option to change the link type, and set the appropriate field settings for the new link type.

Note: You can define parameters in the same link entry to support multiple link types, and then change between link types as needed (these fields are not cleared by changing the link type).

2.3 HLLP Link Setup

Hybrid Lower Layer Protocol (HLLP) links are appropriate for serial connections over reliable serial lines. For example, in an environment where a device is connected serially to a terminal server, and communication between the terminal server and host system is over a LAN, HLLP is usually appropriate. For more information about HLLP, consult the *Health Level Seven Implementation Support Guide*.

When you start an HLLP link, it opens a connection to the other system. In HLLP, both ends of the connection act as both listeners and senders, and the connection is kept open continuously. As long as an HLLP link is "up", a background process runs continuously to support it.

2.3.1 HLLP Synchronization Problems

Either side of an open HLLP link can originate a new transaction over the open connection to the other system. It's not good for both sides to do this over the same connection, however, because some COTS systems, when waiting for a response, accept the next incoming message as the response even if it is a completely new transaction. Using two serial lines, one for transactions from the COTS (and their responses from **V***IST***A**) and the other for transactions from **V***IST***A** (and their responses from the COTS system) alleviates this.

Because of the synchronization problems inherent in HLLP, we recommend that two serial lines be used if both sides of the interface can originate new messages. One line is for **V***IST***A** to initiate messages and receive the corresponding acknowledgments; the other line is for the other system to originate messages and receive acknowledgments. Most commercial vendors with HLLP interfaces also recommend the use of two lines in this situation to prevent synchronization problems.

2.3.2 Device File Entry

For HLLP links, you need to define a Device file entry so that **V***IST***A** HL7 can "open" a serial connection. The important Device file fields to set are **Name**, **Type** (set to TERMINAL) **$I** (so the system knows where the device is located), and **Subtype** (while the terminal type is largely bypassed once the device is open, the selected terminal type should not have an open or close execute).

An example of an appropriate Device file entry is as follows:

```
NAME: HLLPTEST
$I: _LTA9904:
TYPE: TERMINAL
SUBTYPE: P-OTHER
```

OpenVMS sites should also set the following device protection and characteristics, at the OpenVMS level, on the appropriate node(s), both interactively and in DCL device setup file(s):

```
$ SET PROTECT=W:RWLP /DEVICE LTA9904:
$ SET TERM/PERM/NOWRAP/HOSTSYNC/NOECHO/EIGHT/NOBROAD/ALTYPE/PASTHRU LTA9904:
```

2.3.3 Link Fields for HLLP Links

Field	Description
Node	Link name, used to identify the link in places such as the "NODE" column of the "Systems Link Monitor" option. Because only the first 8 characters of the link name show up in the Systems Link Monitor, Link names should be unique and meaningful within the first 8 characters.
Institution	In most cases, you should never set an Institution number for an HLLP link. In particular, you should not override the institution associations provided in nationally distributed links.
Domain	In most cases, you should never set a domain number for an HLLP link. In particular, you should not override the domain associations provided in nationally distributed links.
Autostart	Set to Enabled to allow this link to be autostarted or restarted whenever the "Restart/Start All Links and Filers" option is invoked.
Queue Size	The number of successfully *processed* messages to retain in the link's "In" queue, and the number of successfully *transmitted* messages to retain in the link's "Out" queue. Filers purge messages beyond the queue size. We recommend setting the queue size to 10. This setting affects traceability after messages are sent and received, and needs to be balanced with disk space. Retaining messages can be useful if you are troubleshooting the operation of a link. Also, the Message Requeuer option works only for messages retained in the queue (but this option is usually used only when testing links).
LLP TYPE	Set to HLLP.
HLLP Device	Pointer to the Device file entry for the serial device on the **V*IST*A** system.
Re-Transmission Attempts	Enter the number of times to re-try sending a message. Defaults to 5 tries if this field is left blank. For HLLP, if the limit is exceeded, an error is set into GROSS COMMUNICATIONS ERROR field of the link.
Read Timeout	Number of seconds the link remains in a read state for data to come in on the link. Defaults to 10 seconds.
Ack Timeout	Number of seconds to wait for the receiving application to generate an acknowledgment. If less than the Read Timeout, that value is used instead. Set carefully depending on the amount of time it may take an application to process a message and return an acknowledgment.
LLP Start Block	While the LLP Start Block is negotiable, the *HL7 Implementation Guide* recommends using the 'VT' character (numeric 11). Defaults to 11.
LLP End Block	In most cases, you don't need to set this field. While the LLP End Block is negotiable, the *HL7 Implementation Guide* recommends the using the 'FS' character (numeric 28). Defaults to 28.
Protocol Id Version	HLLP communications protocol version number, which may be different from the HL7 standard itself. Defaults to 2.1. Sending side and receiving side may need to match up the same HLLP protocol version.

2.4 TCP Link Setup

TCP links are suited for connections between systems that are networked together via TCP/IP:

- To originate an *outgoing* message to any destination over TCP/IP, an outgoing TCP link must be set up and running for that destination.
- To receive *incoming* new messages over TCP/IP, a running TCP/IP listener process is used.

In both cases, outgoing and incoming, the originating system controls the connection between the two systems, and is always the "sender". The receiving system can *reply* to an incoming message over the same open connection, but cannot originate a new message over that connection (it must make a separate connection to do that). This helps to maintain synchronization between systems.

This section describes how to set up outgoing TCP links. For more information on TCP/IP listeners, please see the "Managing TCP/IP Listeners" chapter.

2.4.1 Link Fields for Outgoing TCP Links

Field	Description
Node	Link name, used to identify the link in places such as the "NODE" column of the "Systems Link Monitor" option. Because only the first 8 characters of the link name show up in the Systems Link Monitor, link names should be unique and meaningful within the first 8 characters. We recommend that the name represent the target facility and/or application, and the link type, e.g., PACS1TCP.
Institution	Associates the TCP link with an Institution or VISN. Both institutions and VISNs are entries in the Institution File (#4). Packages such as CIRN-PD rely on this field when dynamically addressing messages to institutions. This field is used by LINK^HLUTIL3 to return the link(s) associated with an institution or VISN. There is a 1:1 correspondence enforced between link and institution. In general, you should not override the institution associations provided in nationally distributed links.
Domain	Associates the TCP link with a domain. Packages such as CIRN-PD rely on this field when dynamically addressing messages to domains. This field is used by LINK^HLUTIL3 to return the link associated with a domain. There is a 1:1 correspondence enforced domain and link. In general, you should not override the domain associations provided in nationally distributed links.
Autostart	Set to Enabled to allow this link to be autostarted or restarted whenever the "Restart/Start All Links and Filers" option is invoked.
Queue Size	Leave null (TCP links do not use queues.)

Field	Description
LLP Type	Set to TCP.
TCP/IP Service Type	Set to "CLIENT (SENDER)".
TCP/IP Address	Set to the IP address of target system, e.g., 152.001.32.5.
TCP/IP Port	Set to the Port to connect to on target system, e.g., 5000.
Ack Timeout	Number of seconds to wait for an acknowledgment from the other system. If less than the Read Timeout, that value is used instead. Set this carefully depending on the amount of time it may take an application to process a message and return an acknowledgment.
Read Timeout	Number of seconds to read across the link before timing out. Defaults to 10.
Block Size	Size of a read that will be set to a global node. Make sure that this is not set above your system's maximum global node size.
Re-Transmission Attempts	Enter the number of times to re-try sending a message. Defaults to 5 tries if this field is left blank. For TCP, if the limit is exceeded, 1) an alert is sent to the mail group specified in the "Mail Group for Alerts" site parameter, and 2) the Exceed Attempts Action field determines any additional action to perform on the link.
Exceed Attempts Action	Action to take if the number of attempts to transmit a message exceeds the Re-Transmission Attempts setting. Choices are: Shutdown (to shut down the link), Restart (to shut down and restart the link), or Ignore (to keep retrying infinitely). Defaults to Ignore.
Startup Node	Set the startup node for the link. This setting is honored if a) Task Manager is running on the specified startup node, or b) you are an OpenVMS site running TaskMan in a DCL context.
Persistent	Set to "YES" to keep the link's connection open even when there are no messages to send. The connection is only disconnected if either side shuts down the link. Appropriate for many COTS devices. Set to "NO" if the link's connection does not need to remain open. Appropriate for connections to other **V*IST*A** facility systems. Please see the "Managing a **V*IST*A** HL7 System" chapter for more information.
Retention	For non-persistent links only (Persistent is set to "NO"). Determines the maximum time in seconds that the process supporting the link should wait, after delivering all messages, before terminating. Defaults to the value of the "Default Retention" site parameter. Please see the "Managing a **V*IST*A** HL7 System" section chapter for more information.
Uni-Directional Wait	If the message sent from **V*IST*A** requires neither commit nor application acknowledgment, it is delivered unidirectionally. Some vendor systems need a delay between each such message; use this field to set that delay.

2.4.2 TCP Links to Other VHA Facilities

Patch HL*1.6*39 provides the first set of operational TCP links for all of VHA's **V*IST*A** facilities. To exchange messages with a VA facility over TCP, ordinarily you should use the links provided in patch HL*1.6*39, or subsequent patches, to do this.

Until an adequate administrative implementation of DNS for sites' HL7 TCP/IP listeners is in place, the IP addresses and ports for TCP links will be maintained manually, via the patch release process. This is similar to the situation for maintaining IP addresses and ports in the Domain file for VA MailMan.

Note: All links are exported as client (outgoing) links. The link for *your* site distributed in patch HL*1.6*39 must be changed to a TCP Listener and enabled, to receive connections.

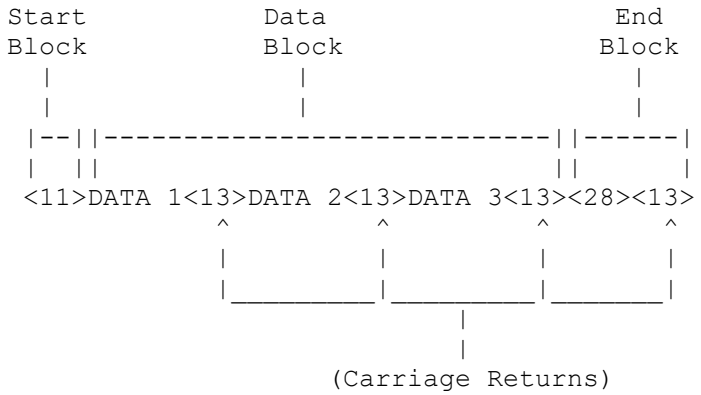
2.4.3 V*IST*A HL7 TCP/IP LLP Implementation Details

The information provided in this section is useful if you are interfacing a device or system with **V*IST*A** over TCP/IP and need to know the specifics of MLLP as it is implemented by **V*IST*A** HL7. MLLP is a very simple protocol, and can be used because the TCP/IP channel itself provides most services needed for error-free transmission of messages.

V*IST*A HL7's implementation of messaging over TCP/IP uses Minimal Lower Layer Protocol (MLLP). For more information about MLLP, consult the *Health Level Seven Implementation Support Guide*.

The MLLP message format provided by (and expected by) **V*IST*A** HL7 is as follows:

- Start Block = $C(11)
- End Block = $C(28)
- Carriage Return = $C(13)



This format is fixed and not subject to modification. Vendors interfacing with **V*IST*A** over TCP/IP must support **V*IST*A**'s implementation of MLLP.

2.5 MailMan Link Setup

MailMan links are used to send *outgoing* messages to other **V*IST*A** systems via VA MailMan.

They deliver messages to the appropriate target systems by sending a message to all members of a specified mail group. Each member should be an HL7 server option at a target facility (e.g., S.HL V16 SERVER@REDACTED). Multiple members allow delivery of HL7 messages to multiple systems.

To send outgoing messages, you need to start MailMan links just as with any other link type. A background process runs continuously to support the link.

Note that whether or not a particular MailMan link is running or stopped has no effect on *incoming* messages received through VA MailMan. Incoming messages bypass links and are instead processed directly by the server option that receives them. The only way to prevent processing of such messages is to disable the server options.

2.5.1 Configuring Server Options to Receive Messages

When an HL7 message is *received* via MailMan on your system, MailMan invokes one of the **V*IST*A** HL7 server options on your system (the message will be addressed to the server option on your system). The server option in turn directly invokes the appropriate message processing routine (no "In" queue or inbound filer is involved).

The server options (special entries in the Option file) exported by **V*IST*A** HL7 are:

- HL V16 SERVER (1.6 server, for messages sent over **V*IST*A** HL7 V. 1.6 interfaces)
- HL SERVER (1.5 server, for messages sent over VISTA HL7 V. 1.5 interfaces)

As per the Kernel documentation for server options, in order to function, each *server option* above **must** be associated with a mail group. To make this association:

1. Edit each server option's SERVER MAIL GROUP field.
2. Set the field to a mail group that has at least one active, local user member (for example, a local IRM mail group).

Note: V*IST*A HL7's server options are exported with SERVER ACTION set to RUN IMMEDIATELY, but with SERVER MAIL GROUP set to null.

For more information about setting up and configuring Kernel server-type options, please refer to the *Kernel Systems Manual*.

2.5.2 Link Fields for MailMan Links

Field	Description
Node	Link name, used to identify the link in places such as the "NODE" column of the "Systems Link Monitor" option. Because only the first 8 characters of the link name show up in the Systems Link Monitor, Link names should be unique and meaningful within the first 8 characters. We recommend that the name represent the target facility and/or application, and the link type, e.g., REDACTEDMM.
Institution	In most cases, you should never set an Institution number for a MailMan link. In particular, you should not override the institution associations provided in nationally distributed links.
Domain	In most cases, you should never set a domain for a MailMan link. In particular, you should not override the domain associations provided in nationally distributed links.
Autostart	Set to Enabled to allow this link to be autostarted or restarted whenever the "Restart/Start All Links and Filers" option is invoked.
Queue Size	The number of successfully *transmitted* messages to retain in the MailMan link's "Out" queue, before purging those messages. This setting affects traceability after messages are sent, and needs to be balanced with disk space. Retaining messages can be useful if you are troubleshooting the operation of a link. Also, the requeue/retransmit message options work off of the "Out" queue (but these are usually used only when testing links). We recommend setting the queue size to 10.
LLP Type	Set to MailMan.
Mail Group	Pointer to File #3.8 (Mail Group). This mail group should contain the destination members to whom the HL7 messages going out over this link should be addressed. Each member should be the HL7 server option at a particular remote facility. For example: S.HL V16 SERVER@VAFACILITY1.MED.VA.GOV

2.5.3 Cleaning up Old Messages in Server Option Mail Baskets

You should occasionally review the S.* POSTMASTER mailboxes in MailMan. Each HL7 server option has a mail basket under the POSTMASTER. **V*IST*A** HL7 cleans up old messages received in these mailboxes by the HL7 server options. However, there have occasionally been old messages found that weren't cleaned up, particularly before procedures were established for cleaning up server baskets. In any such case that you find currently, try to determine the cause of why the messages are left in the server basket (check the error traps to see for HL7 errors). The only way to clean up old HL7 messages in the server baskets is manually.

2.5.4 Controlling Incoming Message Volume with a Resource Device

You should consider attaching a resource device to the HL V16 SERVER option to prevent the option from becoming overloaded. The resource device lets you restrict the number of concurrent HL V16 SERVER jobs to a specified number. To do this, you should:

1. Create a resource device, named HLCS SERVER
2. Attach the resource device to the HL V16 SERVER option

For more information on resource devices, please see the *Kernel Systems Manual*.

An example of setting up the HLCS SERVER and attaching it to the HL V16 SERVER option is provided below.

Setting up the Resource Device

```
Select Systems Manager Menu Option: Device Management

Select Device Management Option: Edit devices by Specific Types

Select Edit Devices by Specific Types Option: RESOURCE DEVICE EDIT

Select Resource Device: HLCS RESOURCE
NAME: HLCS RESOURCE// <RET>
$I: HLCS RESOURCE
VOLUME SET(CPU): <RET>
RESOURCE SLOTS: 3      <- enter how many slots to best handle your load

Select Resource Device:
```

Attaching the Resource Device to the HL V16 SERVER Option

```
>D Q^DI
VA FileMan 21.0

Select OPTION: 1  ENTER OR EDIT FILE ENTRIES

INPUT TO WHAT FILE: OPTION
     1   OPTION                          (1183 entries)
     2   OPTION SCHEDULING               (18 entries)
CHOOSE 1-2: 1
EDIT WHICH FIELD: ALL// 227  SERVER DEVICE
THEN EDIT FIELD:

Select OPTION NAME: HL V16 SERVER          HL7 v1.6 Mail Message
Server
SERVER DEVICE: HLCS RESOURCE        <- name of newly created resource

Select OPTION NAME:
```

2.6 X3.28 Link Setup

X3.28 links provide complicated and comprehensive error checking. This error checking is redundant when used over reliable communications lines. As such, X3.28 should usually be used for unreliable serial connections (something which VA facilities strive to eliminate). Setting up X3.28 links is complex because of the many parameters for error checking and recovery that, in other circumstances, are handled by the underlying communications layer.

V*IST*A HL7's implementation of the X3.28 lower layer protocol is the most similar to its implementation of HLLP. Unlike HLLP, however, while the connection is up, each side of an X3.28 interface negotiates, on-the-fly, which side will be the "master", sending messages, and which side will be the "slave", receiving messages.

When you start an X3.28 link, it opens a connection to the other system. As long as an X3.28 link is "up", a background process runs continuously to support it.

Guidance on the specifics of setting up X3.28 links goes beyond the scope of this manual. For more information, please consult the *Health Level Seven Implementation Support Guide*, the ANSI X3.28 standard, and your vendor documentation if you are interfacing a vendor device that supports X3.28.

2.6.1 Device File Entry

For an X3.28 link, you need to define a Device file entry so that **V*IST*A** HL7 can "open" a serial connection. The important Device file fields to set are **Name**, **Type** (set to TERMINAL) **$I** (so the system knows where the device is located), and **Subtype** (while the terminal type is largely bypassed once the device is open, the selected terminal type should not have an open or close execute).

An example of an appropriate Device file entry is as follows:

```
NAME: HLLPTEST
$I: _LTA9904:
TYPE: TERMINAL
SUBTYPE: P-OTHER
```

OpenVMS sites should also set the following device protection and characteristics, at the OpenVMS level, on the appropriate node(s), both interactively and in DCL device setup file(s):

```
$ SET PROTECT=W:RWLP /DEVICE LTA9904:
$ SET TERM/PERM/NOWRAP/HOSTSYNC/NOECHO/EIGHT/NOBROAD/ALTYPE/PASTHRU LTA9904:
```

2.6.2 Link Fields for X3.28 Links

Field	Description
Node	Link name, used to identify the link in places such as the "NODE" column of the "Systems Link Monitor" option. Because only the first 8 characters of the link name show up in the Systems Link Monitor, Link names should be unique and meaningful within the first 8 characters. We recommend that the name represent the target application, and the link type, e.g., KURZWEIL1, KURZWEIL2, etc.
Institution	In most cases, you should never set an Institution number for an X3.28 link. In particular, you should not override the institution associations provided in nationally distributed links.
Domain	In most cases, you should never set a domain for an X3.28 link. In particular, you should not override the domain associations provided in nationally distributed links.
Autostart	Set to Enabled to allow this link to be autostarted and/or restarted whenever the "Restart/Start All Links and Filers" option is invoked.
Queue Size	Number of successfully *processed* messages to retain in the link's "In" queue, and successfully *transmitted* messages to retain in the link's "Out" queue, before those queues are purged by filers. Defaults to 100. This setting affects traceability after messages are sent and received, and needs to be balanced with disk space. Retaining messages can be useful if you are troubleshooting a link. Also, the requeue/retransmit message options work off of the "Out" queue (but these are usually used only when testing links). We recommend setting the queue size to 10.
LLP TYPE	Set to X3.28.
X3.28 Device	Pointer to the Device file entry for the serial device on the **V*IST*A** system.
Maximum Message Size	Defaults to 99999.
Maximum Block Size	Size of a read that will be set to a global node. Make sure that this is not set above your system's maximum global node size. Defaults to 245.
Timer A	Time in seconds for the Response Timer. Defaults to 6 seconds.
Timer B	Time in seconds for the Receive Timer. Defaults to 3 seconds.
Timer D	Time in seconds for the Inter-Block timer. Defaults to 30 seconds.
Timer E	Enter the time for the Line Check Timer. Default to 180 seconds.

CHAPTER

3. Managing a V*ISTA* HL7 System

3.1 Managing Filers

Filers are one of **V***IST***A** HL7's background processes. There are two types of filers, incoming and outgoing:

- *Incoming* filers copy incoming messages received by a link (and stored in the link's "In" queue) into the Message Text and Message Administration files, and call the appropriate application processing routine to process the message.
- *Outgoing* filers assemble and copy outgoing messages from the Message Text and Administration files to the appropriate link's "Out" queue, after which the link assumes responsibility and in turn delivers the messages to other systems.

As such, keeping the filers up and running is necessary to keep messaging coming in and out of your system.

Note: Filers are not needed for all link types. The following table identifies which link types depend on incoming and/or outgoing filers:

Link Type	Receiving Messages Depends on Incoming Filer?	Sending Messages Depends on Outgoing Filer?
HLLP	Yes	Yes
X3.28	Yes	Yes
MailMan	Yes	Yes
TCP	TCP links do not use incoming filers, *except* when the transaction requires a deferred, two-phase acknowledgment. For deferred, two-phase acknowledgments, upon message receipt, the **V***IST***A** HL7 TCP/IP listener invokes an incoming filer, to trigger the return of an application ack over a separate connection.	No
Same System (no link)	No	No

For more information about link types, please see the "Link Setup" chapter.

3.1.1 Monitoring Filers

Use the "Monitor, Start, Stop Filers" option to monitor the incoming and outgoing filers that are running on your system. This option provides a Filer Monitor in List Manager (a list processor utility) screen format, including actions to start, stop, or delete filers.

Your goal when monitoring filers (if the **V*IST*A** HL7 system is running) is to ensure that the appropriate numbers of incoming and outgoing filers are running and that none have errorred or stopped inappropriately.

```
Task Number of    Asked
Incoming Filer    To Stop   Last Known Date/Time   Time Difference
2713062           No        16-JUN-99 @ 14:33:52   0 Day  00 Hr  00 Min  00 Sec
[End of list - total of 1]



Task Number of    Asked
Outgoing Filer    To Stop   Last Known Date/Time   Time Difference
2713063           No        16-JUN-99 @ 14:33:49   0 Day  00 Hr  00 Min  03 Sec
[End of list - total of 1]



(+I) Start incoming filer  (-I) Stop incoming filer  (*I) Delete incoming filer
(+O) Start outgoing filer  (-O) Stop outgoing filer  (*O) Delete outgoing filer
 (N) Next 4 lines in list   (B) Back 4 lines in list  (Q) Quit
Type selection:
```

- **Task Number of (Incoming/Outgoing) Filer** - The task number of each filer.
- **Asked To Stop** - Indicates whether the filer has been flagged to stop.
- **Last Known Date/Time** - The most recent date and time that the filer updated its status.
- **Time Difference** - The difference between the current date and time and the last date and time that the filer updated its status. This difference should never be greater than five to six minutes; otherwise, the filers are probably not running.

If there is a filer error, instead of the above values for "Last Known Date/Time" and "Time Difference", one of the following messages is displayed, for both incoming and outgoing filers:

```
Outgoing filer has not started yet
Outgoing filer stopped but didn't delete itself
** Outgoing filer is no longer defined **
** Outgoing filer has not been [re]scheduled **
** Outgoing filer has stopped due to error **
```

3.1.2 Actions in the Filer Monitor

To execute any of the following actions, at the "Monitor, Start, Stop Filers" option's "Type selection:" prompt, enter the character(s) shown in the parentheses. *Do not press the Return or Enter keys after you enter your response, as these will return you to the menu.*

- **(+I) Start incoming filer** - Queues a background job, assigns a task number, and adds it to the list of incoming filers. The message "**Incoming filer has not started yet**" is displayed until the filer is physically started.

- **(+O) Start outgoing filer** - Queues a background job, assigns a task number, and adds it to the list of outgoing filers. The message "**Outgoing filer has not started yet**" is displayed until the filer is physically started.

- **(-I) Stop incoming filer** - Searches through the list of incoming filers until it finds the *first* one that is running and flags it to stop. "Yes" is displayed in the "Asked To Stop" column. The message "**Incoming filer has been asked to stop**" is displayed until the filer is physically stopped, at which time it is automatically deleted from the Filer Monitor display and the list of incoming filers that are running.

- **(-O) Stop outgoing filer** - Searches through the list of outgoing filers until it finds the *first* one that is running and flags it to stop. "Yes" is displayed in the "Asked To Stop" column. The message "**Outgoing filer has been asked to stop**" is displayed until the filer is physically stopped, at which time it is automatically deleted from the Filer Monitor display and the list of outgoing filers that are running.

- **(*I) Delete incoming filer** - Use this option only when "Error" is displayed in the "Asked To Stop" column, the message "**Incoming filer has stopped due to an error**" is displayed, and the filer is stopped. First, use the error trap to diagnose the problem, and then use this action to delete the filer(s) from the Filer Monitor display and from the list of incoming filers that are running. If "Error" is displayed for multiple filers, you are prompted to select which of the filer(s) you want to delete. If "Error" displays for a single filer, it will be automatically deleted when you use this action.

- **(*O) Delete outgoing filer** - Use this option only when "Error" is displayed in the "Asked To Stop" column, the message "**Outgoing filer has stopped due to an error**" is displayed, and the filer is stopped. First, use the error trap to diagnose the problem, and then use this action to delete the filer(s) from the Filer Monitor display and from the list of outgoing filers that are running. If "Error" is displayed for multiple filers, you are prompted to select which of the filer(s) you want to delete. If "Error" displays for a single filer, it will be automatically deleted when you use this action.

- **(N) Next 4 lines in list** - Displays the next 4 lines in the Filer Monitor for both incoming and outgoing filers.

- **(B) Back 4 lines in list** - Backs up 4 lines in the Filer Monitor for both incoming and outgoing filers.

- **(Q) Quit** - Exits the option and returns you to the menu.

3.1.3 Options to Start Filers

Filers must be running on the system for messages to be processed. **V*IST*A** HL7 provides the following options to start filers:

- Default Filers Startup
- Monitor, Start, Stop Filers
- Restart/Start All Links and Filers

Ordinarily, the Restart/Start All Links and Filers option should be scheduled to run at system boot. You can also run this option interactively from the **V*IST*A** HL7 menu.

Note: Process creation will take some time if you have many links to start up. For information on how to start filers automatically on system boot, please see the "Managing a **V*IST*A** HL7 System" chapter.

To add additional running filers beyond the default number of filers, use the "Monitor, Start, Stop Filers" option. This allows you to add individual incoming and outgoing filers to the ones already running on your system.

3.1.4 Stopping Filers

Stop individual filers with the "Monitor, Start, Stop Filers" option. To stop *all* filers, use the "Stop All Messaging Background Processes" option.

The consequences of stopping filers are as follows:

- **Incoming Filers Not Running**

  When incoming filers aren't running, incoming messages over HLLP and X3.28 links will not be processed until filers are started (and will build up in the HL Logical Link file, in each link's "In" queue). Also, two-phased transactions over TCP links will not be able to complete. Messages coming in over MailMan links are unaffected.

- **Outgoing Filers Not Running**

  When outgoing filers aren't running, outgoing messages to HLLP, X3.28 and MailMan links will not be delivered until filers are started. Messages going out over TCP links are unaffected.

- **Filers Running, but Link(s) Not Running**

  In this case, the link's "Out" queue would build up, as outbound filers drop off more messages to be transmitted. Inbound filers would empty the link's "In" queue until empty.

For a graceful system shutdown, you should always shut down the filers before shutting down your system. For more information on gracefully shutting down **V*IST*A** HL7, please see the "Managing a **V*IST*A** HL7 System" chapter.

3.1.5 Setting the Default Number of Incoming & Outgoing Filers

The number of filers running on your system helps determine the responsiveness of your **V*IST*A** HL7 system to incoming and outgoing HL7 messages. More filers can result in faster message processing; there is a point of diminishing returns however, and running more filers (which run as background processes) results in a consumption of more system resources.

For most VA facility systems, the *optimum* number of filers is:

- 4 incoming filers (or less)
- 4 outgoing filers (or less)

A *minimum* of one incoming and one outgoing filer, running, are needed for **V*IST*A** HL7 to process messages. For operational purposes, however, we recommend a minimum of *two* incoming and *two* outgoing filers.

Site Parameters for Filers

Once you determine the appropriate number of filers to start, use the Site Parameter Edit option to edit the following site parameters:

- Default Number of Incoming Filers
- Default Number of Outgoing Filers

The values you enter for these fields are used to start the corresponding number of filers by the "Default Filers Startup" and "Restart/Start All Links and Filers" options. A default value of "1" is set for both fields during package initialization.

Tuning the Number of Incoming Filers

For incoming filers, to help determine you have an appropriate number running, go to the Systems Link Monitor. For *running* links, compare the number of messages received to the number of message processed. If the number of messages processed is, on a regular basis, significantly lagging the number of messages received, you probably do not have enough incoming filers running. Consider adding additional filers to process incoming message more quickly.

Tuning the Number of Outgoing Filers

We recommend that you set the default number of outgoing filers equal to the number of inbound filers.

3.2 Managing Links

3.2.1 Systems Link Monitor

The single **V*IST*A** HL7 option you'll most use on a daily basis to monitor the status of your HL7 interfaces is the Systems Link Monitor. The Systems Link Monitor screen is displayed as follows:

```
                SYSTEM LINK MONITOR for ISC SAN FRANCISCO (T System)

              MESSAGES  MESSAGES   MESSAGES  MESSAGES  DEVICE
    NODE      RECEIVED  PROCESSED  TO SEND   SENT      TYPE     STATE
    M-NXT2KR  0         0          6         0                  Halting
    RH L7090  1         1          1         1            SS    Reading
    RH CLIEN  1         1          1         1                  Shutdown









    Incoming filers running => 1            TaskMan running
    Outgoing filers running => 1            Link Manager running

    Select a Command:
(N) NEXT (B) BACKUP (Q) QUIT (A) ALL LINKS (S) SCREENED (?) HELP:
```

Your goal when monitoring links is to ensure that each active link is sending and receiving messages. To do this, for each link:

1. Examine "Messages Received" vs. "Messages Processed". The count in each column should be equal, or, if not equal, the number of messages processed should be incrementing. If not, you may have a problem with inbound filers.
2. Examine "Messages to Send" vs. "Messages Sent". The count in each column should be equal, or, if not equal, the number of messages sent should be incrementing. If not, there may be a problem with the link.
3 Examine the "State" column. If the link should be running, and its state is one of the non-operational states listed on the following page, there may be a problem with the link.
4. If a link *flashes* in the Messaging Monitor, it means that an LLP communications error has occurred. To display the most recent communications error for a particular link, use the Show Communications Error option. To clear the communications error for a particular link, use the Clear Communications Error option.

The Systems Link Monitor displays statuses for up to 10 links at a time. Navigate through your link displays with the following keystrokes (*do not press the Return or Enter keys after you enter your response, as these will return you to the menu*):

(N) NEXT	Takes you to the next page of the display of links.
(B) BACKUP	Takes you back one page.
(Q) QUIT	Terminates the monitor.
(A) ALL LINKS	Provides a display of all links defined on your system.
(S) SCREENED	Displays only those links that have message traffic (i.e., the sum of the 4 counter columns > 0). This is the default display at startup.
(?) HELP	Provides this help online.

Besides each running link, the Systems Link Monitor also indicates:

- The number of incoming filers running
- The number of outgoing filers running
- Whether TaskMan is running
- Whether the TCP Link Manager is running

Monitor Column	Explanation
Node	Name of the link.
Messages Received	# of messages received over the link, regardless of whether the message is waiting to be processed or has already been passed to application processing routines.
Messages Processed	# of messages passed to application processing routines.
Messages To Send	# of messages placed in the link's "Out" queue since the last time it was reset, regardless of whether the message has been transmitted or is waiting to be transmitted.
Messages Sent	# of messages transmitted.
Device Type	Type of link. As of patch HL*1.6*49, can be one of the following: PC Persistent TCP Client NC Non-Persistent TCP Client SS Single-threaded TCP Server MS Multi-threaded TCP Server SH Serial HLLP SX Serial X3.28 MM MailMan
State	Current state of the LLP (e.g., Idle, Shutdown, Reading, Writing, Validating, Enabled, Inactive, etc.)

For more information on troubleshooting links, please see the "Troubleshooting: Solving Transmission Problems" chapter.

3.2.2 Operational Link States (Normal)

State	Explanation
Bidding	X3.28 links: Switching roles, server to sender
Check Out	Checking the "Out" queue for messages to send.
Disconnect	X3.28: Line is disconnected.
Done	HLLP: Message was validated.
Enabled	Non-persistent TCP links: Link has been started.
Idle	No messages are waiting to be sent or received. Idle cycle time is 3 seconds.
Inactive	Non-persistent TCP links: Link has been started and has delivered messages, but because there are no messages to deliver currently, the background job has been inactivated. The TCP Link Manager will reactivate it as needed.
Open	Link is attempting to open a connection.
Polling	X3.28: Link is checking if there is a message to send.
Reading	Link is reading a new message from the connected system.
Retention	Non-persistent TCP: Link has delivered messages, but has no more to send; the background process is waiting until either the retention time expires or new messages show up that need to be delivered.
Send	Link is transmitting a message.
Validate	HLLP: Link is calculating a checksum and verifying the value.
Wait ACK	X3.28: Link is waiting for an acknowledgment.
Writing	HLLP: Link is sending a message.

3.2.3 Abnormal or Non-Operational Link States

State	Explanation
Error	Link encountered an error.
Halting	Link has been asked to shut down.
NAK	HLLP: A negative acknowledgment has been sent.
OpenFail	Link could not open a connection to its associated device or target system.
Send NAK	X3.28: Link is sending a negative acknowledgment.
Shutdown	Link has been shut down.
Timeout	HLLP: When trying to read from the connected system, a timeout was encountered.

3.2.4 Starting Links

V*IST*A HL7 provides the following options to start Links:

Option	Action
Start/Stop Links	Lets you start individual links. It asks you to select a link to start, and the mode in which to start it. You have the following choices: • Background (normal) • Foreground (for troubleshooting only) • Quit without starting the receiver For all purposes other than debugging, you should always run links in the background.
Restart/Start All Links and Filers	Starts all links whose Autostart field is set to Enabled. If links are already running, they are shut down *and* restarted (but links without Autostart enabled are neither shutdown or restarted). Ordinarily, you would schedule this option to run at system boot.

3.2.5 Non-Persistent Links: TCP Link Manager

TCP links can be either *persistent* or *non-persistent*, controlled by each link's "Persistent" setting:

- *Persistent:* The M process supporting the TCP link runs continuously, whether or not any messages are being exchanged. This is how HLLP, X3.28 and MailMan links *always* operate.
- *Non-persistent:* A new feature for TCP links only. The M process supporting the TCP link only runs as long as there are messages to exchange.

Processes for non-persistent links are relinquished when there are no more messages to exchange over the link. This is accomplished through the auspices of the TCP Link Manager. The TCP Link Manager runs in the background, and starts up a new M process whenever a message needs to be delivered over an inactive (but not shut down) non-persistent link. This is useful for interfaces to other VA facilities; you wouldn't want 150+ running link processes.

To start the TCP Link Manager, use the "TCP Link Manager Start/Stop" (interactive startup), or the "Autostart Link Manager" option (automatic startup).

The "Default Retention Time" site parameter controls how long the process supporting a non-persistent link remains active after there are no more messages to send, before being relinquished. It is overridden by the link's "Retention" setting. For more information, please see "**V*IST*A** HL7 Site Parameters" elsewhere in this chapter.

Note: Non-persistent TCP links must still be explicitly *started*, either manually or by utilizing the link's AUTOSTART feature.

3.2.6 Starting TCP/IP Listeners

All TCP/IP listeners are tracked by the Systems Link Monitor option, and can be monitored by using that option. Start TCP/IP listeners as follows:

- For single-threaded and Caché multi-threaded listeners, start the listener's link.
- For UCX-based listeners on OpenVMS systems, start the listener's UCX service.

For more information on starting each listener type, please see the "Managing TCP/IP Listeners" chapter.

3.2.7 Stopping Links

When you shut your M system down, it is important to gracefully shut the links down. This helps to prevent message synchronization problems between systems.

You may also need to stop links during installation of packages that use **V*IST*A** HL7 messaging, to avoid overwriting routines while active message processing is taking place.

V*IST*A HL7 provides the following options to stop links:

- Start/Stop Links
- Shut Down all Messaging Background Processes

Using these options, you can stop any running link *except* for UCX-based TCP/IP listeners on OpenVMS Systems. For UCX-based TCP/IP listeners, to stop incoming messages from being received, disable the UCX service for the listener. For more information, please see the "Managing TCP/IP Listeners" chapter.

Effects of Stopping Links

LLP Type	Stops receipt of inbound messages?	Stops delivery of outbound messages?
HLLP	Yes	Yes
X3.28	Yes	Yes
MailMan	No	Yes
TCP (outgoing links)	N/A	Yes
TCP (listener, non-UCX)	Yes	N/A
TCP (listener, UCX)	N/A (use UCX to stop)	N/A

Note: For MailMan links, unlike other types of links, you cannot stop inbound messages from being received and processed simply by shutting down the link. Instead, to stop incoming messages from being *processed*, you must either:

- Disable the HL7 server options, stopping *all* inbound MailMan-based HL7 processing, or
- Stop all inbound filers, which disables *all* inbound messaging over non-TCP links.

3.3 Checking for Message-Level Errors

On a regular basis, you should monitor incoming and outgoing messages that have been placed in an error status by **V*IST*A** HL7.

The options to monitor message-level errors are:

Option	Description
View Transmission Log (TCP Only)	For messages sent/received over TCP links, allows you to review a listing of all messages with an error status.
Failed Transmissions Report (non-TCP)	Lists all messages transmitted over non-TCP links with a status of ERROR in field #20 of File #772.
Link Error/Status Report (non-TCP)	Lists all messages transmitted over non-TCP links with an error status in a link's "In" or "Out" queue.

To review errors for messages sent/received over TCP interfaces:

1. Use the "View Transmission Log (TCP Only)" and review the listing of all messages with an error status.
2. Get the message IEN from the first column.
3. Look up the message in File #773, look at the Error Message and Error Type fields for more information about the error.

To review errors for messages sent over HLLP, X3.28, MailMan (and same-system) interfaces:

1. Run the "Failed Transmissions Report (non-TCP).
2. For each listed message, review the "Error Message" and "Error Type" fields. Note: the Error Type field is a pointer to the HL7 Error Message file (#771.7).

In order to *purge* messages with an error status, you need to run the Purge Messages option interactively.

You should review messages with an error status before purging them. In most cases, you should attempt to find the cause of the errors, and fix the cause of the error. Then, you may want to use the sending application (on either side of the interface) to retransmit the message, if necessary.

3.4 V*IST*A HL7 Background Processes

HL7 Process	Main Routine	OpenVMS Proc. Name	Description
Filer (incoming)	HLCSIN	BTask #######	Moves incoming messages to **V*IST*A** applications from links.
Filer (outgoing)	HLCSOUT	BTask #######	Moves outgoing messages generated by **V*IST*A** applications to links.
HLLP link	HLCSDR	BTask #######	One process runs continuously per link.
MailMan link	HLCSMM1	BTask #######	One process runs continuously per link
X3.28 link	HLCSDL	BTask #######	One process runs continuously per link.
TCP link (outgoing)	HLCSTCP2	HLClnt:ien	One process runs per outgoing, active, persistent TCP link. Processes for non-persistent TCP links are started by the TCP Link Manager and run only as long as there are messages to transmit. IEN in the process name is the IEN of the link in the HL Logical Link file.
TCP Link Manager	HLCSLM	HLmgr: #######	Runs full-time in the background. It starts temporary processes to service non-persistent (outgoing client) TCP links.
TCP Listener (single-threaded)	%ZISTCP	HLSrv:ien	If in use at your site, listens for incoming TCP connections (and only allows one TCP connection at a time). IEN is the IEN of the listener's entry in the HL Logical Link file.
TCP Listener (multi-threaded for Caché)	%ZISTCPS	HLSrv:ien	Part of the communications module; listens for incoming TCP connections (and allows multiple simultaneous connections). IEN is the IEN of the listener link's entry in the HL Logical Link file.
TCP Listener (UCX)	not an M process	not an M process	Multi-threaded listeners for OpenVMS systems are implemented using UCX For more information, please see the "Managing TCP/IP Listeners" chapter.
Processes spawned by TCP Listener (UCX)	HLCSTCP1	HLSrv:ien	When the UCX listener spawns a job to process an incoming connection request, an M process is launched to handle the request. IEN is the IEN of the listener's entry in the HL Logical Link file.

Note: ####### denotes Task Number in the process names above.

3.5 HL7 Startup

3.5.1 Manual Startup

To manually start up **V*IST*A** HL7, execute the following steps in the listed order:

1. Use the "Restart/Start All Links and Filers" option to start the default number of filers, and all links whose "Autostart" field is enabled (including non-UCX TCP/IP listeners).
2. Use the "TCP Link Manager Start/Stop" option to start the TCP Link Manager.
3. For OpenVMS sites only, start up any UCX-based TCP/IP Listener processes according to the guidelines provided in the "Managing TCP/IP Listeners" chapter.

3.5.2 Automatic Startup

You can set up the **V*IST*A** HL7 background processes to start automatically on system boot:

Process	How to Automatically Start on System Boot
Filers and Links	1. Go to TaskMan Management | Schedule/Unschedule Options. Add [HL TASK RESTART] ("Restart/Start All Links and Filers") as an option to schedule. Set SPECIAL QUEUEING to "Startup Persistent". Doing this will run this option at system startup. 2. For each link that you want to start automatically on system boot, set the link's AUTOSTART field to ENABLED. Each link set this way will be started by the "Restart/Start All Links and Filers" option. 3. Use **V*IST*A** HL7's site parameters to set the default number of incoming filers and outgoing filers. This number of filers will be started by the "Restart/Start All Links and Filers" option.
TCP Link Manager	1. Go to TaskMan Management | Schedule/Unschedule Options. Add [HL AUTOSTART LINK MANAGER] ("Autostart Link Manager") as an option to schedule. Set SPECIAL QUEUEING to "Startup Persistent". Doing this will run this option at system startup.
TCP Listeners	Varies depending on system type. Please see the "Managing TCP/IP Listeners" chapter for more information.

3.6 HL7 Shutdown

When you shut down your system, you should include shutting down **V*IST*A** HL7 as part of your normal shutdown procedure. Explicitly shutting down **V*IST*A** HL7 helps avoid message resynchronization problems between systems.

To gracefully shutdown **V*IST*A** HL7, execute the following steps in the listed order:

1. For OpenVMS sites only, if you have any TCP/IP listeners implemented through UCX, use UCX to shut those listeners down. Do this by disabling the each listener's service in UCX.
2. Stop the TCP/IP Link Manager with the "TCP Link Manager Stop/Start" option.
3. Run the "Stop All Messaging Background Processes" option. This stops all running links, filers and non-UCX TCP listeners.

3.7 V*IST*A HL7 Site Parameters

Each of the HL7 parameters is important and should be set appropriately.

Use the "Site Parameter Edit" option to edit the site parameters for **V*IST*A** HL7:

```
                        Edit HL7 Site Parameters


                     Current Domain: MNTHM.VA.GOV
                Current Institution: MOUNTAIN HOME
Is this a Production or Test Account? Production

   Default Number of Incoming Filers: 3
   Default Number of Outgoing Filers: 3

              Mail Group for Alerts: HL7

     Days to Keep Completed Messages: 5
  Days to Keep Awaiting ACK Messages: 25
    Days Before Purging All Messages: 60

             Default Retention Time: 120
_______________________________________________________________________________



COMMAND:                                      Press <PF1>H for help    Insert
```

Parameter	Description
Current Domain	Pointer to the Domain file. Select the domain name for this environment. It should be unique compared to all other environments running at your site. For example, if your production domain is SITE.MED.VA.GOV and this is your test system, you should have a different entry in your Domain file, e.g., TEST.SITE.MED.VA.GOV. This is used to populate the sending facility field of HL7 message headers generated on your system.
Current Institution	Pointer to the Institution file. Select the institution for this environment. This is used to populate the sending facility field of HL7 message headers generated on your system.
Is this a Production or Test Account?	Type of environment for this account. Used by **V*IST*A** HL7 as the default value to place in the Processing ID field of outgoing message headers, and the value against which to validate incoming message headers. This can prevent inappropriate messages from being processed. Set appropriately depending on what type of account this is. **V*IST*A** HL7 won't generate new messages to send out if this is left null.
Default Number of Incoming Filers	The number of incoming filers to start with the "Default Filers Startup" option.
Default Number of Outgoing Filers	This is the default number of outgoing filers started by the "Default Filers Startup" option.
Mail Group for Alerts	Enter a mail group whose members are local IRM staff responsible for monitoring **V*IST*A** HL7. This group is used for delivery of alerts or notification of significant events related to Messaging System operations.
Days to Keep Completed Messages	The number of days "Successfully Completed" messages are retained before purging. If no value is entered, the default is 7 days. Used as the actual value when the "Purge Messages" option is scheduled through TaskMan, and as the default when run interactively.
Days to Keep Awaiting ACK Messages	The number of days "Awaiting Application Acknowledgment" messages are retained before they are purged. If no value is entered, the default is 30 days. Used as the actual value when the "Purge Messages" is scheduled through TaskMan, and as the default when run interactively.
Days Before Purging All Messages	The number of days to retain messages before purging (except for ERROR status messages). Defaults to 90 days. The value set in this field is used as the actual value when the "Purge Messages" option is scheduled through TaskMan, and as the default when run interactively.
Default Retention Time	Used by the TCP Link Manager. The maximum time, in seconds, in which the process supporting a non-persistent outgoing TCP link will idle, waiting for new messages to deliver, before terminating. If the Retention field of the link is set to a value, it overrides the site parameter setting.

3.8 Purging

3.8.1 Purging Messages

Messages that have been sent *from* your system and received *on* your system are both retained in your system until you purge them. Use the "Purge Messages" option to do this:

- **Interactively:** This allows you to interactively answer questions about what purging time periods to use. You can run the resulting purge in the foreground or background.
- **As a scheduled task:** Schedule the option to run through TaskMan. When tasked, the option purges based on the time periods specified in the HL7 site parameters. We recommend queuing the purge daily.

The Purge Messages option allows you to purge selectively based on certain message statuses:

Site Parameter	Affects Messages with Status:	Days to Purge Messages (Up to and Including)
Days to Keep Completed Messages	Successfully Transmitted	Defaults to Today minus Site Parameter (if site parameter is undefined, default is T-7.)
Days to Keep Awaiting ACK Messages	Awaiting Acknowledgment	Defaults to Today minus Site Parameter (if site parameter is undefined, default is T-30.)
Days Before Purging All Messages	Any status except Error	Defaults to Today minus Site Parameter (if site parameter is undefined, default is T-90.)
Days Before Purging All Messages	Error	Defaults to the same purging period as for "Any status except Error". But, you can only purge error status messages *interactively*.

Purging Example (Interactive)

```
Enter inclusive date up to which to purge SUCCESSFULLY COMPLETED
messages:  T-20// <RET>   (MAY 29, 1999)

Enter inclusive date up to which to purge AWAITING ACK
messages:  T-30// <RET>  (MAY 19, 1999)

Enter inclusive date up to which to purge all messages, regardless
of status (except for ERROR status):  T-85// <RET>  (MAR 25, 1999)

Do you also want to purge messages with an ERROR status?  NO// YES <RET>

  WARNING: You should have investigated all errors because purging
           these messages permanently removes them from the system.

  Enter inclusive date up to which to purge ERROR
  messages:  T-85// <RET>  (MAR 25, 1999)

Would you like to queue this purge?  YES// <RET>
```

Warning: Review all messages with an error status before purging them. In most cases, you should fix the error that caused the problem and then, if appropriate, retransmit or reprocess the message. In order to purge messages with an *error status*, you need to run the Purge Messages option interactively.

3.8.2 Globals Impacted by V*IST*A HL7

Applications using **V*IST*A** HL7 can generate thousands of HL7 messages. If there is a blockage, these messages can accumulate and consume large amounts of disk space, potentially filling system storage areas and resulting in "disk full" errors. Globals that can be impacted by **V*IST*A** HL7 usage in these situations are:

Global	Remedy
^HL	File #772 (HL Message Text) is stored in ^HL. Use the "Purge Messages" option to clean up this file.
^HLCS	Contains the HL Logical Link file. Disk space is consumed primarily by the "In" queues for HLLP and X3.28 links, and the "Out" queues for HLLP, X3.28 and MailMan links. First, make sure the queues and filers are up and running, and that messages are being delivered in and out of the link. Then, use each link's "Queue Size" setting to restrict the number of messages retained in the queues. Finally, you can use the "Clear a Queue of All Entries" option to purge queues, but with this warning: Do not use "Clear a Queue of All Entries" if there are messages received that haven't been processed, or messages to send that haven't been delivered. For more information, please see the "Clear a Queue of All Entries Option" section in the "**V*IST*A** HL7 Troubleshooting Options" chapter.
^HLMA	File #773 (HL Message Administration) is stored in ^HLMA. Use the "Purge Messages" option to clean up this file.
^TMP(HL,)	For V. 1.5 interfaces, if transmission logging is enabled, this global will be impacted. Use the "Start/Stop Log of HL7 Transmissions" to disable logging. It also asks if you want to purge existing entries; choose YES to purge existing logging.
^XMB	For **V*IST*A** HL7 V. 1.6, the only impact on ^XMB is for storage of incoming and outgoing messages over MailMan links. Use MailMan's purging features to purge ^XMB.

3.8.3 Moving HL7 Globals to Different Volume Set

DSM sites may occasionally need to move globals to different volume sets. This must be done carefully in the case of HL7 globals. You must ensure HL7 activity has ceased and will not recur, before moving globals. Even with filers and links shut down, HL7 messages can still be delivered over "same system" interfaces, if an event triggers message delivery. So, to ensure no messages

will be created or delivered, the entire system should be "down": no users signed on, no TaskMan running, no tasks running, no mail being received, no HL7 links running and no HL7 filers running. In short, there should be no way that HL7 messages can be generated or received. Then it will be safe to move the globals.

3.8.4 Significant Operational V*IST*A HL7 Files

File Name	#	Data Source	Description
HL7 Application Parameter	771	Interface developer	Stores HL7 application definitions. Entries must be unique by application name. Each protocol that is part of an application points to the application entry in this file.
Protocol	101	Interface developer	Each HL7 entry in the Protocol file represents one side of a transaction, either message sender (event driver protocol) or message receiver (subscriber protocol). HL7 protocols use a completely different set of fields (numbered from 770 to 775) versus the fields used for Order/Entry functionality in the same file. Each subscriber protocol points to a single entry in file #870 (HL Logical Link), determining the destination a message for that transaction is delivered to.
HL Logical Link	870	**V*IST*A** HL7, Interface Developer	Entries in this file define the target systems with which this system can exchange HL7 messages. Akin to the Kernel's Device file and MailMan's Domain file, this file contains the specific HLLP, X3.28, TCP, and MailMan information needed by **V*IST*A** HL7 to reach destination systems. **V*IST*A** HL7 also uses some fields in this file internally to manage the incoming and outgoing message queues associated with a given link.
HL7 Message Text	772	**V*IST*A** Applications, and interfaced systems	Together with File #773 (see below), this file stores HL7 messages, both incoming (received from another system) and outgoing (generated on your system), including message header, text, and sending and receiving applications.
HL7 Message Administration	773	**V*IST*A** Applications, and interfaced systems	See File #772 description above.
HL Communication Server Parameters	869.3	Site Manager	Stores the site parameters for **V*IST*A** HL7.

CHAPTER

4. Troubleshooting: Solving Transmission Problems

HL7 messaging is designed to exchange messages between heterogeneous applications running on disparate systems. Because of this, an inherent task in managing HL7 interfaces is troubleshooting the variety of errors that will occur over these interfaces. Errors can occur in any one of several different layers of messaging functionality within any given HL7 transaction.

This chapter lists the most common problems that may occur. It is drawn largely from three years of NOIS (National Online Information System) calls drawn from the experiences of National **V***IST***A** Support (NVS).

4.1 Filer Problems

Symptom	Recommended Response

Filers do not appear to be running. The time difference between "now" and the last filer cycle (Monitor, Start, Stop Filers screen) is large. Attempted to restart the filers, but they did not restart.	First try the following steps:
	1. Stop the existing filers.
	2. Delete the existing filers, through the Monitor, Start, Stop Filers option.
	3. Restart the filers.
	Sometimes this still doesn't get the filers running. It may be that the task numbers for filers recorded by **V***IST***A** HL7 no longer exist on the system. Use VA FileMan to edit the HL Communications Server Parameter file, and delete all task number entries in the "Incoming Filer Task Number" and "Outgoing Filer Task Number" multiples:
	`Select HL COMMUNICATION SERVER PARAMETERS ONE: 1`
	`ONE: 1//`
	`DEFAULT NUMBER INCOMING FILERS: 1//`
	`DEFAULT NUMBER OUTGOING FILERS: 1//`
	`Select INCOMING FILER TASK NUMBER: 5031578// @`
	`SURE YOU WANT TO DELETE THE ENTIRE '5031578'`
	`INCOMING FILER TASK`
	`NUMBER? Y (Yes)`
	`Select INCOMING FILER TASK NUMBER: ?`
	`Answer with INCOMING FILER TASK NUMBER`
	`You may enter a new INCOMING FILER TASK NUMBER, if you wish`
	`Enter the task number of an incoming filer`
	`Select INCOMING FILER TASK NUMBER:`
	`Select OUTGOING FILER TASK NUMBER: 5031580// @`
	`SURE YOU WANT TO DELETE THE ENTIRE '5031580'`
	`OUTGOING FILER TASK NUMBER? Y (Yes)`
	Now try restarting the filers.

Filer Problems (continued)

Symptom	Recommended Response

Cannot get rid of bogus filers from "Monitor, Start, Stop Filers" screen ("delete" option does not remove them). The task number of each bogus filer 0.	If an active filer is deleted (by FORCEX or a shutdown), bad cross-references may be left behind.
	1. Stop all filers.
	2. Look in the outgoing (or incoming) filer multiple in File #869.3 under the Incoming Filer Task Number and Outgoing Filer Task Number multiples.
	3. If no task numbers are listed, look at the "B" cross reference of the "Incoming Filer Task Number" and "Outgoing Filer Task Number" multiples.
	Incoming Filer Task Number:
	`^HLCS(869.3,1,2,"B",IEN)`
	Outgoing Filer Task Number:
	`^HLCS(869.3,1,3,"B",IEN)`
	4. Examine each entry in the cross-reference. For either multiple, if any IEN in the "B" cross-reference refers to a non-existent entry in the multiple, kill it.

Many filers in the "Monitor, Start, Stop Filers" screen with a status of "undefined" whenever the system is shut down and brought back up: "** Incoming filer is no longer defined **" Or "** Outgoing filer is no longer defined **"	Make sure that when you shut down your system, you shut down **V*IST*A** HL7. For the procedure to do this, please see the "Managing a **V*IST*A** HL7 System" chapter. To eliminate bogus filer entries in the "Monitor, Stop, Start Filers" screen, use the "Delete Filer" options to remove the undefined filers from the "Monitor, Start, Stop Filers" screen. If the system crashed or otherwise shut down unexpectedly, you may need to do this as well.

4.2 Link Problems

Symptom	Recommended Response
A link is not starting up.	Look at the particular link's Task Number field in File #870. If a task number is there, use the TaskMan option in the Kernel Toolbox and look at the status of task. If the status is "Waiting for a submanager", TaskMan may be backed up. If the status is "Waiting for a partition", the system may be at its capacity. Also, make sure TaskMan has not been placed in a wait state.
A link has a state of "OpenFail" on the Systems Link Monitor.	For HLLP and X3.28 links, a state of "OpenFail" indicates failure to link's device, typically because some other process owns the device, or the terminal server needs the port logged. Use normal device troubleshooting procedures to ensure the device is available and can be opened by %ZIS. For TCP links, a state of "OpenFail" indicates failure to connect to the target system over TCP/IP. If the target system is **V*IST*A** HL7 at another VA site, try the "Ping" option. If you can't ping the site, the site may be down, or their listener may not be running. For non-VA systems, try to determine if the target system is up and accepting connections.
A link is blinking in the Systems Link Monitor.	This is caused by the most recent communication error that occurred over the link, stored in the Gross Communications Error field of the link. View error with the "Show Communications Error" option; clear the error with the "Clear Communication Error" option to stop the blinking. If the error recurs, you will need to take corrective action. Some of the possible errors are: • LLP Could not Enqueue the Message • LLP Exceeded Retry Param • LLP Timed Out While Reading in a Message • LLP Unable to Dequeue Message • Message Low Level NAK'd by Other System • Message did not Validate, Look at Checksums
No messages are moving over a TCP Listener's link. The listener type is single-threaded.	Check for other TCP listeners (e.g., the Broker TCP listener) running on the same port. On a VMS system, do this by executing the UCX "show device" command. Only one listener can listen on any given port.
Link is constantly in a WRITING status. No messages are transmitting.	Stop the link and then restart it.

Link Problems (continued)

Symptom	Recommended Response
Link has Read, Write or Disconnect error state.	For HLLP and X3.28 links, if a connection is dropped, the links goes into a disconnect error state. HLLP links trap the error and attempt to reopen the link. For TCP links, if a connection is dropped, the link has a read or write error briefly, but clears itself up and will attempt to reopen the link.
The HLLP or X3.28 link is in a constant state of "Reading Ack".	HLLP and X3.28 links are susceptible to all of the device problems any other device is susceptible to, including the device's VMS Terminal Characteristics and LATCP definition, problems on the target system, the terminal server port being XOFFed, the device definition in SYSGEN, and line failure. Check each of these that apply for your system to ensure that the device setup is correct.
A COTS system with an interface to **V*IST*A** HL7 over a TCP link appears to be in a loop reading the same message.	Sometimes HL7 applications lose synchronization. This is probably what has happened. For more information, see "Resynchronizing TCP and HLLP Links" below.
For TCP links, received the following alert: "HL7 LL *name* exceeded retries. LL will shutdown [or restart, or keep trying]."	See the "Alerts Posted to the **V*IST*A** HL7 Mail Group for Alerts" section of this chapter. The link may have lost synchronization.
No messages are coming in over a MailMan link.	Check the server option's mail basket under the Postmaster. If recent incoming HL7 mail messages exist in the mail basket, the server option is probably not running. Check that the mail group associated with the HL V16 SERVER option (SERVER MAIL GROUP field) has at least one active, local user in it. If the server mail group is OK, check if the HL V16 SERVER option has a resource device attached. If it does, use the Monitor TaskMan option, look at the resource device, and see if it has tasks waiting. If it does, you may need to use the Kernel "Clear One Resource" option to clear the resource device.

Link Problems (continued)

Symptom	Recommended Response
Messages received over HLLP or X3.28 links are not being processed. Or, messages are not moving out over HLLP, X3.28 or MailMan links.	A valid, working link should have "Messages Sent" approaching and eventually equaling "Messages To Send" for outgoing messages. It should have "Messages Processed" approaching and eventually equaling "Messages Received" for incoming. If there is a backlog of messages to send or process, and the destination system is up and running: 1. Check that the destination (or source) system is up and running. If it is, 2. Shut down the link and restart it. If this doesn't work, 3. Shut down the appropriate (incoming or outgoing) filers, shut down the link, restart the filer, and restart the link. If this doesn't work, 4. Follow the procedure for looking for stub records in the appropriate queue (see "Stub Records and Corrupted Pointers in Link Queues" below). If this doesn't work, 5. Follow the procedure for looking for looking for corrupted queue pointers in the appropriate queue (see "Stub Records and Corrupted Pointers in Link Queues" below). If this doesn't work, 6. For "Out" queue problems, follow the procedure for looking for corrupted "A-XMIT-OUT" cross-reference entries (see "Stub Records and Corrupted Pointers in Link Queues" below). If this doesn't work, 7. For HLLP and X3.28 links, try starting the link in foreground. If you get a timeout (HLTRANS=TIMEOUT is displayed in the output in foreground mode), look for problems with the device, its cabling, and so forth.

Link Problems (continued)

Symptom	Recommended Response
Even though messages are being generated for an interface, the number in the link's "Messages to Send" column of the Systems Link Monitor for a link (HLLP, X3.28 or MailMan) is not incrementing.	First, check that at least one outgoing filer is running. Then, check if the event driver protocol has a link defined. Use the Protocol Edit option. If the link has been deleted, add it back in. Otherwise, there may be a problem with the link's "Out" queue multiple. There have been cases where the "Out" queue multiple's data structure was corrupted, and the outgoing filer could not add messages to the queue. To solve this problem: 1. Make sure there are no unsent messages in the "Out" queue that need to be sent. In the Systems Link Monitor, the number of messages in the "Messages To Send" column should be equal to that of the "Messages Sent" column. 2. Use the "Clear a Queue of All Entries" option to clear the link's "Out" queue. This purges the queue and all of its subfile data structures. All that is left is the zero node. Once cleared, the filer should be able to add messages to the queue correctly.

4.3 Postmaster Receiving HL7 Error Notifications

The Postmaster may receive a message that says, "Unable to transmit HL7 message due to the following Application Error: …" This means that an outgoing MailMan link encountered an error trying to send the message. Currently, this message is generated if the link does not have a valid mail group with recipients.

4.4 Alerts Posted to the V*IST*A HL7 Mail Group for Alerts

The following alerts will be sent by **V*IST*A** HL7 to the mail group specified in the "Mail Group for Alerts" site parameter, when certain conditions occur.

Alert Text	Meaning/Recommended Response
HL7 Logical Link NNN shutdown due to TaskMan unable to process task request.	The TCP Link Manager tasked a job to start a non-persistent link, but after 30 minutes, the link has still not started. Something is probably wrong with TaskMan. Check that TaskMan is running, is not in a WAIT state, and has not fallen critically behind.
"HL7 LL *name* exceeded retries. LL will shutdown [or restart, or keep trying]."	A TCP link is attempting to send a message, and has retried sending it up to the limit in the link's Retransmission Attempts setting. But the target link has not returned the expected response. This occurs when: a) Your message is rejected as a bad message, b) Your message takes longer to process than the link's ACK TIMEOUT setting, or c) There is a synchronization problem. First, make sure the link's "Retransmission Attempts" field is set to a positive number, e.g., 3, but not 0. To see if your message is being rejected as bad, use the View Transmission Log option; choose "Error Listing". Look at errors for the link (error status). If you do find a message with an error status, troubleshoot what's wrong, typically looking at the application generating the message. If there is no message error, the link is either not waiting long enough for the ack to be returned, or there is a synchronization problem with the link. To see if there is a synchronization problem, see "TCP Link Synchronization problems". Otherwise, increase the ACK TIMEOUT. Depending on the network and application, 45 seconds is reasonable for some interfaces.

4.5 Stub Records and Corrupted Pointers in Link Queues

For HLLP, X3.28 and outgoing MailMan links only, under certain situations, there is the possibility of stub records left in links' "In" and "Out" queues, and corrupted queue "front" and "back" pointers. This can happen under the following circumstances:

- The system shuts down unexpectedly
- There is a "clobber" routine error when installing an HL7 patch

Stub records and corrupted queue pointers may cause links to be hung up and unable to process incoming and/or outgoing messages.

4.5.1 Finding and Fixing Stub Records in a Link's "In" or "Out" Queue

A stub record is any message entry in a link's "In" or "Out" queue that has a .01 field, a status of "Stub Record", and a missing or incomplete message body. If incoming and/or outgoing message transmission for a link is hung up, one thing to look for is whether there are any stub records in the queue.

The main ways to get a stub record in the "Out" queue are:

- A clobber error during patch installs if HL7 left running
- An abnormal shutdown (system crash, or shutting down system without shutting down HL7 first).

To see if there are stub records:

1. Do a global listing of the queue that is stuck. "IN" queues are descendant from ^HLCS(870,Link IEN,1). "OUT" queues are descendant from ^HLCS(870,Link IEN,2).
2. Stub records have an "S" as the value of the second piece of the queue entry's 0 node:

```
^HLCS(870,1,2,14070,0) = 14070^S
```

The basic strategy to clean up stub records is to find them, set their status to "Done", and edit the queue's front pointer to bypass them.

To clean up stub records:

1. Stop the link in question.
2. For "In" queue problems, stop all incoming filers. For "Out" queue problems, stop all outgoing filers.
3. Using VA FileMan, edit the link in File #870. Under the appropriate multiple in the link ("In Queue" or "Out Queue"), edit the queue entry for each stub record found (choose the Message Number equal to the IEN of the stub record). For each "stub" record, change its status from "Stub Record" to "Done". For example, in the following listing, the record you would edit is message #14070:

```
^HLCS(870,1,2,14067,0) = 14067^D
^HLCS(870,1,2,14068,0) = 14068^D
^HLCS(870,1,2,14069,0) = 14069^D
^HLCS(870,1,2,14070,0) = 14070^S
^HLCS(870,1,2,14071,0) = 14071^P
^HLCS(870,1,2,14072,0) = 14072^P
^HLCS(870,1,2,14073,0) = 14073^P
^HLCS(870,1,2,14074,0) = 14074^P
```

4. Using VA FileMan, edit the link in File #870, and edit the front pointer field of the appropriate queue (either "In Queue Front Pointer" or "Out Queue Front Pointer"). Its value should be one less than the IEN of the next "Pending" message in the queue. In the example above, suppose that the "In Queue Front Pointer" was left at 14069. You've set the status of message #14070 to "Done". However, the next "Pending" message entry is #14071. Set the front queue pointer to one less than the IEN of the next pending message, i.e., set it to 14070. For more information on the correct values for queue pointers, see "Finding and Fixing Corrupted Queue Pointers" below.
5. Restart the link and the filers. The link should start processing messages again.

4.5.2 Finding and Fixing Corrupted Queue Pointers

If incoming and/or outgoing message transmissions for a link are hung up, but there are no stub records in the queue, another thing to look for is whether the "In" or "Out" queue's front or back pointer is corrupted.

The values for a queue's front and back pointers, for both "In" and "Out" queues, should be as follows:

Pointer	Value
Front	Set to 1 less than the message IEN of the next "Pending" status message. Or, if there are no "Pending" status messages, set equal to the last (highest numbered) IEN in the queue.
Back	Set equal to the last (highest numbered) IEN in the queue.

To fix a queue's front or back pointer:

1. Stop the link in question.
2. For "In" queue problems, stop all incoming filers. For "Out" queue problems, stop all outgoing filers.
3. Do a global listing of the queue that is stuck. "IN" queues are descendant from ^HLCS(870,Link IEN,1). "OUT" queues are descendant from ^HLCS(870,Link IEN,2).
4. Using VA FileMan, edit the link in File #870, and edit the appropriate queue's front or back pointer field. Set it to the appropriate value based on the table above.
5. Restart the link and the filers. The link should start processing messages again.

4.5.3 Finding and Fixing Corrupted "A-XMIT-OUT" Cross References

For *outgoing* HLLP, X3.28 and MailMan queues only, if there are no stub records, and if the queue's front and back pointers are correct, but the queue is still stuck, one additional corruption to look for is in File #772's "A-XMIT-OUT" cross reference. The "A-XMIT-OUT" cross-reference is used by the outbound filer to move messages to a link's "Out" queue.

To find and fix corruptions in File #772's "A-XMIT-OUT" cross-reference:

1. Stop the link in question.
2. Stop all outgoing filers.
3. Loop through File #772's "A-XMIT-OUT" cross reference for the link in question:

   ```
   ^HL(772,"A-XMIT-OUT",Link IEN,Message IEN)
   ```

   Link IEN is the IEN in File #870 of the link in question.
4. For each entry for the link in the cross-reference, check for:

   ```
   $D(^HL(772, Message IEN, 0))
   ```
5. If the cross-referenced message is not present in File #772, kill the bad cross-reference node in the "A-XMIT-OUT" cross-reference.
6. Restart the link and filers.

4.6 Resynchronizing TCP and HLLP Links

Communication between systems via HL7 requires synchronization between original messages sent out and responses coming back:

- When links are synchronized, each system is able to match up each response coming back with the message it originated, thereby completing the requested transaction.
- When links lose synchronization, the originating system receives the wrong message back, and rejects the message. The other system keeps trying to resend the message. The interface is hung until the systems are able to regain synchronization.

4.6.1 Determining if a Loss of Synchronization Has Occurred

The main symptom on the **V*IST*A** HL7 side when a link loses synchronization is that the link exceeds its "Re-Transmission Attempts" setting. TCP and HLLP have this setting; X3.28 and MailMan links do not. This does not always mean that a loss of synchronization has occurred. **V*IST*A** HL7 attempts to retransmit an original message if it encounters any of the following situations when it attempts to receive and process the response:

- **Error in Header:** The expected response had an error in header, and **V*IST*A** HL7 NAK'd the acknowledgment.
- **Ack Timeout: V*IST*A** HL7 didn't get an expected response within the Ack Timeout time limit. In this case, the target system either is down or is not able to respond within the link's time out. This can usually be corrected by increasing the link's Ack Timeout setting.
- **Not an Ack:** The expected response was not an acknowledgment (status=error, error type="incorrect message received", error message = same). This is a loss of synchronization.
- **Different Transaction:** The expected response was for a different transaction (status=error, error type=wrong message id, error message=same). This is a loss of synchronization.

If a link has exceeded its retransmission attempts, you need to determine whether it is a synchronization loss. The method for doing this is different depending on whether the link is a TCP or HLLP link.

4.6.2 Automatic Resynchronization of TCP Links

V*IST*A HL7 has several self-correcting strategies that it uses to maintain synchronization for TCP links:

- When receiving a message, **V*IST*A** HL7 now looks for duplicates. If the message has already been received, **V*IST*A** HL7 does not reprocess the message a second time; it marks it with an error status of "Duplicate" and re-sends the acknowledgment again that had been sent for the original message.
- You can use a new link setting, "Exceed Attempts Action", to tell **V*IST*A** HL7 to perform an action when a link exceeds retransmission attempts (a sign of synchronization loss). If you set this action to "Restart" for a link, whenever the link exceeds retransmission attempts, **V*IST*A** will automatically the link and restart it, to flush the message buffers and in most cases regain synchronization.

If you exceed the re-transmission attempts for TCP links (the main sign that a loss of synchronization may have occurred), you'll get an alert:

```
HL7 Message ien NNN exceeded retries. LL NNN was SHUTDOWN.
```

4.6.3 When TCP Links Lose Synchronization

To determine if there was a loss of synchronization:

1. Make sure the "Re-Transmission Attempts" field is set to a positive number. We recommend a setting of three in most cases.
2. Use the View Transmission Log option. Choose "Error Listing". Look at errors for the link.
3. If you find message(s) with an error status of "Application Reject Error", the problems it that the messages you are receiving have an incorrect header and **V*IST*A** HL7 is NAK'ing them.
4. If you find messages with an error other than "Application Reject Error" or "Incorrect Message Response", the problem is probably *not* a loss of synchronization. Troubleshoot what's wrong by looking at the application generating the message.
5. If you find message(s) with an error status of "Incorrect Message Response", you've probably lost synchronization.
6. To determine if an Ack timeout is the problem (in which case the link's Ack Timeout setting should be increased), use the "IEN Record #" column value from one of the records from the View Transmission Log option with an error status of "Incorrect Message Response". With that IEN, look at the following global trace:

   ```
   ^HLMA("AF", IEN Record #, inbound response)
   ```

   There should always be at least one entry for an inbound response (same IEN as the outbound message). But, if there is more than one inbound response beyond the original node, you've lost synchronization (you're getting wrong responses back). If there are no responses beyond initial, problem is timeout.

To correct the problem causing loss of synchronization:

1. If the problem appears to be an Ack Timeout, try bumping up the ACK Timeout setting for the link. This field should be set carefully depending on the amount of time it may take for an application to process a message and return an acknowledgment.
2. If increasing the ACK Timeout doesn't help, set the "Exceed Re-Transmit Action" setting for the link to "Restart". If the link exceeds its "Re-Transmit Attempts" limit, **V*IST*A** HL7 shuts the link down automatically and brings it back up. This often regains synchronization without requiring intervention.

3. If the link does not resynchronize automatically, and the other system is not a **V*IST*A** HL7 system, then you may need to use a manual procedure to regain synchronization. **Warning:** Do the following manual procedure only if you know both sides are "caught up" because this *destroys* any messages still present either system's queues:
    a. Inactivate the **V*IST*A** HL7 application that is sending or receiving message (use the "Application Edit" option).
    b. Stop the link.
    c. Kill the ^HLMA("AC","O",ifn) global where "O" is the letter O, and ifn is the number of the link from File #870.
    d. Clear the queue of the link.
    e. Have the vendor machine clear their queues of all messages.
    f. Start the link.
    g. Reactivate the application.
    h. Restart the vendor application.

4.6.4 Resynchronizing HLLP Links

HLLP links can lose synchronization for several reasons:

- Sometimes an acknowledgment gets lost over the network (serial connections can be susceptible to this).
- Sometimes a response takes too long, the original message is resent, the first response comes back, but a second response has been initiated, and so forth. This slows down subsequent transmissions until both sides catch up.
- If **V*IST*A** HL7 and the COTS system can both originate messages to each other, sometimes one system originates a new message in the moment it should also return an acknowledgment for a different message. In this case, the sequence of message exchange loses synchronization.
- Sometimes the system vendor doesn't follow the HL7 standard.

When an HLLP link has lost synchronization; the error "LLP Exceeded Retry Param" is set into Gross Communications Error field of the link. To determine if synchronization is being lost:

1. Use the "Link Error/Status Report (non-TCP)" option on the link in question.
2. If messages that went out over the link have an error of "Application Reject Error", **V*IST*A** HL7 is NAK'ing the response because there is a problem in the incoming message's header. Work with the target system to correct the error in the message headers.
3. If messages that went out over the link have an error of "Incorrect Message Response", the link is probably losing synchronization.

To clear the input and output buffers on both sides of the interface and regain synchronization:

1. On the other system, shut down HL7 messaging to **V*IST*A**.
2. On **V*IST*A**, shut down the link to the other system.
3. Then start up messaging on both sides of the interface again.

If both systems in an HLLP interface can originate new messages to the other, we recommend having two serial lines for the interface, rather than one:

- Messages originating from **V*IST*A** HL7 should always go over one line (and responses come will back over the same line).
- Messages originating from the other system should always go over the other line (as will the responses **V*IST*A** HL7 sends back).

This prevents a new message from "getting ahead" of a response in either queue, lessening the chances for synchronization loss for HLLP.

CHAPTER

5. V*IST*A HL7 Troubleshooting Options

5.1 TCP vs. Non-TCP Options

At the time of writing, **V***IST***A** HL7 stores messages in two different formats:

- The original format (from the original, unpatched V. 1.6 **V***IST***A** HL7)
- A newer format introduced in patch HL*1.6*19, for messages sent over TCP links only.

The newer format, which takes advantage of new, simplified file structures in **V***IST***A** HL7, is used only for messages sent over TCP links. It enables greater performance efficiency by effectively halving the number of disk writes needed to both send and receive messages.

The original message storage format is still used for messages delivered over non-TCP links, however. As a result, many of the options to manage and troubleshoot message transmissions are divided into two categories: TCP and non-TCP.

5.2 V*IST*A HL7 Message Statuses

Status	Explanation
Awaiting Application Acknowledgment	The outgoing message has been sent to its destination, but an expected acknowledgment has not been returned yet from the destination system.
Awaiting Processing	The incoming message has been received and is waiting to be processed.
Being Generated	The outgoing message is in the process of being created.
Error	The message has encountered an error condition, during either sending or receiving. For more information, check the specific error type and error message.
Pending Transmission	The outgoing message is awaiting transmission.
Successfully Completed	For outgoing messages, the message has been sent to its destination, and any expected acknowledgments have been received. For incoming messages, it was received, processed, and any required acknowledgment was returned to the sending system.

5.3 Ping (TCP Only)

The Ping option lets you ping another **V*IST*A** HL7 system (the target system must have installed patch HL*1.6*57 for this feature to be enabled.) This option is helpful when troubleshooting interfaces; it can eliminate the network and the listener as the source of the problem, or point to one or the other as the source of the problem.

Ping only works over TCP links to another VA site running **V*IST*A** HL7. You can run it without any problems to both production and test systems; the only effect is that the target system returns success if it is up.

Another possible use of Ping is if you can't connect from a COTS system at your site to your **V*IST*A** HL7 system. While you can't ping to the COTS system, you could have another **V*IST*A** HL7 system Ping your system. That will at least tell you whether your listener is actually running.

A successful Ping looks like:

```
Select Filer and Link Management Options Option: Ping (TCP Only)
Select HL LOGICAL LINK NODE: REDACTED
PING worked
```

An unsuccessful Ping looks like:

```
Select Filer and Link Management Options Option:  Ping (TCP Only)
Select HL LOGICAL LINK NODE: VAZZZ
-1^Inital Connection Failed
```

The possible failure codes returned by Ping are:

Failure Code	Explanation
-1^Inital Connection Failed	Network path is down, or the other system does not have a listener running.
-1^No response	**V*IST*A** HL7 was able to get a connection, but failed to get a response. In this case, a listener is running on the target system, but something else is wrong: the listener is a foreign listener (e.g., the RPC Broker), or the link entry for the listener on the receiving system is not defined correctly.
-`^Incorrect response	You may see this response if you run the Ping option against a non-**V*IST*A** HL7 system.

5.4 View Transmission Log (TCP Only)

Use the "View Transmission Log (TCP only)" option to:

- Search completed HL7 message transmissions (both incoming and outgoing).
- Monitor *pending* HL7 message transmissions. *Messages should remain in a PENDING TRANSMISSION status for only a short time (i.e., an hour or less).*
- Monitor outgoing HL7 message transmissions with an error status.

This option tracks messages transmitted over *TCP links only*.

5.4.1 Filtering Which Messages to View

The "View Transmission Log" option lets you select a subset of messages to examine before generating a list of messages and putting that list into the viewer. It presents a dialog screen as follows:

```
                          Search Transmission Log


     Select one of the following:

          M         Message Search
          P         Pending Transmissions
          E         Error Listing
          Q         Quit (also uparrow, or <RETURN>)


Selection:
```

Each of the three report choices above lets you further refine the set of messages to view. The additional (and optional) filtering criteria, by report choice, are:

Report Choice	Further Filter By
Message Search	• Date Range • Message Status (one status or all; please see "**V***IST***A** HL7 Messages Statuses" elsewhere in this chapter) • Link (one Link or all) • Message Type (one message type or all) • Event Type (one event type or all)
Pending Transmissions	• Link
Error Listing	• Error Type

5.4.2 Viewing Message Summaries of All Matching Messages

Once you've chosen your selection criteria, the View Transmission Log option displays summaries of all matching messages, in VA FileMan's Browser. Outgoing messages are sorted by the date/time that the message was transmitted; incoming messages are sorted by the date/time that the receiving application processed the message. The following information is displayed for each message:

Column	Description
IEN Record #	IEN of message in File #773.
Message ID#	Unique Message ID number assigned by **V*IST*A** HL7 to the message.
Log Link	Link on which the message is to be, or was transmitted.
Msg:Evn	Message Type, Event Type pair indicating the nature of the message.
IO	Indicates whether the message is incoming, 'IN', or Outgoing, 'OT".
Sndg Apl	Application that is sending the message.
Rcvr Apl	Application to which the message is directed.
HDR	Message header. Use the right arrow key to view continuation of header.
ERR	For the Error listing only; replaces the HDR shown in the other listing types. Displays the message entry's Error Message and Error Type fields.

Display of all Matching Messages:

```
IEN RECORD #   MESSAGE ID #          Log Link   Msg:Evn IO Sndg Apl Rcvr Apl HDR
2              160002               RH CLIENT  6:1 OT 34       35        MSH|~^\&
3              160003               RH CLIENT  6:1 OT 34       35        MSH|~^\&
4              160004               RH CLIENT  6:1 OT 34       35        MSH|~^\&
5              160005               RH CLIENT  6:1 OT 34       35        MSH|~^\&
6              160006               RH CLIENT  6:1 OT 34       35        MSH|~^\&
7              160007               RH CLIENT  6:1 OT 34       35        MSH|~^\&
8              160008               RH CLIENT  6:1 OT 34       35        MSH|~^\&
9              160009               RH CLIENT  6:1 OT 34       35        MSH|~^\&
10             1600010              RH CLIENT  6:1 OT 34       35        MSH|~^\&




Col>   1 |<PF1>H=Help <PF1>E=Exit| Line>     9 of 9      Screen>     1 of 1
```

You can use the right arrow key to scroll right so that you can view the full message header:

```
IEN RECORD #   MESSAGE ID #         Log Link   Msg:Evn IO Sndg Apl Rcvr Apl HDR
R RG CIRN  MSH^~&\^1600^MPI-STARTUP^NXT^RG CIRN^19980716075643^^ADT~A01^4^P^2.2^
  MPI-STAR MSH^~&\^NXT^RG CIRN^1600^MPI STARTUP^19980716075750^^ACK~A01^4^P^2.2^
R RG CIRN  MSH^~&\^1600^MPI-STARTUP^NXT^RG CIRN^19980716075643^^ADT~A01^4^P^2.2^
  MPI-STAR MSH^~&\^NXT^RG CIRN^1600^MPI STARTUP^19980716075750^^ACK~A01^4^P^2.2^
Col>  66  |<PF1>H=Help <PF1>E=Exit| Line>      4 of 4      Screen>   1 of 1
```

5.4.3 To View a Message's Text:

While in the Transmission Log option's main screen, you may want to view the full message text for a particular message.

1. Press <PF1>Z while viewing the transmission log.
2. At the "Enter Message ID Number:" prompt, enter the Message ID #. **Note:** Message IDs are unique.
3. The Browser displays the requested message. To view characters beyond the right or left margins, use the arrow keys.
4. To return to the Transmission Log main screen, enter R (for return).

For example, while viewing the transmission log, to view the message text for Message ID # 20 in the HL7 Message Text file, press <PF1>Z and enter the following:

```
Enter Message ID #: 20 <ret>
```

The option then displays the text for that message in a new Browser screen. To return to the previous screen enter R (for return):

```
          HL7 MESSAGE TEXT:[JUL 16,1998@07:57:46] (wp): MESSAGE TEXT
MSA^AA^1
Col>   1  |<PF1>H=Help <PF1>E=Exit| Line>      2 of 2      Screen>   1 of 1
```

5.4.4 Using the VA FileMan Browser

You can use all of the standard features of the VA FileMan Browser to search through the displayed list. Some particular features of use are:

Keystroke	Description
Left and Right Arrow Keys	View text that is beyond either screen margin.
<PF1> Arrow Down	Page Down
<PF1> Arrow Up	Page Up
<PF1>F	Find Text
<PF1>N	Find Next

Documentation on the Browser is provided in the VA FileMan User Manual, and in the Browser's online help (in the Browser, press <PF1>H).

5.5 Awaiting/Pending Transmissions Report (non-TCP)

Use this option to list all messages for transmission over non-TCP Links with a status of PENDING TRANSMISSION or AWAITING ACKNOWLEDGMENT. Use this report to monitor outgoing message transmissions:

- Messages should only remain in a PENDING status for a short time (i.e., an hour or less).
- Messages should only remain in an AWAITING ACKNOWLEDGMENT STATE until the target system can return an acknowledgment.

This report lists messages in chronological order by ID (date/time entered). The report lists some of the following fields for each message, depending on the transmission status of the message:

- Message ID
- Message text
- Date/time transmission entered
- DHCP application (i.e., sending application)
- Non-DHCP application (i.e., receiving application)
- Transmission type
- Related MailMan message number
- Transmission status
- Error message
- Error type
- Number of characters in message
- Number of events in message

5.6 Failed Transmissions Report (non-TCP)

Use this option to list messages transmitted over non-TCP links with a message status (field #20 in File #772) of "ERROR". Use this list to help detect or troubleshoot transmission problems. Output for each message may include the following:

- Message ID
- Message text
- Date/time transmission entered
- DHCP application
- Non-DHCP application
- Transmission type
- Related MailMan message number
- Transmission status
- Error message
- Error type
- Number of characters in message
- Number of events in message

The entries on the output are listed in chronological order by ID (date/time entered).

Example

```
Log of Failed HL7 Transmissions                NOV 25,1994  13:16    PAGE 1
-------------------------------------------------------------------------------

ID: NOV 17, 1994@09:34:51
 MESSAGE TEXT:
 MSH|^~\&|INFOLIO TEST|MTHOME VAMC|IB PERIPH
TEST|500|19900314130405|ORU|123456|D|2.1|
 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
 OBX|ST|VR^RESPIRATORY^L||12|/MIN
 OBX|CE|OR^RETURN VISIT^L||CARDIOLOGY^19930415|
 OBX|CE|OO^ORDER^L||71010^R|
 OBX|CE|OO^ORDER^L||81000^L|
  STATUS: ERROR DURING TRANSMISSION
  DATE/TIME PROCESSED: NOV 17, 1994@09:34:51
  ERROR MESSAGE: Missing Event Type     ERROR TYPE: 13
  NO. OF CHARACTERS IN MESSAGE: 547     NO. OF EVENTS IN MESSAGE: 1
```

5.7 Link Error/Status Report (non-TCP)

Use this option to look at the messages being transmitted over any given non-TCP link, to see which (if any) messages being transmitted over that link are ending up in an error status. The option does this by checking for messages with the Error (#2) field populated in the link's "In" or "Out" queues.

The report prompts you as follows:

1. Select either one particular link, or "ALL".
2. Select the "In" queue, "Out" queue, or both.
3. Select one or more particular error codes for which to include messages in the report, or choose "ALL".
4. Select one or more status codes for which to include messages in the report, or choose "ALL".

The output provides the following information about each message in error:

- Node
- Queue
- Message number
- Message status
- Message error
- Message text

Note: For TCP links, use the "View Transmission Log" option to obtain the same results.

5.8 Clear a Queue of All Entries Option

5.8.1 Resetting a Queue's Counters (TCP Only)

For TCP Links only during normal operations, use the "Clear a Queue of all Entries" option to reset a TCP link's "Message Received" and "Messages Processed" ("In" queue) and "Messages Sent" ("Out" queue) counters to zero. This can be useful, because the Systems Link Monitor's *screened* view only displays links with non-zero numbers in any of the four "message" counter columns. Zeroing the message counters of TCP links means they won't show up in the screened view of the Systems Link Monitor until the next time there is message traffic.

The option prompts you for the name of the link node and which queue(s) -- in and/or out -- to reset.

For TCP links, this option resets the queue counters only. Messages sent and received over TCP links are never stored in links' "In" and "Out" queues, and therefore are not cleared by this option.

Warning: This option should not be used on non-TCP (e.g., HLLP, X3.28 or MailMan) Links for the sole purpose of resetting counters. For non-TCP Links, it also clears out both unsent and unprocessed messages in the queues, without the possibility of recovery.

5.8.2 Clearing All Messages from a Link's Queues (HLLP, X3.28 and MailMan)

When used on the "In" or "Out" queues of **HLLP, X3.28 or MailMan Links**, the "Clear a Queue of all Entries" option does two things:

- Re-initializes the queue's counters to zero
- Clears out all the messages in a link's "In" and "Out" queues

Warning: All messages in HLLP, X3.28 and MailMan links' "In" and "Out" queues, including un-transmitted and unprocessed messages, are deleted by this option and are unrecoverable. Under ordinary circumstances, this option should not be used on these types of links outside of a test environment.

The only circumstances for which you should even *consider* using this option on a production HLLP, X3.28 or MailMan link are if:

1. In the Systems Link Monitor, the number of messages in the "Messages Received" column is equal to that of the "Messages Processed" column.
2. In the Systems Link Monitor, the number of messages in the "Messages Sent" column is equal to that of the "Messages To Send" column.
3. The link is not active, for incoming or outgoing messages.
4. You need to purge the queues for some significant reason, such as large amounts of disk space being consumed by the queue if the queue is not being purged properly, or to eliminate broken subfile structures in the queue.

5.9 Requeuing Messages (Non-TCP)

Warning: *The Message Requeuer (non-TCP) option should ordinarily be used for testing only.* Typically, during testing you may need to send and re-send a particular test message as you test the interface. The Message Requeuer provides an easy way to do this.

The "Message Requeuer (non-TCP)" option allows you to *requeue* (retransmit) HL7 messages that have already been transmitted. You can only requeue messages originally transmitted over non-TCP links.

Messages are not retransmitted directly from the Message files (#772 and #773). They can only be retransmitted from a link's "Out" queue. The number of message retained in an "Out" queue before automatic purging is determined by a link's Queue Size setting. Therefore, a message can be requeued as long as it:

- Has been successfully transmitted already, and
- Remains in a link's "Out" queue. Messages are purged from a link's "Out" queue by the outgoing filer based on the Queue Size setting for the link (if nothing is set, the default is to retain 100 successfully transmitted messages.)

To requeue a message for transmission over a particular Link

1. Select the Message Requeuer (Non-TCP) option. Selecting this option displays the Known Links screen.

```
HL7 Message Requeuer          Sep 18, 1995 13:28:06          Page:    1 of    1
                              Known Logical Links

  Index  Queue Name           Queue Size          Sent Msgs       Pending
  1      TEST                 11                  2               9









+         Enter ?? for more actions

EP  Expand Entry
Select Action: Quit// EP
```

- *Index* - The number to use to select a queue for further action (Expand Entry action).
- *Queue Name* - Name of the link.
- *Queue Size* - Number of messages in the link's "Out" queue.
- *Sent Msgs* - Number of messages with a status of "D" (Done) in the link's "Out" queue.
- *Pending* - Number of messages with a status of "P" (Pending) in the link's "Out" queue.

2. From the 'Known Links" screen, choose "Expand Entry" to select the link for which you want to retransmit one or more messages.

3. "Expand Entry" displays the "Processed Messages in Link" screen, which lists all processed messages for the selected link. From this screen, you can:

   - View any of the processed messages (VM).
   - Select a message for re-transmission (SE) — an asterisk is placed to the left of the message to indicate that it is selected.
   - Deselect a message for re-transmission (DE).

```
HL7 Message Requeuer          Sep 18, 1995 13:29:23          Page:    1 of    1
                    Processed Messages in Logical Link

  Number    Date Processed                    Remote DHCP Application
  1         JUN 22, 1995@14:27:08             ZADT RECEIVE




+         Enter ?? for more actions
SE  Select Messages       DE  Deselect Messages     VM  View Message
Select Item(s): Quit// SE
```

   - *Number* - The number assigned to the message when it is enqueued into the FIFO queue
   - *Date Processed* - The date and time the message was actually processed (sent or received).
   - *Remote DHCP Application* - The application with which you are exchanging information

4. When you choose QUIT to exit the "Processed Messages in Link" screen, you are prompted whether to actually retransmit any messages that you selected. If you answer Yes, a copy of the message is created and added to the "Out" queue for the link to retransmit.

C H A P T E R

6. Managing TCP/IP Listeners

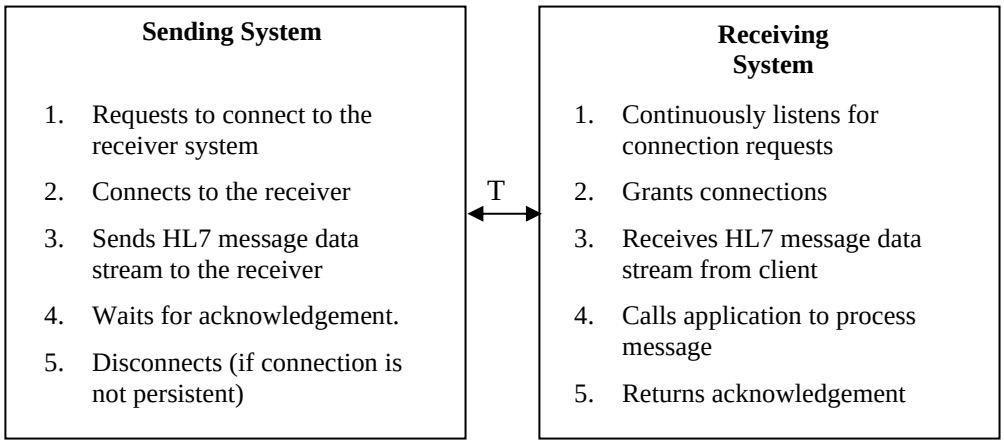
6.1 Introduction

V*IST*A HL7 uses TCP/IP listeners to "listen" on a particular port for incoming TCP/IP connections from other systems. Listeners are necessary whenever another system in an interface needs to initiate a connection **to** the **V*IST*A** system over TCP/IP.

Client and Server Roles in HL7 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)



6.1.1 Requirements for Systems connecting to a V*IST*A System using TCP/IP

If the target system connecting to **V*IST*A** is not a **V*IST*A** system, it must support synchronous bi-directional TCP/IP transactions. This means that when a message is sent over a line, the expected response to that message is returned immediately, over the same open line. The sending system does not initiate a new transaction until the current transaction is completed. The receiving system must respond to the original message without attempting to initiate a new transaction. If **V*IST*A** is to connect to the target system, the target system must have its own TCP/IP listener process that responds to connection requests.

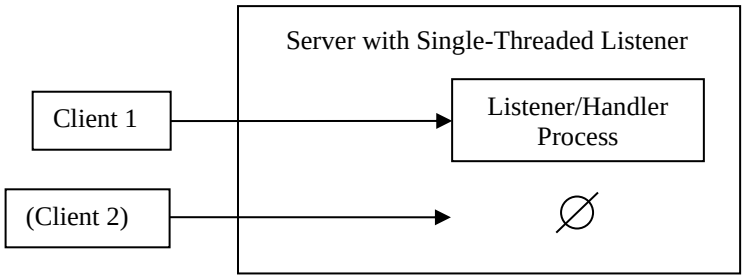
The listener doesn't "turn around" its connection to initiate a new transaction in the opposite direction. If there are messages to *send* from the listener's system to the connected client's system,

the listener's system must itself initiate a new "client" connection in the opposite direction to deliver those messages.

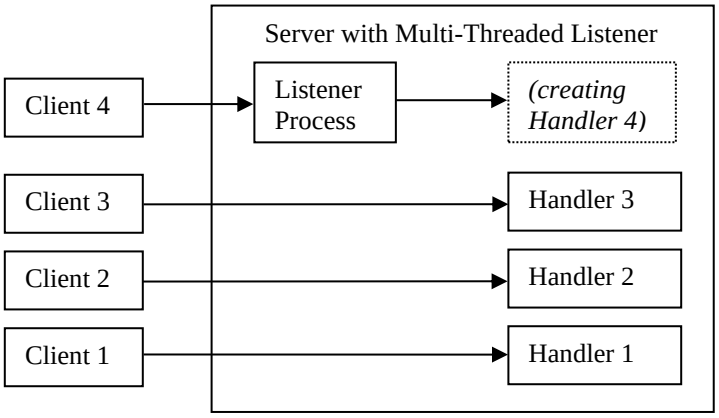
The TCP/IP connection can itself be *persistent* or *non-persistent*. This is determined by the connecting system. If the connecting system drops the connection after a transaction completes, the connection is non-persistent. If the link is left open, the connection is persistent.

6.1.2 Listener Types

- **Single-threaded** listeners are best suited for connecting to a *single, known* device over TCP/IP, such as a single Commercial Off-The-Shelf (COTS) system. **V*IST*A** HL7's single-threaded listener runs in M, waits for connection requests, and handles all processing for each connection directly without spawning off another job. Single-threaded listeners not suited to handle incoming requests from multiple systems; if the listener is busy with one request, other connection requests will fail.



- **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.



V*IST*A HL7 supports two types of multi-threaded listeners:

- Multi-Threaded Listener for Caché on NT
- UCX Multi-Threaded Listener for OpenVMS

6.2 Single-Threaded Listener

Single-threaded listener mode for TCP/IP messaging is available for all currently supported M operating systems:

- Caché on NT
- DSM for OpenVMS

To set up a single-threaded listener:

- Define an entry for the listener in the HL Logical Link file, using the "Link Edit" option. The link type to use is "Single-threaded TCP/IP Server".

Note: You can set up more than one single-threaded listener in the same M account, as long as the TCP/IP Port setting for each listener is unique.

Link Settings for the Single-Threaded Listener

Field	Description
LLP TYPE	TCP
TCP/IP Address	Null (DSM for OpenVMS); IP address of listener system (Caché for NT)
TCP/IP Port	Port to listen on, e.g., 5000
TCP/IP Service Type	SINGLE LISTENER
Startup Node	Sets the startup node for the listener when the link is Autostarted. This setting is honored if a) a Task Manager process is running on the specified startup node, or b) you are an OpenVMS site running TaskMan in a DCL context.
Autostart	If ENABLED, will start up automatically on system boot
Ack Timeout	Begin by leaving at the default setting; increase from default if network is slow.

6.2.1 Starting and Stopping the Listener

To start single-threaded listeners, use the Start/Stop Links option as you would for any outgoing link. Choose the link entry you defined for the listener.

To stop the listener, use any of the options to stop running links.

6.3 Multi-Threaded Listener for Caché on NT

Kernel patch XU*8*78 provides a multi-threaded listener for TCP/IP messaging for Caché on NT systems.

To set up this multi-threaded listener:

- Define an entry in the HL Logical Link file for it. The link type to use is "Multi-threaded TCP/IP Server".

No additional setup is required.

Link Settings for the Multi-Threaded Listener (Caché for NT)

Field	Description
LLP TYPE	TCP
TCP/IP Address	IP address of listener system
TCP/IP Port	Port to listen on, e.g., 5000
TCP/IP Service Type	MULTI LISTENER
Startup Node	Sets the startup node for the listener when the link is Autostarted. This setting is honored if a) a Task Manager process is running on the specified startup node, or b) you are an OpenVMS site running TaskMan in a DCL context.
Autostart	If ENABLED, will start up automatically on system boot
Ack Timeout	Begin by leaving at the default setting; increase from default if network is slow.

6.3.1 Starting and Stopping the Listener

To start multi-threaded listeners for Caché, use the Start/Stop Links option as you would for any outgoing link. Choose the link entry you defined for the listener.

To stop the listener, use any of the options to stop running links.

6.4 UCX Multi-Threaded Listener for OpenVMS

For DSM for OpenVMS, multi-threaded listeners are implemented externally, using DSM for OpenVMS's UCX (also known as Digital TCP/IP Services for OpenVMS). UCX permits multiple TCP/IP clients to connect and run as concurrent processes, up to the limits established by the system. UCX listens on a particular port, and launches a specified HL7 handler process for each client connection.

For the UCX HL7 handler process, you need to create:

- A link
- An OpenVMS account
- A home directory
- A DCL (Digital Command Language) login command procedure

An example is used throughout this section. For this example the following names are used:

- The OpenVMS HL7 handler account name for UCX is HLSEVEN
- The home directory is [HLSEVEN]
- The DCL login command procedure is named HLSEVEN.COM

Note: You can set up more than one UCX service for HL7, although it is not necessary to do this. Some sites prefer one username, command file and link per type of TCP/IP HL7 activity, to distinguish where the messages are coming from over any particular listener's link. To set up more than one UCX service for HL7, follow these steps for each listener, but use different user accounts, directories, command files and UCX service names for the listeners, with a different internal entry number in each command file to direct the spawned process to a particular link.

6.4.1 Set Up Link

Create a link for the Multi-Threaded Listener as follows:

Link Settings for Multi-Threaded Listener, DSM for OpenVMS (using UCX)

Field	Description
LLP TYPE	TCP
TCP/IP Address	null
TCP/IP Port	null
TCP/IP Service Type	MULTI LISTENER
Startup Node	null
Autostart	null
Ack Timeout	(Increase from default if network is slow)

6.4.2 Set Up OpenVMS User Account

The easiest way to configure an OpenVMS account to be an HL7 handler is to copy most of the parameters from VA MailMan TCP account. To do this:

1. Determine an unused User Identification Code (UIC), typically in the same group as other DSM for OpenVMS accounts.
2. Using the OpenVMS Authorize utility, copy the XMINET account to a new HLSEVEN account with the unused UIC. You must have SYSPRV to do this.
3. Make sure that the account settings for the new HLSEVEN account are the same as in the following example, or, if they are different, that the impact of the different settings is acceptable for your system. In particular, make sure that the DisCtlY, Restricted and Captive flags are set for security reasons.

Example (Contains Recommended Settings)

1. Review the XMINET (TCP/IP MailMan) VMS account.

```
$SET DEF SYS$SYSTEM
$MC AUTHORIZE
UAF> SHOW XMINET

Username: XMINET                        Owner:  DSM
Account:                                    UIC:    [50,44] ([XMINET])
CLI:      DCL                               Tables: DCLTABLES
Default:  SYS$SYSDEVICE:[XMINET]
LGICMD:   NL:
Flags:  DisCtlY Restricted Captive
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), 10-FEB-1998 15:30 (non-interactive)
Maxjobs:         0  Fillm:       500  Bytlm:       100000
Maxacctjobs:     0  Shrfillm:      0  Pbytlm:           0
Maxdetach:       0  BIOlm:       150  JTquota:       4096
Prclm:           8  DIOlm:        18  WSdef:         1344
Prio:            4  ASTlm:       176  WSquo:         2688
Queprio:         4  TQElm:        10  WSextent:     65536
CPU:        (none)  Enqlm:      3000  Pgflquo:     100000
Authorized Privileges:
  NETMBX    OPER      SHARE     TMPMBX
Default Privileges:
  NETMBX    OPER      SHARE     TMPMBX
```

2. Copy XMINET (TCP/IP MailMan) account to a new account with unused UIC

Note: This example assumes that UIC [51,45] is an unused UIC. Substitute an unused UIC on your system.

```
UAF> COPY /ADD XMINET HLSEVEN/UIC=[51,45]
%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> SHOW HLSEVEN

Username: HLSEVEN                          Owner:  DSM
Account:                                   UIC:    [51,45] ([HLSEVEN])
CLI:      DCL                              Tables: DCLTABLES
Default:  SYS$SYSDEVICE:[XMINET]
LGICMD:   NL:
Flags:  DisCtlY Restricted Captive
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:       500  Bytlm:       100000
Maxacctjobs:     0  Shrfillm:      0  Pbytlm:           0
Maxdetach:       0  BIOlm:       150  JTquota:       4096
Prclm:           8  DIOlm:        18  WSdef:         1344
Prio:            4  ASTlm:       176  WSquo:         2688
Queprio:         4  TQElm:        10  WSextent:     65536
CPU:        (none)  Enqlm:      3000  Pgflquo:     100000
Authorized Privileges:
  NETMBX    OPER      SHARE     TMPMBX
Default Privileges:
  NETMBX    OPER      SHARE     TMPMBX
```

3. Modify home login directory of the new account.

```
UAF> MOD HLSEVEN/DIR=[HLSEVEN]
%UAF-I-MDFYMSG, user record(s) updated
UAF> EXIT

%UAF-I-DONEMSG, system authorization file modified
%UAF-I-RDBDONEMSG, rights database modified
```

6.4.3 Set Up Home Directory for the HL7 Handler Account

You need to create a home directory for the HL7 handler account. This directory will house the DCL command procedure that is executed whenever a client connects, as well as log files. 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
```

6.4.4 Create a DCL Login Command Procedure for the HL7 Handler

Create a DCL command procedure in the home directory for the handler account. Make sure the command procedure file is owned by the HL7 handler account.

1. Adjust the DSM command line (environment, UCI and volume set) for your system.
2. 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).
3. Replace 999 with the internal entry number of the link you created (in the HL Logical Link file) to be invoked by the listener-spawned process.

Sample DCL Login Command Procedure

```
$!HLSEVEN.COM - for incoming connect requests
$!-----------------------------------------------------------------
$ set noon                      !Don't stop
$ set verify
$ purge/keep=2 sys$login:*.*
$ set proc/priv=(share)         !Required to use the MBX device
$ x=f$trnlnm("sys$net")         !This is our MBX device
$ write sys$output x            !This can be viewed in the log file
$ set nover                    !Turn off verify
$!-----------------------------------------------------------------
$! **Be sure this command line is correct for your system
$! **and if access control is enabled that this account has
$! **access to this uci,vol & routine.  The number 999 should be replaced
$! **with the internal entry number in file 870 for this Logical Link
$!                                                        /
$ dsm/environ=MGRISC/uci=VAH/vol=ROU/data="''x'^999" EN^HLCSTCP
$!-----------------------------------------------------------------
$ logout/brief
```

6.4.5 Set Up and Enable the UCX Service

Once you create the HL7 handler, create the UCX service to listen for connections and launch the HL7 handler. You need to choose:

- The OpenVMS node to run the listener on. Choose the node that you want to run the resulting M jobs on to process incoming HL7 messages. This is also the node whose IP address will be advertised to other systems as the location of your HL7 listener.
- The port it should listen on.
- The user account and command file name to invoke when a connection is received.

Prior to setting UCX up in production, you can set up a "test" UCX service that logs into an M test account for testing. The "test" UCX service can use the same OpenVMS account and directory that the production UCX service will. Just create a different DCL command file with the UCI and volume set of the test account.

Note: It is possible to run different UCX listener processes on multiple nodes. It is not necessary to do this; however, depending on your site configuration and needs, you may find a need to do so.

The steps to set up a UCX service for **V*IST*A** HL7 are:

1. Set up the "HLSEVEN" UCX service
2. Enable and save the "HLSEVEN" UCX service

1. Set up the HLSEVEN UCX Service

Since UCX is node specific, make sure you are on the same node that you want the listener to run on.

```
$UCX
UCX> SET SERVICE HLSEVEN/USER=HLSEVEN/PROC=HLSEVEN /PORT=5000-
_UCX> /PROTOCOL=TCP/REJECT=MESSAGE="All channels busy" -
_UCX> /LIMIT=50/FILE=SYS$SYSDEVICE:[HLSEVEN]HLSEVEN.COM

UCX> SHO SERVICE HLSEVEN/FULL

Service: HLSEVEN
                          State:     Disabled
Port:             5000    Protocol:  TCP            Address:  0.0.0.0
                          User_name: not defined    Process:  HLSEVEN
```

2. Enable and Save the HLSEVEN UCX Service

Since UCX is node specific, make sure you are on the same node that you want the listener to run on.

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

Service: HLSEVEN
                           State:     Enabled
Port:              5000    Protocol:  TCP             Address:  0.0.0.0
Inactivity:           5    User_name: HLSEVEN         Process:  HLSEVEN
Limit:               50    Active:      0             Peak:       0

File:          SYS$SYSDEVICE:[HLSEVEN]HLSEVEN.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

UCX> SHO CONFIG ENABLE SERVICE

Enable service
     FTP, FTP_CLIENT, HLSEVEN, MPI, TELNET, XMINETMM
UCX> EXIT
```

6.4.6 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  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:  SYS$SYSDEVICE:[DSMMGR]DSM$ACCESS.DAT.
```

6.4.7 How to Control the Number of Log Files Created by UCX

The HLSEVEN UCX service automatically creates log files (UCX does this and it cannot be prevented) in the HLSEVEN directory named HLSEVEN.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 disk$:[HLSEVEN]HLSEVEN.LOG; disk$:[HLSEVEN]HLSEVEN.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 disk$:[HLSEVEN]HLSEVEN.LOG;
```

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

6.4.8 Starting and Stopping the Listener

Although the multi-threaded listener for OpenVMS systems requires the setup of a link, you never actually start or stop this link with **V*IST*A** HL7 options. Instead, because the listener is implemented as a UCX service, start it up and shut it down by using the generic tools UCX provides to enable and disable a UCX service. For example:

```
UCX> ENABLE SERVICE HLSEVEN <RET>                    (Start up UCX service)

UCX> DISABLE SERVICE HLSEVEN <RET>                   (Stop UCX service)
```

CHAPTER

7. V*ISTA* HL7's Interface Framework

7.1 If You're Starting Out with HL7

If you're starting out with HL7, the best starting point to learn about HL7 before building interfaces is the *Health Level Seven Standard* itself. The most important sections of the standard to read are chapter one (Introduction) and chapter two (Control/Query). In particular, chapter two lays out the rules for constructing, sending and responding to HL7 messages.

The ***VISTA** HL7 Site Manager & Developer Manual*, while serving as a reference for **V***IST***A** HL7, does not attempt to be either a tutorial or a reference for the HL7 standard.

7.2 HL7 Interfaces

An *interface* is the defined set of HL7 transactions between two applications. It defines which HL7 messages are sent between two or more systems, in what direction, how each message should be acknowledged, and what the contents should be for each message.

An interface *specification* is a paper representation of the negotiated, agreed upon messages, segments, acknowledgments, and so forth to be exchanged between systems.

In addition, each side of the interface typically needs to have the interface modeled so that application code can generate and process the appropriate messages for the interface.

7.3 Modeling Interfaces for V*IST*A HL7

For **V***IST***A** M-based systems, **V***IST***A** HL7 provides the facilities to model a given HL7 interface for **V***IST***A** HL7's use. It provides a framework within which **V***IST***A** HL7 supports transactions between two systems. A **V***IST***A** HL7 interface defines the sending and receiving parties (applications) on each system, the transaction type (message sent) and the connectivity for each system.

A **V***IST***A** HL7 comprises the following components:

Component	Purpose	Stored in
Application	Defines sending and receiving parties	HL7 Application Parameter file
Protocols	Defines each transaction (message) type	Protocol file
Links	Defines connectivity to other systems	HL Logical Link file

In general, the steps to model an interface between a **V***IST***A** M system and another system are:

1. Create an application entry in **V***IST***A** HL7 for each application, i.e., one for your application on the **V***IST***A** side of the interface, and one for the application on the system on the other side of the interface.
2. Create event driver protocol/subscriber protocol pairs for each transaction originating on the **V***IST***A** side of the interface:
    a. Create an event driver protocol for each HL7 message type/event type generated by the application on the **V***IST***A** side of the interface.
    b. Create a subscriber protocol for each HL7 message type/event type that will be received by the other side of the interface. Subscribe that protocol to the corresponding event driver protocol.
3. Create event driver protocol/subscriber protocol pairs for each transaction originating on the other side of the interface:
    a. Create an event driver protocol for each HL7 message type/event type generated by the application on the other side of the interface.
    b. Create a subscriber protocol for each HL7 message type/event type that will be received on the **V***IST***A** HL7 side of the interface. Subscribe that protocol to the corresponding event driver protocol.
4. Create a link for the system on the other side of the interface. Attach the link to:
    a. Each subscriber protocol representing the other system's subscription to an HL7 message type/event type generated on your system.

Note: Later on, you may also need to attach the link to each subscriber protocol representing *your* system's subscription to an HL7 message type/event type generated on the other system, *only* if you need to return acknowledgments, and only under certain conditions. For more information, please see the "Building Interfaces: Acknowledgments" chapter.

7.4 V*IST*A HL7 Applications

According to the HL7 standard, an HL7 *application*:

> "...uniquely identifies the sending [or receiving] application among all other applications within the network enterprise. The network enterprise consists of all those applications that participate in the exchange of HL7 messages within the enterprise. Entirely site-defined."[5]

In **V*IST*A** HL7, when you create an interface, you need to create an entry in the HL7 Application Parameter file for *each* sending application and *each* receiving application. This models the sending and receiving applications for **V*IST*A** HL7. For example, to model the interface between a COTS system and **V*IST*A**, you would create two **V*IST*A** HL7 application entries, one for the COTS system and one for the **V*IST*A** application interfacing with the COTS system.

To create and edit HL7 applications (entries in the HL7 Application Parameter file), use the following option on the Interface Developer Options Menu:

- Application Edit

This option uses a one-form ScreenMan form so that you can edit application entries:

```
                          HL7 APPLICATION EDIT
------------------------------------------------------------------------------

               NAME: RH SERVER APP                ACTIVE/INACTIVE: ACTIVE

      FACILITY NAME: RH ISC                          COUNTRY CODE: US

HL7 FIELD SEPARATOR: |                    HL7 ENCODING CHARACTERS:

         MAIL GROUP: HL7




_______________________________________________________________________________



COMMAND:                                      Press <PF1>H for help     Insert
```

V*IST*A HL7 uses application entries as one of the identifying keys to uniquely identify a transaction, for messages being sent and received.

[5] Health Level Seven, *Health Level Seven, Version 2.3.1*, copyright 1999, p. 2-94.

7.4.1 Application Fields

Field	Description
Name	(required) In **V*IST*A** HL7, the application name is one of the identifying keys used to uniquely identify a transaction. It defines one side of a particular interface between systems.
Facility Name	(optional) The name of the facility to which this application belongs. This information is placed in the Sending Facility field of the message header for HL7 messages generated on the system. Typically, this field is used to enter the Facility Name for application entries for COTS systems. For application entries for **V*IST*A** facilities, we recommend leaving this field unpopulated, so that the default value based on site parameters (sending applications) or the link (receiving applications) is used instead.
Active/Inactive	(optional) Use the Activate/Inactivate DHCP Application option to activate or inactivate a specific application. An application must be active in order to send or receive messages. If you *inactivate* an application, incoming and outgoing transactions won't work: • If a message is received, it will be stored but not processed; **V*IST*A** HL7 puts the message in an error state. • If a disabled application tries to send a message, it is again held by **V*IST*A** HL7; it is not transmitted, and it's put in an error state.
Country Code	(optional) The country code applicable to this application.
HL7 Field Separator	(optional) The field separator separates two adjacent fields within a segment. It also separates the segment ID from the first data field in the segment. The field separator can be defined differently for each HL7 message. The fourth character in the message header segment serves as the field separator for all segments in the HL7 message. If this field is left blank, the default will be the '^' character. However, the preferred character for message readability (and the one used by most messaging applications) is the vertical bar ("|").
HL7 Encoding Characters	(optional) The HL7 encoding characters to use for messages generated by the application (component separator, repetition separator, escape character, sub-component separator). Defaults to the characters ~|\&. **Note:** If you choose this default, the Field Separator must be set to "^". If you choose the vertical bar ("|") as a field separator, you should enter "~^\&" as your encoding characters. This avoids a potential conflict with VA FileMan, and is the preferred format for message readability. VA FileMan does not support interactive entry of "^" as the first (component) separator.
Mail Group	(optional) The application-specific mail group to which notifications should be sent if the application determines that problems have arisen with delivering messages. **Note: V*IST*A** application code should use the GETAPP^HLCS2 entry point to retrieve the mail group associated with a particular HL7 application, which it can then send notifications to.

7.5 V*IST*A HL7 Protocols

For each transaction (a message type/event type pair, e.g., ADT~A01) sent or received over a given interface, you need to define an *event driver protocol* on your system to represent the transaction from the point of view of the sending application, and a *subscriber protocol* to represent the transaction from the point of view of the receiving application.

- **Event Driver Protocol**: Represents the sending application's side of a transaction.
- **Subscriber Protocol**: Represents the receiving application's side of a transaction

By creating an event driver protocol and subscriber protocol for each transaction/message type exchanged over your interface, you model that particular transaction for **V*IST*A** HL7.

To edit HL7 protocols (entries in the Protocol file), use the following option on the Interface Developer Options menu:

- Protocol Edit

This option uses a two-form ScreenMan form so that you can edit application entries:

- Form page 1: Fields for all protocols
- Form page 2: Fields specific to the type of protocol (event driver or subscriber)

The first page of the form looks like:

```
                        HL7 INTERFACE SETUP                          PAGE 1 OF 2
------------------------------------------------------------------------------

            NAME: RH ORU CLIENT

DESCRIPTION (wp):   (empty)


ENTRY ACTION:

 EXIT ACTION:


        TYPE: subscriber



_______________________________________________________________________________



COMMAND:                                      Press <PF1>H for help     Insert
```

Each protocol is assigned to an application. This relationship is important; **V*IST*A** HL7 uses application entries as one of the identifying keys to uniquely identify a transaction, for messages being sent and received.

7.5.1 All Protocol Types (Form Page 1)

Field	Description
Name	(required) The name should uniquely identify one or more of the following characteristics of the protocol: application, message type, role (event driver or subscriber), and link type (e.g., TCP).
Description	(optional) A word processing field to hold a description of the protocol.
Entry Action	(optional) Not typically used. M code executed: • If **V*IST*A** HL7 is sending the message, before a message is sent (event driver protocol only) • If **V*IST*A** HL7 is receiving the message, when receiving an inbound message (subscriber protocol only)
Exit Action	(optional) Not typically used. M code executed: • If **V*IST*A** HL7 is sending the message, after returning from sending a message (event driver protocol only) • If **V*IST*A** HL7 is receiving the message, after the processing routine is called and the message is processed (processing may include sending an ack) (subscriber protocol only).
Type	(required) A protocol can be one of two types: • Event Driver • Subscriber The type of protocol chosen affects what set of fields is available for editing in the second form of the Protocol Edit option.

The remaining fields to edit are specific to the type of protocol, i.e., event driver or subscriber.

7.6 Event Driver Protocols

An event driver protocol represents the sending application's side of a transaction for a particular message type/event type, whether the message originates on the **V*IST*A** side of the interface, or on the other side of the interface. In both cases, an event driver protocol must be set up in **V*IST*A** HL7, and the required fields are Sending Application, Message Type, Event Type and Version ID.

7.6.1 Event Driver Protocol Fields (Form Page 2)

Field	Description
Sending Application	(required) Select the entry representing the sending application from the HL7 Application Parameter File. **Note:** You can directly add new applications.
Transaction Message Type	(required) Enter the message type that is being generated. This is used when generating the header for the outbound message.
Event Type	(optional) Select the HL7 Event Type code for the message that is being generated. *Required* for HL7 versions 2.2 and greater.
Processing ID	(optional) Used to override the site-parameter-generated Processing ID field of the message header. The only value you can override the site parameter setting with is "DEBUG".
Version ID	(required) Select the version of the HL7 Standard used to implement this transaction. The version selected must correspond to the appropriate version of the Message Type selected.
Accept Ack Code	(optional) For version 2.2 or higher transactions only. Select whether or not a commit ack will be generated by **V*IST*A** HL7. Can be AL (always), ER (Error/Reject Conditions Only), NE (never) or SU (successful completions only). Applies only to enhanced mode acknowledgments.
Application Ack Type	(optional) For version 2.2 or higher transactions only. Select whether or not the receiving application is expected to return an acknowledgment. Typically set to AL (always) or NE (Never).
Response Processing Routine	(optional) Enter the M code to invoke on the sending system when an acknowledgment or query response to the original message is received.
Subscribers	(optional) Each subscriber protocol that subscribes to (receives) the message generated by this event driver protocol should be entered here.

7.7 Subscriber Protocols

A subscriber protocol represents the receiving application's side of a particular transaction for a particular message type/event type, whether the message is being received by the **V*IST*A** side of the interface, or by the other side of the interface. In both cases, a subscriber protocol must be set up in **V*IST*A** HL7, and the required fields are Sending Application, Message Type, Event Type and Version ID.

7.7.1 Subscriber Protocol Fields (ScreenMan Page 2)

Field	Description
Receiving Application	(required) Select the entry representing the receiving application from the HL7 Application Parameter File. **Note:** You can directly add new applications.
Response Message Type	(optional) Enter the message type of the expected response, if as part of the transaction an acknowledgment and/or query response needs to be returned.
Event Type	(required) Select the HL7 Event Type code for the application response message.
Sending Facility Required?	(optional) Enter YES if the sending facility field is required in the message header segment.
Receiving Facility Required?	(optional) Enter YES if the receiving facility field is required in the message header segment.
Security Required?	(optional) Enter YES if the security field is required in the message header segment.
Date/Time of Message Required?	(optional) No longer used. **V*IST*A** HL7 now automatically stuffs date and time (with time zone offset) into all outgoing message headers. It no longer checks whether incoming message headers contain date/time.
Logical Link	(optional) Select the link that describes the network path to the subscriber system. Use this type of addressing for point-to-point interfaces, such as to a local COTS application or another **V*IST*A** facility.
Processing Routine	(optional) Enter the M code to process the inbound message in this transaction. If an acknowledgment needs to be returned to the sending system, the code should call GENACK^HLMA1.
Routing Logic	(optional) Enter M code to interpret the message and route the message to those interested in the data when it meets run-time conditions. This field is executed only when a message is in an OUTBOUND state. For more information, please see the "Dynamic Addressing" section of the "Advanced Interface Issues" chapter.

7.8 V*IST*A HL7 Links

The delivery path to external systems is modeled by *links*, the third component of a **V*IST*A** HL7 interface. Links can be one of four types (HLLP, X3.28, MailMan and TCP) depending on how the target system is connected. Akin to the Device and Domain files, this file contains the specific HLLP, X3.28, TCP/IP, or MailMan information needed by **V*IST*A** HL7 to reach and properly communicate with destination systems.

When you send a message to a target system, it is delivered to the link specified in the Logical Link field of the subscriber protocol (representing the target system). Absence of a defined link for a subscriber protocol means that the subscriber protocol is being used for a same-system exchange.

In addition, when you receive a message from a system and need to return an acknowledgment, a link needs to be defined on the "receiving" system's subscriber protocol, in some situations.

To edit links (entries in the HL Logical Link file), use the "Protocol Edit" option on the Interface Developer Options menu. For more information about setting up links, please see the "Link Setup" chapter.

7.9 Next Steps in Building an Interface

The application, protocols and links together comprise a model of the interface — a framework — that **V*IST*A** HL7 uses to support transactions over the interface. Once you put this framework in place, you can:

- **Send Messages.** To send an HL7 message when an event point occurs, you build the message body in an array, call a **V*IST*A** HL7 entry point, and pass the name of the event driver protocol to invoke to the entry point.

  For more information, please see the "Building Interfaces: Sending a Message" chapter.

- **Receive Messages.** To process messages received over the interface, you create a routine to process the incoming message. You configure the appropriate subscriber protocol to call your processing routine whenever an incoming message causes the protocol to be invoked (through **V*IST*A** HL7).

  For more information, please see the "Building Interfaces: Receiving a Message" chapter.

- **Generate and Process Acknowledgments.** To generate acknowledgments, you call a **V*IST*A** HL7 entry point while processing an incoming message.

  To process acknowledgments, you build a processing routine, and then configure the original event driver protocol to call your processing routine whenever an incoming acknowledgment is received for a message generated by the protocol.

  For more information, please see the "Building Interfaces: Acknowledgments" chapter.

C H A P T E R

8. Building Interfaces: Sending a Message

When an event occurs in your **V*IST*A** application for which an HL7 message should be sent, your application's M code needs to a) build the body of the message, and then b) call **V*IST*A** HL7 entry points to send the message. An interface framework (application, event driver protocol, subscriber protocol and link) must be in place in **V*IST*A** HL7 already that represents the interface.

8.1 Interface Setup Example: V*IST*A to COTS System

This example shows how to set up an interface between a **V*IST*A** M system and a non-M system called the ACME system. The interface has the following characteristics:

- The single transaction for this simple interface is for message type ORM, event type O01, HL7 V. 2.3.
- No application acknowledgment is required.
- The message is sent from **V*IST*A** to the ACME system.
- The ACME system is connected to **V*IST*A** via TCP/IP.

To model this interface for **V*IST*A** HL7, set up the following entries in **V*IST*A** HL7 on the **V*IST*A** side for the interface, in the following order:

Step	Create:	Entry Name	Purpose
1.	Application	VIST-AC	Represents the **V*IST*A** side of the **V*IST*A**-ACME interface
2.	Application	ACME	Represents the ACME side of the **V*IST*A**-ACME interface
3.	Link	ACME TCP	Network path to ACME System
4.	Event Driver Protocol	VIST-AC ORM\O01 EVENT	Protocol to invoke when generating a message to send
5.	Subscriber Protocol	ACME ORM\O01 SUBS	Protocol that subscribes to the event, routing the message to the ACME system

8.2 Role of Protocols for Sending Messages

To send a message to another system (in this example, the ACME system), an Event Driver protocol must be defined for the sending application on the **V*IST*A** system. **V*IST*A** HL7 uses the event driver protocol to generate the outgoing message's header.

In addition, for every target system that is a recipient of the message, a subscriber protocol must be defined. The subscriber protocol is used to route the message to a specified recipient:

- The target system to send the message to is specified by the link setting of the subscriber protocol.
- The application on the target system to send the message to is specified in the subscriber protocol's application.

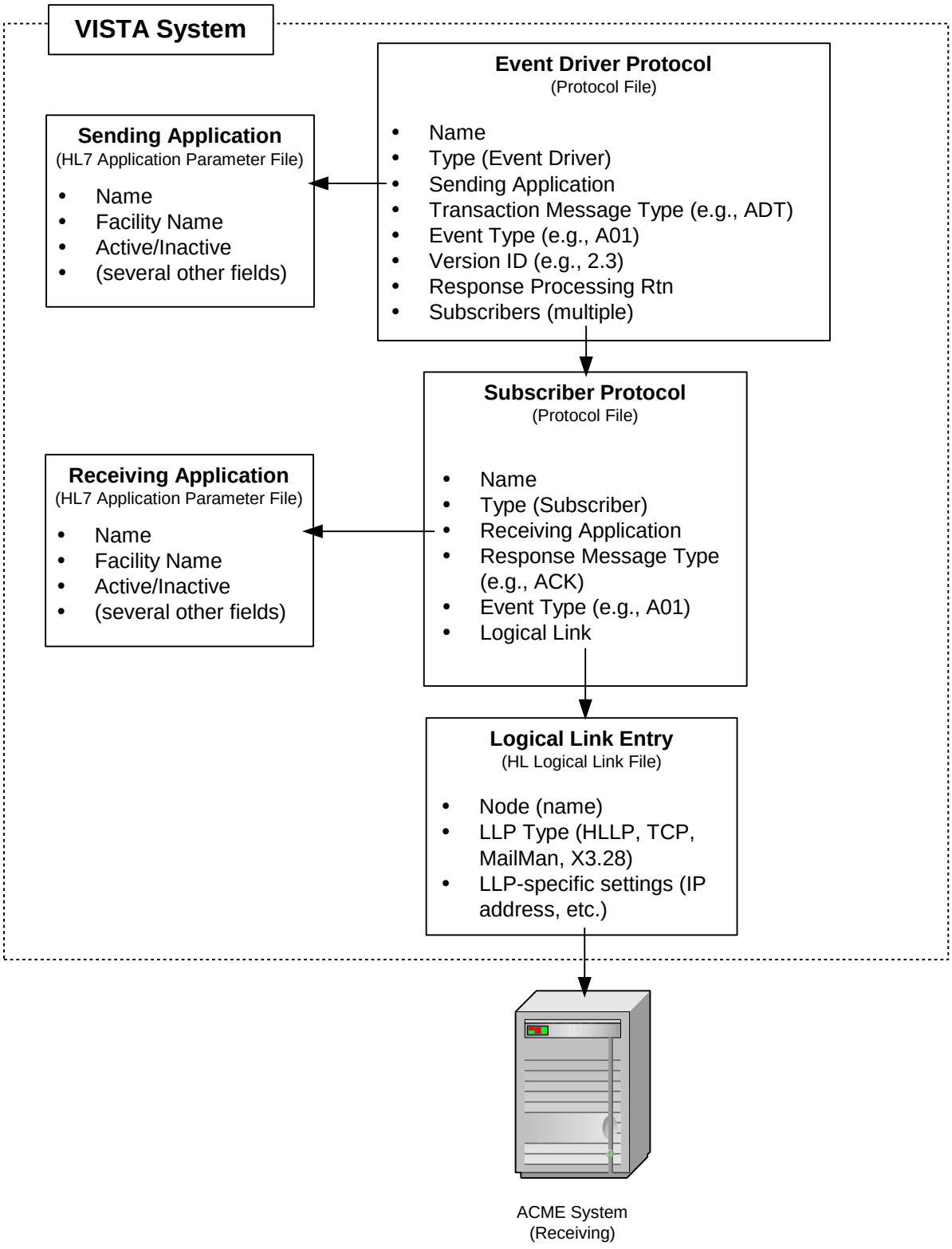
8.3 Sending System: How to Originate a New Message

In your application code, to generate and send a new message, the M routine that determines an event point has happened should:

1. To set up variables needed by your routine to build the HL7 message, call INIT^HLFNC2, passing the appropriate event driver protocol (in this example, VIST-AC ORM\O01 EVENT).
2. Check $G(HL). If this check is true, the call to INIT^HLFNC2 failed. The HL variable contains the error message/reason for failure. Your driver routine should quit.
3. Build the message text (but not the message header). Set the HL7 segments for the message body into local array HLA("HLS",I) for small messages, or ^TMP("HLS",$J,I) for larger messages. In both cases, "I" is a sequential whole number starting with one.
4. To send the message, invoke GENERATE^HLMA, passing the appropriate event driver protocol as a parameter.
5. If the call to GENERATE^HLMA is successful, the Result parameter is returned equal to the message id assigned to the message that was created. If the call was unsuccessful, it is returned with the following three pieces of data:

   message id (or 0 if no message id was assigned)^error code^error message

Example: V*ISTA* to a Single COTS System



8.4 Interface Setup Example

To set up an interface to support one transaction type, a message from **V*IST*A** to the ACME system:

1. Create an application for the **V*IST*A** side of the transaction. Name it VIST-AC:

```
                          HL7 APPLICATION EDIT
--------------------------------------------------------------------------------

               NAME: VIST-AC                       ACTIVE/INACTIVE: ACTIVE

      FACILITY NAME:                                  COUNTRY CODE: US

HL7 FIELD SEPARATOR:                       HL7 ENCODING CHARACTERS: \|~&

         MAIL GROUP:


________________________________________________________________________________



COMMAND:                                        Press <PF1>H for help     Insert
```

2. Create an application for the ACME side of the transaction. Name it ACME:

```
                          HL7 APPLICATION EDIT
--------------------------------------------------------------------------------

               NAME: ACME                          ACTIVE/INACTIVE: ACTIVE

      FACILITY NAME:                                  COUNTRY CODE: US

HL7 FIELD SEPARATOR:                       HL7 ENCODING CHARACTERS: \|~&

         MAIL GROUP:


________________________________________________________________________________



COMMAND:                                        Press <PF1>H for help     Insert
```

3. Create a link for the ACME system named "ACME TCP":
   - Set LLP Type to TCP
   - Set TCP/IP address to that of the ACME system
   - Set TCP/IP Port to the port the ACME system listens on for connections
   - Set the TCP/IP Service Type to CLIENT.

```
                          HL7 LOGICAL LINK
------------------------------------------------------------------------------
   ┌──────────────────────TCP LOWER LEVEL PARAMETERS──────────────────────┐
   │                      ACME TCP                                        │
   │                                                                      │
   │  TCP/IP SERVICE TYPE: CLIENT (SENDER)                                │
   │       TCP/IP ADDRESS: 152.125.X.XXX                                  │
   │          TCP/IP PORT: 5500                                           │
   │           PERSISTENT:                                                │
   │            RETENTION:                                                │
   │                                                                      │
   │ ACK TIMEOUT:                                    READ TIMEOUT:        │
   │  BLOCK SIZE:                          RE-TRANSMISSION ATTEMPTS:      │
   │                                                                      │
   │STARTUP NODE:                              UNI-DIRECTIONAL WAIT:      │
   └──────────────────────────────────────────────────────────────────────┘
_______________________________________________________________________________


COMMAND:                                        Press <PF1>H for help    Insert
```

4. Create an event driver named "VIST-AC ORM\O01 EVENT" for **V*IST*A** (the sender). This will be invoked whenever an ORM\O01 message is sent from **V*IST*A** to subscribing applications, such as the ACME system.

```
                          HL7 EVENT DRIVER                        PAGE 2 OF 2
                        VIST-AC ORM\O01 EVENT
------------------------------------------------------------------------------
      SENDING APPLICATION: VIST-AC
 TRANSACTION MESSAGE TYPE: ORM                      EVENT TYPE: O01
            PROCESSING ID: DEBUG                    VERSION ID: 2.3
          ACCEPT ACK CODE: AL              APPLICATION ACK TYPE: NE

 RESPONSE PROCESSING RTN:

                            SUBSCRIBERS




_______________________________________________________________________________


COMMAND:                                        Press <PF1>H for help    Insert
```

5. If the target system (in this case, the ACME system) will return application acknowledgments for this transaction, you will need to receive and process those acknowledgments. This is done by creating a processing routine for the acknowledgments, and setting the event driver protocol's RESPONSE PROCESSING RTN field to the name of the processing routine. For more information, please see the "Building Interfaces: Acknowledgments" chapter.

6. Create the subscriber protocol that will act as the recipient for the message, and route the message to the ACME system.

Note: While editing event driver protocols, and editing the Subscribers multiple, you can select existing subscriber protocols *or* enter new ones without leaving the current edit screen. So, while editing the VIST-AC ORM\O01 EVENT event driver protocol, you can define the subscriber protocols that will route the ORM\O01 message to subscriber systems as well (in this case, ACME).

So, while editing the VIST-AC ORM\O01 EVENT protocol, tab down to "Subscribers" and add the ACME ORM\O01 SUBS as a new subscriber protocol. For the link (the physical destination to deliver messages to), choose the ACME TCP link that you created earlier.

```
Are you adding 'ACME ORM\O01 SUBS' as a new PROTOCOL? No// Y

 PROTOCOL ITEM TEXT: <RET>

 PROTOCOL IDENTIFIER: <RET>
```

```
                          HL7 EVENT DRIVER                          PAGE 2 OF 2
    ┌──────────────────────────HL7 SUBSCRIBER──────────────────────────┐
---  │                       ACME ORM\O01 SUBS                         │  ---
    ├─────────────────────────────────────────────────────────────────┤
 TR │     RECEIVING APPLICATION: ACME                                 │
    │     RESPONSE MESSAGE TYPE: ACK                EVENT TYPE: A01   │
 RE │SENDING FACILITY REQUIRED?: NO       RECEIVING FACILITY REQUIRED?: NO
    │        SECURITY REQUIRED?:                                      │
  R │              LOGICAL LINK: ACME TCP                             │
    │ PROCESSING RTN:                                                 │
    │  ROUTING LOGIC:                                                 │
    └─────────────────────────────────────────────────────────────────┘
_____________________________________________________________________________


COMMAND:                                    Press <PF1>H for help    Insert
```

7. Create the M routine to generate the ORM\O01 message from **V*IST*A** to send to the ACME system.

Example: Routine to Generate and Deliver a Single Message

```
BLDM    ;BUILD A SINGLE MESSAGE

        K HLA,HLEVN
        N CNT,MC,HLFS,HLCS
        S CNT=0
1       ;set up environment for message
        D INIT^HLFNC2("VIST-AC ORM\O01 EVENT",.HL)
        I $G(HL) D  Q ; error occurred
        . ; put error handler here for init failure
        . W !,"Error: "_$P(HL,2)
        S HLFS=$G(HL("FS")) I HLFS="" S HLFS="^"
        S HLCS=$E(HL("ECH"),1)
        ;
2       ;Add message txt to HLA array
        ; create PID segment for patient DFN -- call segment generator
        S CNT=CNT+1,HLA("HLS",CNT)=$$EN^VAFHLPID(DFN)
        ; add PV1 segment
        S CNT=CNT+1,HLA("HLS",CNT)="PV1"_HLFS_"MEDICINE"_HLFS_"123456"_HLCS_
"HL7Patient_HLCS_"One^^^^23452"_HLCS_"DR"_HLCS_"HL7Doctor"
        ;
3       ;demo use of continuation (very 'long') segments (greater than 245) All
segments are terminated with a carriage return in HL7. To store any sized
segment in a global, an empty global node is used to represent the end of the
segment.
        S CNT=CNT+1,HLA("HLS",CNT)="NK1^" D
        . S HLA("HLS",CNT,1)="^TEST MESSAGE FROM NXT^^^^^^1"
        . S HLA("HLS",CNT,2)="^^^^^^^^^^2"
        ;
4       ;CALL HL7 TO TRANSMIT SINGLE MESSAGE
        D GENERATE^HLMA("VIST-AC ORM\O01 EVENT","LM",1,.MYRESULT,"",.MYOPTNS)
        I +MYRESULT'=MYRESULT D
        . ; put error handler here for message send failure
        Q
```

8.5 How Outgoing Message Headers are Formed

When coding the routine to generate an HL7 message, you are responsible for generating the message body -- the set of segments composing the text of the message. You are not responsible for forming the message header, however; this is created and managed for you by **V*IST*A** HL7.

Most fields in the message header are supported by **V*IST*A** HL7, particularly with the release of patch HL*1.6*57. Fields placed in the message header are derived as follows:

Seq	Header Field	Derived From
1	**Field Separator**	Sending application's HL7 Field Separator field in the HL7 Application Parameter file.
2	**Encoding Characters**	Sending application's HL7 Encoding Characters field in the HL7 Application Parameter file.
3	**Sending Application**	Event driver protocol's Sending Application field, which points to an entry in the HL7 Application Parameter file. **Note:** While optional in standard, this field is ***required*** by **V*IST*A** HL7 1.6.
4	**Sending Facility**	Populated only if the subscriber protocol's "Sending Facility Required" field is set to Yes. If so, it comes from: 1. The sending application's Facility Name field, if defined. 2. The Domain and Institution fields in the **V*IST*A** HL7 site parameters, in the format *Inst.#**cs**Domain**cs**DNS*, where ***cs*** = component separator. **Note:** In most cases we recommend that you **don't** override the site parameter with your own Facility Name.
5	**Receiving Application**	Subscriber protocol's Receiving Application field, which points to an entry in the HL7 Application Parameter file. **Note:** While optional in standard, this field is ***required*** by **V*IST*A** HL7 1.6.
6	**Receiving Facility**	Populated only if the subscriber protocol's "Receiving Facility Required" field is set to Yes. If so, it comes from: 1. The receiving application's Facility Name field, if defined. 2. The subscriber protocol's link's Domain and Institution fields, in the format *Inst.#**cs**Domain**cs**DNS*, where ***cs*** = component separator. For transactions with **V*IST*A** facilities, we recommend that you **don't** override the default format derived from the link.
7	**Date/Time of Message**	Inserted automatically by **V*IST*A** HL7.
8	**Security**	Parameters to the GENERATE^HLMA call.

How Outgoing Message Headers are Formed (cont'd)

Seq	Header Field	Derived From
9	**Message Type**	**V*IST*A** HL7 currently creates the two-component version of this field (compatible with HL7 V. 2.2 and 2.3), from the Transaction Message Type field and Event Type field of the event driver protocol. HL7 V. 2.3.1 introduces a third component (message structure), which **V*IST*A** HL7 does not support at this time.
10	**Message Control ID**	Generated and inserted automatically by **V*IST*A** HL7. Uses the institution number defined in the HL7 site parameter "Current Institution" as the prefix, concatenated with a unique number.
11	**Processing ID**	Comes from either: 1. The event driver protocol's Processing ID field, if defined and set to Debug 2. Otherwise, the **V*IST*A** HL7 site parameter "Is This a Production or Test Account?" **Note:** In most cases we recommend that you not override the site parameter with a protocol setting for Processing ID.
12	**Version ID**	Event driver protocol's Version ID field.
13	**Sequence Number**	null (not supported)
14	**Continuation Pointer**	Parameters to the GENERATE^HLMA call.
15	**Accept AckType**	Event driver protocol's Accept Ack Code field.
16	**Application Ack Type**	Event driver protocol's Application Ack Type field.
17	**Country Code**	Sending application's Country Code field in the HL7 Application Parameter file.
18	**Character Set**	null (not supported)
19	**Principal Language of Message**	null (not supported)
20	**Alternate Character Set Handling Scheme**	null (not supported)

CHAPTER

9. Building Interfaces: Receiving a Message

V*IST*A HL7 can receive messages containing almost any message header format and message format, if the message itself is compliant with the HL7 standard.

9.1 Validations Performed on Message Header by V*IST*A HL7

If any validation step fails in the list below, the message is rejected.

1. Field separator and encoding characters must be valid.
2. A valid message header segment must be present (MSH, BHS or FHS).
3. Message type must match an entry in the HL7 Message Type file.
4. Accept acknowledgment and application acknowledgment fields are validated.
5. Sending application field is validated against the HL7 Application Parameter file. If there is no matching entry, of if the matching entry is "Inactive", the message is rejected.
6. Version must match an entry in the HL7 Version file.
7. If the event type is missing from the incoming message's header, and version is greater than 2.1, the message is rejected.
8. If the message is an acknowledgment (if it contains an MSA segment), it is turned over to the original message's event driver protocol for processing. For more information, please see the "Building Interfaces: Acknowledgments" chapter.
9. If the message is not an acknowledgment, an event driver protocol must be found in File #101's "AHL1" cross reference that matches the message header's fields for Message Type, Event Type, Version ID and Sending Application (or in the "AHL2" cross reference for version 2.1 messages without an event type).
10. A subscriber protocol with an application matching the incoming message's Receiving Application field must be present in the event driver protocol's Subscribers multiple.
11. If the subscriber protocol defines either Sending Facility and/or Receiving Facility fields as required, and if either is required but is not present, the message is rejected.
12. If Processing ID is set to Training or Production, it must match the site parameter setting. If it is set to Debug, it must match the event driver protocol setting.
13. If the subscriber protocol's "Security Required?" is set to YES, the access code (and electronic signature, if passed) must match those of an active user in the New Person file.

The nature of the validations performed illustrates both a) the necessity of having a correct interface set up, and b) the purpose for the interface in the first place: to route transactions to the appropriate sending and receiving applications.

9.2 Message Header Requirements for Incoming Messages

V*IST*A HL7 requires certain fields in the message headers to be present in incoming messages, as defined in the following table. **Note**: Two header fields that are optional in the HL7 standard are *required* in incoming messages by **V*IST*A** HL7.

MSH Sequence	MSH Field Name	Required by Standard (2.3)	Required by V*IST*A HL7
01	Field Separator	Yes	Yes
02	Encoding Characters	Yes	Yes
03	Sending Application	Optional	***Required***
04	Sending Facility	Optional	Optional
05	Receiving Application	Optional	***Required***
06	Receiving Facility	Optional	Optional
07	Date/Time Of Message	Optional	Optional
08	Security	Optional	Optional
09	Message Type	Yes	Yes
10	Message Control ID	Yes	Yes
11	Processing ID	Yes	Yes
12	Version ID	Yes	Yes
13	Sequence Number	Optional	Optional
14	Continuation Pointer	Optional	Optional
15	Accept Acknowledgment Type	Optional	Optional
16	Application Acknowledgment Type	Optional	Optional
17	Country Code	Optional	Optional

Note: The Message Control ID field in the incoming message header cannot contain a caret ("^").

9.3 Message Body Requirements for Incoming Messages

The requirements of the message body for an incoming message are:

- The message body must be a well-formed HL7 message as defined by the HL7 standard.
- If the message is an acknowledgment, it must have an MSA segment. The MSA segment must contain the message control ID of the message it is acknowledging, in field MSA-2.
- If the incoming message contains an MSA segment (for acknowledgments only), the MSA segment must be valid according to the HL7 standard. This is the only segment in the message body that **V*IST*A** HL7 ever validates.

9.4 Role of Protocols for Receiving Messages

When a message is received, V*IST*A HL7 makes use of its message header as follows:

1. The values for Message Type, Event Type, Version ID and Sending Application in the message header must match those of an event driver protocol on the receiving system.
2. In the event driver protocol's Subscribers multiple, a subscriber protocol must be present whose Receiving Application matches that found in the incoming message's header.
3. If found, V*IST*A HL7 invokes that subscriber's processing routine code to process the message.

So, to process a particular incoming transaction, you need to set up the appropriate application entries, event driver protocol and subscriber protocol, and create a processing routine.

9.5 Interface Setup Example

The example used throughout this chapter is for an interface between a **V*IST*A** M system and a non-M system called the ACME system:

- The single transaction for this simple interface is for message type ORM, event type O01, HL7 V. 2.3.
- No application acknowledgment is required.
- The message is sent from the ACME system to **V*IST*A**.
- The ACME system is connected to **V*IST*A** via TCP/IP.

To support the transaction listed above, set up the following entries on the **V*IST*A** side for the interface, in the following order:

Step	Create:	Entry Name	Purpose
1.	Application	ACME	Represents the ACME side of the **V*IST*A**-ACME interface
2.	Application	VIST-AC	Represents the **V*IST*A** side of the **V*IST*A**-ACME interface
3.	Event Driver Protocol	ACME ORM\O01 EVENT	Protocol that **V*IST*A** HL7 matches to the incoming message's message type, event type, version ID and sending application, from the incoming message's header.
4.	Subscriber Protocol	VIST-AC ORM\O01 SUBS	Protocol that subscribes to the event, and whose processing routine code is invoked when a message is received.

Note that a link is *not* needed, unless **V*IST*A** needs to return an acknowledgment to the ACME system, and even then only under certain conditions. For more information on acknowledgments, please see the "Building Interfaces: Acknowledgments" chapter.

9.6 How to Process Incoming Messages

1. Develop an M processing routine to parse the message text and file the data.
2. Set the PROCESSING ROUTINE field of the subscriber protocol for the application on the **V*IST*A** side of the interface to code that executes the M processing routine.
3. If you need to respond with an ACK, include code to your processing routine to do this. This should be done as the last variable-dependent action in a processing routine. For information about generating acknowledgments, please see the "Building Interfaces: Acknowledgments" chapter.
4. Optionally set the HLERR variable if the incoming acknowledgment message's status should not be set to "Successfully Completed". Setting HLERR to some value when processing a message tells **V*IST*A** HL7 to set the status of the incoming message to "Error" rather than "Successfully Completed". If $D(HLERR) is true, the incoming message's status is set to "Error", and its "Error Message" field is set to the value of the HLERR variable.

9.7 How to Parse Message Text

To process acknowledgments and incoming messages, there are conventions you should follow to parse the incoming message text.

All messages received are stored in a multiple field of the Message Text file (#772). Each message segment is stored on its own global node (if the message segment exceeds the maximum storage length, continuation global nodes are used for the remainder of the segment).

To access a message's text, your routine should use the following special variables:

Variable	Description
HLNEXT	M code you can execute to $O through the segments of a message.
HLNODE	The current segment in the message (initially set to null).
HLNODE(n)	Continuation nodes, if any, for long segments (>245 characters). Nodes may break in the middle of a field.
HLQUIT	If not greater than zero, indicates there are no more segments to $O through.

To read each segment of message text, your routine should execute the HLNEXT variable in a loop, analyze the data stored in each HLNODE variable, and quit if the HLQUIT variable is not greater than zero. The following is an example of possible application code to step through each segment of a message's text:

```
N I,J,X
F I=1:1 X HLNEXT Q:HLQUIT'>0  D
. S X(I)=HLNODE,J=0 ;get first segment node
. ;get continuation nodes for long segments, if any
. F  S J=$O(HLNODE(J)) Q:'J  S X(I,J)=HLNODE(J)
```

9.8 Variables You Can Reference During Message Processing

Certain HL7 variables are set up and defined for programmer use when a message is being processed, in either of the following situations:

- Receiving an acknowledgment (when an event driver's acknowledgment processing routine is invoked).
- Receiving a message (when a subscriber protocol's message processing routine is invoked).

Note: You should not set these variables directly when processing a message.

Variable	Description
HL("APAT")	*Application* acknowledgment condition of the message. This should be used by a receiving application to determine the type of acknowledgment, if any, to return the application that sent the message. It will be set to the text code, i.e., AL, NE, ER or SU.
HL("CC")	Country code of the sending application.
HL("CONTPTR")	Continuation pointer, if any, from the message header.
HL("DTM")	Date/time from the incoming message's header, in HL7 format.
HL("DUZ")	If a valid **V*IST*A** access code is contained in the first component of the Security field (#8) of the message header segment, this is set to the DUZ associated with this access code from the New Person file (#200) on **V*IST*A**.
HL("ECH")	HL7 encoding characters (1 to 4 characters) to be used in assembling data into or extracting data from HL7 segments and fields. The four encoding characters are the component separator, repetition separator, escape character, and sub-component separator, in that order. The default encoding characters used by **V*IST*A** HL7 (when an application package does not define its own encoding characters) are: ~|\&
HL("EID")	IEN of the event driver protocol associated with the message.
HL("EIDS")	IEN of the subscriber protocol associated with receiving the message.
HL("ESIG")	This variable might not always exist. If a valid **V*IST*A** electronic signature code is contained in the third component of the Security field (#8) of the message header segment, this is set to the signature block printed name associated with this electronic signature code from the New Person file (#200) on **V*IST*A**.
HL("ETN")	Three character event type name from the Protocol file (#101) (e.g., A01 [Admit a Patient], O01 [Order Message], etc.).
HL("FS")	HL7 field separator character to be used in extracting fields of data from HL7 messages received, or building HL7 segments in messages sent. The field separator is a single character (e.g., ^).
HL("MID")	HL7 message control ID for the message received.
HL("MTN")	Three character message type name from the Protocol file (#101) (e.g., ADT, QRY [Query], ORU [Observation Result Unsolicited], etc.).

Variables You Can Reference During Message Processing (cont'd)

Variable	Description
HL("PID")	The HL7 processing ID for the message received. (P for production, T for Training, D for Debug.) Applications can use this to branch to "production" code if the processing ID is "P", or to other code (e.g., quit without processing), if the processing ID is "D" or "T".
HL("Q")	This variable is set to two quotation marks: "" It can be used to insert a null value in an HL7 field when building HL7 segments.
HL("RAN")	Name of the receiving application from the HL7 Application Parameter file (#771)(e.g., Radiology).
HL("SAF")	Name of the sending facility from the HL7 Application Parameter file (#771).
HL("SAN")	Name of the sending application (e.g., Radiology) from the HL7 Application Parameter file (#771).
HL("VER")	Version number of the HL7 protocol that was used to build the message being sent/received.
HLMTIENS	IEN of the message entry (File #773 for messages received over TCP links, File #772 for all others).
HLNEXT	M code an application can execute to $O through a message's segments.
HLNODE	A node from a message, provided by executing HLNEXT.
HLQUIT	A variable to check after executing HLNEXT when processing a message. If HLQUIT is not greater than zero, there is no more text to $O through.

9.9 Exception Processing

An exception occurs when an incoming message cannot be processed correctly. For example, a field may reference an entry in a table that is missing. The application processing the message must either reject it or store it as an exception for later processing.

V*IST*A HL7 does not provide an exception log. It does provide a set of entry points that facilitate application's needs to reprocess messages. To use these entry points, a processing routine should:

1. Store the IEN of the message with a problem in your own exception log. You can get the IEN from the variable HLMTIENS, which is always defined during message processing.
2. Call $$DONTPURG^HLUTIL or $$SETPURG^HLUTIL(1) (they're equivalent calls) to prevent the HL PURGE TRANSMISSIONS option from purging the message.
3. Fix the error condition that is causing the exception.
4. Call $$REPROC^HLUTIL to reprocess the message.
5. If the reprocessing is successful, call $$TOPURG^HLUTIL or $$SETPURG^HLUTIL(0) (they're equivalent calls) to once again allow purging of the message by the "Purge Messages" option.

For more information on these entry points, please see the "API Reference" chapter.

C H A P T E R

10. Building Interfaces: Acknowledgments

10.1 Acknowledgment Modes

HL7 acknowledgments come in two flavors: *original* and *enhanced* mode. Original mode is the acknowledgment mode used by version 2.1 of the HL7 standard. Enhanced mode was introduced by version 2.2 of the HL7 standard:

> The processing rules were extended in Version 2.2 of the Standard. The extensions provide a greater degree of flexibility in the way that messages can be acknowledged, as specified by several new fields in the Message Header segment. To provide backward compatibility with prior versions, the absence of these fields implies that the extended processing rules are not used. In the remainder of this section, the extended mode is called the *enhanced* acknowledgment mode; the prior version is called the *original* acknowledgment mode.[6]

V*IST*A HL7 supports V. 2.2 and higher acknowledgments (enhanced mode) as well as V. 2.1 acknowledgments (original mode).

10.1.1 Accept (Commit) Acknowledgments vs. Application Acknowledgments

Original mode only had one type of acknowledgment, an application acknowledgment, used both by the protocol software (e.g., **V*IST*A** HL7) and applications, resulting in ambiguities in certain situations.

Enhanced mode separates the acknowledgment returned by the protocol software, e.g., **V*IST*A** HL7 from the acknowledgment returned by the application. The acknowledgment returned by the protocol software is called an accept acknowledgment (also known as a commit acknowledgment):

> With a positive accept acknowledgment, the receiving system commits the message to safe storage in a manner that releases the sending system from the need to resend the message.[7]

The acknowledgment returned by application software is called an application acknowledgment:

> After the message has been processed by the receiving system, an application acknowledgment may be used to return the resultant status to the sending system.[8]

---

[6] Health Level Seven, *Health Level Seven, Version 2.3.1*, copyright 1999, p. 2-72.
[7] Health Level Seven, *Health Level Seven, Version 2.3.1*, copyright 1999, p. 2-2.
[8] Health Level Seven, *Health Level Seven, Version 2.3.1*, copyright 1999, p. 2-2.

10.1.2 ACKs vs. NAKs

Acknowledgments can be either positive or negative. A positive acknowledgment is referred to as an ACK, while a negative acknowledgment is referred to as a NAK. The difference between an ACK and a NAK is in the acknowledgment code, the first field sequence in an acknowledgment's required MSA segment. The possible values for that field in an application acknowledgment are:

Code	Meaning	Description
AA	Application Accept	Positive application acknowledgment
AE	Application Error	Error response due to the message's content or format. Error information should be included in additional segments in the response. This is a NAK.
AR	Application Reject	The application rejects the message for reasons unrelated to content or format (system is down, internal failure, etc.) While this is a NAK, the original message can probably be re-sent again in the future without changing its format.

10.2 Role of Protocols

The following protocol settings control how acknowledgments are requested and processed, and need to be set up correctly in a **V*IST*A** HL7 interface:

Event Driver Protocol

Protocol Field	Purpose
Transaction Message Type	Original (outgoing) message type, e.g., ADT.
Event Type	Original (outgoing) event type, e.g. A01.
Accept Ack Code	Puts the code to request (or not request) an accept acknowledgment in outgoing message headers (e.g., AL).
Application Ack Type	Puts the code to request (or not request) an application acknowledgment in outgoing message headers (e.g., AL).
Response Processing Routine	Name of routine to process the returned acknowledgment response, e.g., A6ARESP.

Subscriber Protocol

Protocol Field	Meaning
Response Message Type	The original message usually has a particular response message type defined by HL7 standard, e.g., ACK.
Event Type	The original message usually has a particular response event type defined by HL7 standard, e.g., A01.

10.3 Sending System: How to Request & Process Acknowledgments

10.3.1 How to Request that an Acknowledgment Be Returned

When you originate messages to recipients, if you want an acknowledgment to be returned, you need to set the acknowledgment mode in the outgoing message's header. Do this by setting two fields in the event driver protocol that is generating the message:

- ACCEPT ACKNOWLEDGMENT
- APPLICATION ACKNOWLEDGMENT

For version 2.2 and higher of the HL7 standard, acknowledgments are expected to be performed as follows, based on these two settings in the message header:

Accept Ack [MSH-15]	App. Ack [MSH-16]	Ack Mode	Behavior Requested from Receiving System
null	**null**	emulates Original	Application acknowledgment only
NE	**AL**	emulates Original	Application acknowledgment only
AL	**AL**	Enhanced	Commit & application acknowledgment (enhanced mode full two-phase commit)
AL	**NE**	Enhanced	Commit acknowledgment only
NE	**NE**	Enhanced	No acknowledgment

Note: Two other acceptable but infrequently used values for Accept Ack and Application Ack are SU (acknowledge successful completion only) and ER (acknowledge error/reject conditions only). **V*IST*A** HL7 does not currently support accept (commit) acknowledgments of SU or ER.

10.3.2 How Incoming Acknowledgments Are Matched To the Original Message

When an acknowledgment is returned by the receiving system to your application, **V*IST*A** HL7 needs to match up the incoming acknowledgment with the original message it is acknowledging. Properly formed acknowledgment messages contain the Message ID of the message they're acknowledging in the MSA segment. **V*IST*A** HL7 locates the original message and corresponding event driver protocol based on this value. This is for original and enhanced mode application acknowledgments only; **V*IST*A** HL7 generates and processes enhanced mode Commit Accepts internally.

10.3.3 How Accept (Commit) Acknowledgments Are Processed

V*IST*A HL7 processes all incoming accept (commit) acknowledgments automatically. The only acknowledgments you need to explicitly process are incoming application acknowledgments.

10.3.4 How to Process Application Acknowledgments

1. Create an M routine to process the returned application acknowledgment messages. For a description of how to parse message text, please see "How to Parse Message Text" in the "Building Interfaces: Receiving a Message" chapter.
2. Set the Response Processing Routine field of the event driver protocol originating the initial message to your acknowledgment processing routine's name.
3. Your processing routine should check the acknowledgment code returned in the MSA segment (AA, AE or AR) to determine if the response is a NAK or not. You may need to take a different processing action if what is returned is a NAK. For more information on types of NAKs, please see "ACKs vs. NAKs" earlier in this chapter.
4. Optionally set the HLERR variable if the incoming acknowledgment message's status should not be set to "Successfully Completed". Setting HLERR to some value when processing a message tells **V***IST***A** HL7 to set the status of the incoming message to "Error" rather than "Successfully Completed". If $D(HLERR) is true, the incoming message's status is set to "Error", and its "Error Message" field is set to the value of the HLERR variable.

10.4 Receiving System: How to Generate Acknowledgments

10.4.1 Accept (Commit) Acknowledgements

V*IST*A HL7 returns all required outgoing accept (commit) acknowledgments automatically. The only acknowledgments you need to explicitly generate are application acknowledgments.

10.4.2 How to Generate an Application Acknowledgment

When processing a message, one of the common actions is to generate an acknowledgment to return to a sending application. For more information about processing messages, please see the "Building Interfaces: Receiving a Message" chapter.

In the environment set up for you by **V*IST*A** HL7 when you process a message, the variable HL("APAT") contains the *application* acknowledgment condition from the incoming message's header. You should use this to determine the type of acknowledgment, if any, required to return the application that sent the message. HL("APAT") is set to the text code for the requested acknowledgment condition, i.e., AL or NE.

1. If HL("APAT") is set to "AL", you need to return an application acknowledgment.
2. Build the acknowledgment message by setting nodes of the local array HLA("HLA",I) equal to the appropriate HL7 segments. Alternately, if the response is a large message, set the acknowledgment's HL7 segments into the global array ^TMP("HLA",$J,I).
3. You are responsible for setting up MSA segment as the first segment of the message body for the ACK.
4 Set the appropriate code (AA for a positive ACK, or AE or AR for a NAK) in field sequence 1 in the MSA segment.
5. Field sequence 2 in the MSA segment must contain the message ID of the message to which you're responding. Set it from the message ID contained in the variable HL("MID"), which is set up in the processing environment for you by **V*IST*A** HL7.
6. Most ACKs only use first two MSA fields. For NAKs, you may also want to set a Text Message or Error Condition in the appropriate MSA field sequence.
7. Add all other message segments needed for your response.
8. Call GENACK^HLMA1 to send the response. Because GENACK changes variables, we recommend calling it as the last variable-dependent action in a processing routine.
9. Optionally set the HLERR variable if the incoming message's status should not be set to "Successfully Completed". You might do this if you're returning a negative acknowledgment, for example. Setting HLERR to some value when processing a message tells **V*IST*A** HL7 to set the status of the incoming message to "Error" rather than "Successfully Completed". If $D(HLERR) is true, the incoming message's status is set to "Error", and its "Error Message" field is set to the value of the HLERR variable.

These steps are valid for acknowledgments for both enhanced mode (2.2 and higher) and original mode (2.1) interfaces, including enhanced mode deferred acknowledgments.

10.4.3 Contents of the MSA Segment

Seq	Header Field	Value to Use
1	**Acknowledgment Code**	Required: AA, AE or AR (for application acknowledgments).
2	**Message Control ID**	Required: The message ID of the message being acknowledged. Obtain from HL("MID").
3	**Text Message**	Optional: Use to further describe an error condition.
4	**Expected Sequence Number**	Optional: Used in sequence number protocol (not supported by **V*IST*A** HL7).
5	**Delayed Acknowledgment Type**	Optional: For backward compatibility only.
6	**Error Condition**	Optional: User-defined error code. **Note**: The HL7 standard defines a set of message error condition codes for this field.

10.4.4 How the Acknowledgment is Addressed

V*IST*A HL7 determines the return path to address acknowledgments as follows:

Message Received Over	Return Address Used
TCP (1-phase transaction)	Returned over the open TCP connection.
TCP deferred application ack (2nd phase of 2-phase transaction)	If HL7 V. 2.3-formatted "Sending Facility" information is present in the message header, **V*IST*A** HL7 looks for a link matching that information; if found, it returns the acknowledgment over that link. If "Sending Facility" information is not present, **V*IST*A** HL7 uses the link associated with the subscriber protocol.
HLLP or X3.28	"Sending Facility" institution and domain information should not be present in message headers for HLLP and X3.28 links; these link types are typically used for COTS systems, not inter-facility messaging. In this case, **V*IST*A** HL7 uses the link associated with the subscriber protocol.
MailMan	**V*IST*A** HL7 looks for the domain information provided by MailMan. If found, **V*IST*A** HL7 looks for a link associated with the domain, and returns the acknowledgment over that link (even if the link is not a MailMan link). Otherwise, the link associated with the subscriber protocol is used.

10.4.5 Returning Responses to Dynamically Addressed Messages

V*IST*A HL7 uses the Sending Facility information if it is present in the message header so that even if the arriving message was dynamically addressed, responses can be returned to it based on the message header rather than on an irrelevant fixed link definition in the subscriber protocol.

10.5 LLP-Specific Acknowledgment Behaviors

10.5.1 Acknowledgments over VA MailMan Links

There are no special issues for sending acknowledgments over MailMan links.

10.5.2 Acknowledgments over HLLP and X3.28 Links

X3.28 links gracefully and reliably originate and process new messages and responses over the same connection.

HLLP, on the other hand, can be more problematic. Either side of an open HLLP link can originate a new transaction over the open connection to the other system. Some COTS systems, however, when waiting for a response, will accept the next incoming message as the response even if it is a completely new transaction. This causes a loss of synchronization.

Because of the synchronization problems inherent in HLLP, we recommend that two serial lines be used if both sides of the interface can originate new messages. One line is for **V*IST*A** to initiate messages and receive the corresponding acknowledgments; the other line is for the other system to originate messages and receive acknowledgments. Most commercial vendors with HLLP interfaces also recommend the use of two lines in this situation to prevent synchronization problems.

10.5.3 Original Mode Acknowledgments over TCP Links

In Original mode, the receiving application is always expected to return a response (i.e., an application acknowledgment). This response may or may not contain data, but always contains an ack. For example, a query response will contain both data and an ack message.

Expected Behavior, Original Mode Acknowledgments over TCP/IP

Sending System	Receiving System
1. Open Connection. 2. Send Message.	
	3. Process Message. 4. Send Query Response or General Ack (AA, AE, AR).
5. Process Response. If not a positive ack, retransmit (go to step 2.) 6. Drop connection if link is non-persistent, and there are no additional new transactions to originate.	

10.5.4 Enhanced Mode Ack with Full Two-Phase Commit over TCP Links

For Enhanced mode acknowledgments with full two-phase commit *over TCP links*, a Commit Ack is always returned immediately over the current connection, and the application acknowledgment is always be deferred, and performed as a separate transaction.

Expected Behavior, Enhanced Mode Two-Phase Commit over TCP/IP

Phase I	
Sending System	**Receiving System**
1. Open Connection. 2. Transmit Message.	
	3. Receive Message. 4. Validate Header. 5. Store on local system. 6. Transmit "Commit Accept Ack" (MSA contains 'CA').
7. Receive Commit Ack. 8. Process/validate Commit Ack. If not positive, retransmit (go to step 2.) 9. Repeat steps 2-8 for any remaining new transactions. 10. Drop connection if link is non-persistent, and there are no additional new transactions to originate.	

Phase II	
Sending System	**Receiving System**
	1. Open Connection. 2. Transmit Application Response. Acknowledgment mode in header must be AL/NE or NE/NE.
3. Receive Application Response. 4. Validate Header. 5. Store on local system. 6. If MSH-15='AL', transmit "Commit Accept Ack" (MSA contains 'CA').	
	7. If Ack mode of application response was NE/NE, processing has completed for this transaction. Otherwise, receive, validate and process Commit Accept Ack. If not positive, retransmit (go to step 2.) 8. Drop connection if link is non-persistent, and there are no additional new transactions to originate.

C H A P T E R

11. Advanced Interface Issues

11.1 Sending Facility Field in the Message Header

The role of the Sending Facility (MSH-4) message header field it to identify the facility sending the message.

In an environment where messages are dynamically addressed between 150+ VA facilities, knowing the sending facility of a message is important so that you know where to return acknowledgments. It is impractical to set up enough point-to-point protocol definitions, with fixed links, to represent the number of possible relationships between 150+ facilities. So, for dynamically addressed messages to **V***IST***A** facilities, for the purpose of addressing acknowledgments, it becomes critical to use the Sending Facility field.

11.1.1 How to Put the Sending Facility in the Message Header

On the sending system, when generating a message, to populate the Sending Facility field in the header, set the subscriber protocol's "Sending Facility Required?" field to YES.

On the sending system, **V***IST***A** HL7 populates the field in one of two ways:

1. From the HL7 site parameters for Current Domain and Current Institution, or
2. From the sending application's Facility Name field; if defined, it overrides the site parameter).

For **V***IST***A**-to-**V***IST***A** interfaces, we recommend that the sending application's Facility Name be left null so that the defaults from the site parameters are used. For COTS interfaces, you may use sending application's Facility Name field instead.

11.1.2 How the Sending Facility Field is used on Receiving Systems

If the Sending Facility Required? field in the subscriber protocol on the receiving system is set to YES, a valid value in the message header for Sending Facility must be present or the message will be rejected. When required, the value is considered valid if it meets the following criteria:

- If, on the receiving system, the sending application's Facility Name field is null, a link on the receiving system must be associated with the same domain and institution information that is found in the message header. Otherwise, the message is rejected. If a matching links is found, it is the link used to return any expected asynchronous acknowledgment.
- If, on the receiving system, the sending application's Facility Name field is non-null, its value must match the value found in the message header.

11.2 Receiving Facility Field in Message Header

The role of the Receiving Facility field in a message header is to ensure that when the message is received, it is received at the appropriate facility.

11.2.1 How to Put the Receiving Facility in the Message Header

On the sending system, when generating a message, the Receiving Facility field in the header is populated if the subscriber protocol, on the sending system, has its Receiving Facility Required set to YES.

On the sending system, the value to populate the field with is pulled from:

1. Receiving Application's Facility Name field (if populated), or
2. Subscriber protocol's link, if the link is associated with a domain and institution. **Note:** Only VA* namespaced links should contain this information.

For COTS interfaces, you may set the Receiving Facility field from the receiving application's Facility Name. For **V*IST*A**-to-**V*IST*A** interfaces, we recommend that the receiving application's Facility Name be left null so that the default from the subscriber protocol's link is used.

11.2.2 How the Receiving Facility Field is used on Receiving Systems

On the receiving system, if the subscriber protocol's Receiving Facility Required? field is set to YES:

- The value in the header for receiving facility must be present. If not, the message is rejected.
- In addition, the value in the header must match either a) the Current Domain and Current Institution HL7 site parameters, or, b) if defined in the subscribing application, that application's Facility Name field. If there is a mismatch, the message is rejected.

11.3 Same-System Interfaces

A same-system interface is one in which an HL7 message is delivered from one **V*IST*A** application to another, *on the same system*. To set a same-system interface up:

1. Do not enter a link for either application's subscriber protocol. Absence of a defined link for a subscriber protocol means that the subscriber protocol is being used for a same-system exchange.
2. Otherwise, set up the interface as you would any other interface.

11.4 V*IST*A-to-V*IST*A Interfaces

When a given transaction can be conducted with 150+ **V*IST*A** facilities, that interface is no longer "point-to-point". Instead, it involves many possible permutations of paths between facilities. For such a situation, the fixed "point-to-point" method of **V*IST*A** HL7 interfacing, in which the subscriber protocol has a fixed link determining the return path for any given transaction, no longer holds water, since that subscriber protocol could be invoked for 150+ different return paths.

You wouldn't want to create a separate pair of event driver and subscriber protocols for every possible sending facility/receiving facility combination. Fortunately, **V*IST*A** HL7 provides ways to avoid this scenario. In particular, after patch HL*1.6*57, the same subscriber protocol and event protocol can be distributed to various systems, and can be used to both send and receive (initiate and/or process) the same transaction, between any and all **V*IST*A** facilities.

11.4.1 Same Set of Protocols Can Be Exported to Sending and Receiving Systems

You can now export the *same* set of protocols to multiple **V*IST*A** systems, independent of the role of a given system as sender or receiver. The same protocols can be used either to generate a transaction (send) or to process a transaction (receive). This is true even in the case where the same transaction is initiated going both ways between the two systems, when both systems act sometimes as the sender and sometimes as the receiver.

To export the same set of protocols to sending *and* receiving systems, populate all "sender" and "receiver" fields for your subscriber and event driver protocols, even if a field is not explicitly needed on either a sender or receiver system. The goal is to have a uniform transaction definition on all systems. The transaction definition consists of the rules of engagement between the two systems/applications.

For example, the subscriber protocol on the sender system does not need to have its Processing Routine field populated, because the processing routine for subscriber protocols is only executed on the receiver system. However, if you set this field anyway, you can export the same subscriber protocol to both sender and receiver systems.

Changes introduced by patch HL*1.6*57 supporting this new capability include:

- **Event Driver Protocol:** On the sending system, when a message is being generated, its message type and event type both come from the fields in the event driver protocol only (Transaction Message Type and Event Type). Messages are delivered to each of the subscribers found in the new Subscribers multiple. The Item multiple is no longer used for HL7 transactions.
- **Subscriber Protocol:** On the sending system, subscriber protocols are now used wholly for parameters specific to reaching a particular receiving system (e.g., receiving application, and, in the absence of routing logic, the link to use). And the Response Message Type and Event Type fields are *not* used on the sending system, but rather are only used on the receiving system, to define the message type and event type for responses.

11.4.2 Interfaces Prior to Patch HL*1.6*57

No conversion of *properly defined* interfaces built prior to patch HL*1.6*57 is necessary to take advantage of features introduced by patch HL*1.6*57. Installation of the patch converts existing protocols and links. Moreover, installation of older protocols and links on post-HL*1.6*57 systems invokes an automatic conversion, via the Kernel Installation and Distribution System (KIDS). **Note:** To do this, KIDS must be updated with Kernel patch XU*8*131.

Some interface setups, however, may be improperly defined, but most will work anyway, except under unusual circumstances.

Example of Correctly Defined Event driver protocol (for both Sending and Receiving System)

```
NAME: JC ADT SENDER                     TYPE: event driver
  CREATOR: HL7Developer, One
        SENDING APPLICATION: JC NXT SEND
  770.3 TRANSACTION MESSAGE TYPE: ADT
  770.4 EVENT TYPE: A01
        PROCESSING ID: PRODUCTION
        ACCEPT ACK CODE: AL
        VERSION ID: 2.2
        RESPONSE PROCESSING ROUTINE: Q
        SENDING FACILITY REQUIRED?: YES
        RECEIVING FACILITY REQUIRED?: NO
        SUBSCRIBERS: JC ADT RECV
```

Example of Correctly Defined Subscriber Protocol (for both Sending and Receiving System)

```
NAME: JC ADT RECV                       TYPE: subscriber
  CREATOR: HL7Developer, One
        RECEIVING APPLICATION: JC KRN REC
  770.3 TRANSACTION MESSAGE TYPE: ADT
  770.4 EVENT TYPE: A01
        PROCESSING ID: PRODUCTION
        VERSION ID: 2.2
  770.11 RESPONSE MESSAGE TYPE: ACK
        PROCESSING ROUTINE: Q
        SENDING FACILITY REQUIRED?: NO
        RECEIVING FACILITY REQUIRED?: NO
```

11.4.3 Validate Interfaces Option

To assist with validating interfaces, you can use the new "Validate Interfaces" option to examine an interface's application, event driver protocols and subscriber protocols. It generates a report of any potential problems it finds. Use the report as a guide for making modifying your interface. If your setup results in fewer protocols, be sure to remove the unneeded protocols. Remember: If the report on a pre-HL*1.6*57 interface reports errors, it does NOT mean the interface has been broken or will stop working because of patch HL*1.6*57. Use the report as a guide only for upgrading or for troubleshooting an existing post-HL*1.6*57 interface.

11.4.4 Link Type to Use for Inter-Facility Messaging

We recommend that **V*IST*A** facility to **V*IST*A** facility messaging should be done over TCP links, rather than using VA MailMan links. We do not recommend the use of MailMan for any new interfaces, and recommend converting old interfaces from MailMan to TCP, for better security and improved efficiency. Valid TCP links for all **V*IST*A** facilities were introduced by patch HL*1.6*39, and for the near future will be maintained through subsequent patches.

11.4.5 Response Return Path for Multi-Facility Interfaces

In an environment where a particular message transaction is exchanged between 150+ facilities, rather than between 2 points, the method of specifying the return path as the link entry in the subscriber protocol no longer works (since the subscriber protocol may be invoked to return the message to any of the 150+ facilities).

The easiest way around this is if your transactions are one-phase (accept acknowledgment or application acknowledgment required, but not both) over TCP. In this case, the acknowledgment is always returned over the open connection for which the original message was delivered, so setting the appropriate return path is not a problem.

Otherwise, set up your transactions to require the use of the "Sending Facility" field in the message header. This allows the return path to be obtained from the message header. There are other reasons why doing this is a good idea, even for one-phase transactions over TCP. For more information on how to do this, please see "Sending Facility Field in the Message Header" earlier in this chapter.

11.4.6 Transactions between the Same Application on Different Systems

In the case where you are exchanging the *same* message between instances of the *same* application, but on different systems, choose the *same* application entry for the event driver protocol (sending application) as you do for subscriber protocol (receiving application) in the interface.

11.4.7 Guarding Against Processing Messages Intended for Other VA Facilities

You can put checks in place to protect against processing messages at one VA facility intended for a different facility.

To protect against this, you should:

1. Set the subscriber protocol's "Receiving Facility Required?" field to YES, on both sending and receiving systems.
2. Make sure the sending application's Facility Name field is set to null, on both sending and receiving systems.
3. Make sure the HL7 site parameters at sending and receiving sites have Current Institution and Current Domain set correctly to reflect each site's appropriate domain and institution.
4. Make sure the links used to reach other VA facilities at sending and receiving sites have appropriate associations for Domain and Institution. If you use the TCP links maintained by **V*IST*A** HL7 (first released in patch HL*1.6*39), these associations are set up correctly for you.

Doing this ensures that:

- Outgoing messages generated at a **V*IST*A** facility will have that facility's domain and institution concatenated together in the message header's Sending Facility field, and
- If a message is ever received at an unintended **V*IST*A** facility, **V*IST*A** HL7 at that facility will reject the message.

For more information, please see the "Sending Facility Field in the Message Header" section earlier in this chapter.

11.5 Dynamic Addressing

Unconditional broadcasting (the only addressing mechanism available prior to patch HL*1.6*14) is usually sufficient to distribute messages between a **V*IST*A** system and other systems within a single site, and between a **V*IST*A** system and a single destination, such as the Austin Automation Center or the HEC.

However, consider the case of a message that may need to be sent from one medical center to another medical center (which one is not known in advance), but not to *all* medical centers. The use of a fixed-in-place set of subscriber protocol relationships, representing fixed point-to-point interface definitions, becomes unwieldy in this case. There would need to be one subscriber protocol for every target medical center, and only one medical center receiving the message would actually use it. In this case, unconditional broadcasting results in a mushrooming and unnecessary network traffic, as well as extremely complicated interface design.

To solve these problems, patch HL*1.6*14 introduced *dynamic addressing*:

- Recipients *on remote systems* for a message can be set dynamically for each message at run-time.
- Routing of messages to remote systems can be determined based on an individual message's content and business rules, not just on a broad event point.

11.5.1 Dynamic Addressing Via the HLL("LINKS") Array

To dynamically address messages, use the HLL("LINKS") array. By setting nodes in this array, you can dynamically specify one or more recipients (on remote systems only) for an HL7 message. Add one node per recipient to the HLL("LINKS") array. The expected format for each node is:

```
HLL("LINKS",n)="DESTINATION PROTOCOL^DESTINATION LOGICAL LINK"
```

Subscript *n* is a unique, arbitrary integer subscript. Any additional ^-pieces beyond piece 2 are ignored for any HLL array node.

For example:

```
HLL("LINKS",1)=RH ADT SUBS SEND^TEST LLP
```

The pieces of an HLL("LINKS",n) act as follows:

Piece	Piece Contents	Description
1	Destination Protocol (name or IEN)	(mandatory) The subscriber protocol associated with the application that will receive and process the message.
2	Destination Link (name or IEN)	(mandatory) The link that identifies the path to the target system for the message. The destination must be on a different system. The link can be any link type.

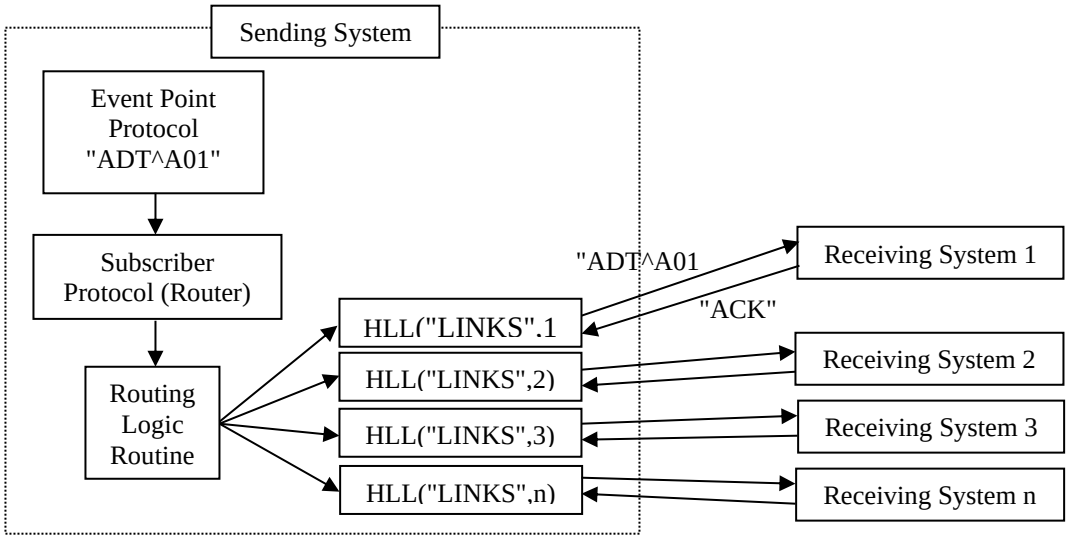
You can dynamically add recipients to the HLL("LINKS") array for an **outbound** HL7 message at the following points:

- When you are preparing a message for sending. Set nodes in HLL("LINKS") prior to calling GENERATE^HLMA to send the message.
- When, on a sending system, you add a subscriber protocol as a subscriber to the event driver protocol generating the outbound message, and set up the subscriber protocol as a router. See "*How To Create a Router Protocol"* below.

Note: You cannot dynamically address to a receiving application on the same system.

11.5.2 How to Create a "Router" Protocol

You can set up a subscriber protocol, on the *sending* system, to act as a router. A router protocol's only purpose is to dynamically add message recipients for the outbound message. It can examine the message text, and conditionally route messages to interested systems based on message content and business rules.



To set up a Subscriber protocol to be a Router:

1. Create a routing routine. The routine can examine each line of a message by executing 'X HLNEXT' in the documented manner. Using the environment variables HL("FS") (field separator) and HL("ECH") (encoding characters), the routing routine can parse each segment for the appropriate fields to screen on to determine who the supplemental addressees are. At the time the routine is executed, the message *header* has not been created yet; the router is scanning the original message *body* in the HL7 Message Text file.
2. To add additional message recipients, the routing routine should add one node to the HLL("LINKS") array for each application/system it determines should be a recipient. Remember that the format of HLL("LINKS") is:

   ```
   HLL("LINKS",n)="DESTINATION PROTOCOL^DESTINATION LOGICAL LINK"
   ```

   where name or IEN can be used for both protocol and link.
3. Use the Interface Toolkit to do the initial set up of the router protocol as a standard Subscriber protocol.
4. Set the ROUTING LOGIC field of the router protocol to a routine that you will use to perform routing. With ROUTING LOGIC defined:
   - The router protocol's PROCESSING ROUTINE, if any, is ignored on the sending system.
   - The router protocol's LOGICAL LINK field, if any, is ignored on the sending system.
5. Attach the new "router" Subscriber protocol to an event driver protocol. It now performs routing for that event point's messages.

11.5.3 Responding to Dynamically Addressed Messages

On a receiving system, the subscriber protocol a message is addressed to does not have to be attached to any particular event driver protocol for the message to be processed.

Because there is no linkage to an event driver protocol, another means is needed to determine who to address replies to:

- For dynamically addressed messages received over serial connections and TCP/IP, the channel is assumed to be open and bi-directional. The response is immediate by default for both commits and application ACKs.
- For dynamically addressed messages received through Mailman, commit responses are addressed by **V*IST*A** HL7 using the Mailman environment variable, XMFROM (which contains the domain of origin, and can be resolved to the equivalent link).
- For dynamically addressed messages received through MailMan, to send an application ACK (typically needed for queries only), set a node in HLL("LINKS") based on the contents of the MailMan XMFROM variable, prior to calling GENACK^HLMA1. This in effect uses dynamic addressing to reply to a dynamically addressed message.

11.5.4 What About Local Recipients?

Dynamic addressing **cannot**, at this time, route messages to applications on the local system. This means that you must still use the unconditional broadcast mechanism to send messages to local recipients.

If a message needs to be processed/filed on the current (sending) system as well as being routed dynamically to remote systems, set up a separate Subscriber protocol for each local recipient. It should be a traditional V. 1.6 style Subscriber protocol, with its LOGICAL LINK field null (so that the message is received on the sending system) and its PROCESSING ROUTINE set to the routine that will process the (inbound) message on the current (local) system.

11.5.5 What if No Recipients are Selected?

The router protocol may determine that no recipients are interested in the message being routed. In this case, HLL("LINKS") will be undefined. If no other recipients are defined for the message, the message will not be delivered.

11.5.6 Avoiding Duplicate Messages

Duplicate messages occur when the same message text is sent to the same **application** on the same **system** more than once. For dynamically addressed messages, **V*IST*A** HL7 eliminates the possibility of duplicate destinations. If you set duplicate nodes into the HLL("LINKS") array, those duplicates are filtered out by **V*IST*A** HL7.

However, if a Subscriber protocol has a fixed link definition (not a router) it is possible for duplicate messages to occur in the following, highly unlikely situations:

- If another Subscriber protocol to the same event point has the same link definition and receiving application.
- If another Subscriber protocol to the same event point is a router, and it dynamically addresses a message to the same link and recipient.

11.5.7 Using a Router to Modify an Existing Message

Sometimes you want to subscribe a new application to an existing message generated on your system, but the new application needs the message in a modified format. If you cannot change the original message, you can use a Router to copy and modify the original message as follows:

1. Create a router protocol that subscribes to original event point protocol.
2. The router's processing routine should copy the original message body into ^XTMP (with any needed modifications), and then pass that data location via TaskMan to another routine. **Note:** TaskMan must be used because **V*IST*A** HL7 is not re-entrant.
3. The routine invoked via TaskMan calls GENERATE^HLMA, uses a new, different interface with different event driver and subscriber protocols, and conducts a new transaction with the modified message copy to the new application.

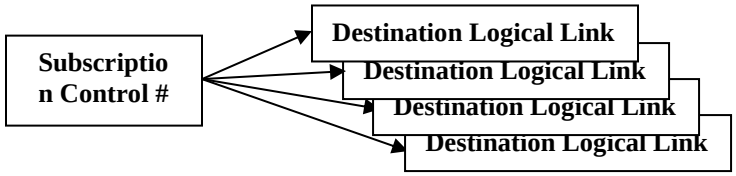
11.6 Subscription Registry

The subscription registry is a convenient location where potential message recipients can be stored. An API is provided to store and retrieve subscribers. It is particularly useful as a location where applications can maintain a list of subscribers for dynamically addressed messages.

The subscription registry does not need to be involved for dynamic addressing to be performed. If application code or routing logic can determine the recipients of a message (destination link) on its own, it can go ahead and set nodes in HLL("LINKS") without consulting the subscription registry.

Each entry in the subscription registry consists of a subscription control number and a multiple containing destination subrecords (and related information) for each subscriber. This structure allows a single subscription control number to hold multiple destinations.

Subscription Registry Record



Applications are responsible for knowing which records in the subscription registry pertain to their application. For example, CIRN has a pointer from the Patient file to the subscription registry. This is the mechanism CIRN uses to associate a particular subscription registry control number (and all destinations within that control number) with a particular patient.

The subscription registry should only be updated through the following API calls (described in the "API Reference" chapter:

- Create new entries in the subscription registry with the $$ACT^HLSUB entry point.
- Add destination records to a subscription registry entry using the UPD^HLSUB entry point.
- Retrieve active destination records from a subscription registry entry using the GET^HLSUB entry point.

Purging functionality for the registry is not currently supported.

11.6.1 SUBSCRIPTION CONTROL File

The subscription registry is physically stored in the Subscription Control File (#774). Its file structure is as follows:

Subscription Registry, Top Level
Create records with $$ACT^HLSUB

Field	Field Name	Description
.01	**Number**	Subscription number.
.02	**Package**	The application or package responsible for creating this subscription.
.03	**Description**	Free text description.

Subscription Registry, Destination Multiple (Field #1, Subfile #774.01)
Create/modify subrecords with UPD^HLSUB

Field	Field Name	Description
.01	**Destination**	Full network path to the receiving application, in a format similar to a mail address: RECEIVING APP@DOMAIN or RECEIVING APP@LOGICAL LINK. The contents of this field are automatically created by the UPD^HLSUB call. **Note:** Domain is at this time reserved for future use.
1	**Receiving Application**	Place to set the receiving application that is the subscriber.
2	**Domain**	Reserved for future use.
3	**Logical Link**	The HL7 1.6 address of the receiving system (pointer to the HL Logical Link file).
4	**Type**	Subscription type (arbitrary; mainly for use by CIRN). Current types supported are: 0=patient descriptive only; 1=patient clinical and descriptive; 2=other (locally defined). Non-CIRN applications should probably use type 2.
5	**Creation Date/Time**	Date the destination subrecord was created.
6	**Activation Date/Time**	Date the subrecord should be considered active. GET^HLSUB only returns records considered active.
7	**Termination Date/Time**	Date the subrecord should be considered inactive. GET^HLSUB only returns records considered active.
8	**Modification Date/Time**	A multiple containing the history of modifications to this subscription registry record.

11.6.2 Subscription Registry Case Study: CIRN

The first use of the subscription registry is by the CIRN package. CIRN provides message routers that scan messages, dynamically forwarding each message to all subscribers of the patient the message pertains to. This allows clinical information to be updated appropriately when a patient is seen at several facilities.

CIRN uses the subscription registry in the following manner:

1. CIRN creates an entry in the subscription registry with $$ACT^HLSUB for each patient it processes. A pointer from the Patient file to the Subscription Control file for the patient in question maintains the link between patient and subscription registry record.
2. When another application/facility becomes interested in a patient (e.g., the patient is seen at another facility) that application sends a subscription message to the patient's primary facility requesting to subscribe to that patient. When the message is processed, a call is made to UPD^HLSUB to add/update the destination for that patient's subscription registry record.
3. CIRN "Patient" router subscriber protocols subscribe to most Event driver protocols. The routers scan outgoing messages, parsing out the patient name. They dynamically route messages pertaining to a particular patient to all destinations listed in that patient's subscription registry record. The routers use GET^HLSUB to retrieve destinations for a particular patient.

CIRN routes messages based on the patient (implemented by a pointer from the Patient file to the Subscription Registry). Other applications should maintain their own associations with records they add to the subscription registry (one way is by pointing from one of that application's files to the subscription registry).

11.7 HL7 Batch Messages

A batch message is delivered as a single HL7 message, but contains one or more HL7 messages inside. **V*IST*A** HL7 handles batch HL7 messages in a different way than non-batch HL7 messages. For more information about batch HL7 messages, please see the "Control/Query" chapter of the *Health Level Seven* standard.

11.7.1 Send Batch Message Procedure

The procedure to create a batch HL7 message in **V*IST*A** is different from the procedure to send an ordinary (non-batch) HL7 message. To send a batch message:

1. Create the batch message (Do once per batch.)

   This is the single message that will be used to transmit the entire batch.

   a. Call:

      ```
      INIT^HLFNC2("BATCH PROTOCOL",.return_array)
      ```

      This initializes variables to build the batch message.

   b. Call:

      ```
      CREATE^HLTF(.HLMID,.MTIEN,.HLDT,.HLDT1)
      ```

      This creates a message stub in the HL7 Message Text file (IEN is returned in MTIEN) and reserves a batch message id (returned in HLMID).

2. Create and store each message that will be in the batch.

   You must fully construct the batch message before you can send it to HL7.

   Depending on the design of your application, each individual message destined to be part of a single batch message could be triggered by multiple concurrent processes, over the course of one or more days. If this is the case, you must store HLMID, MTIEN, and a batch message counter such that they are available to all processes that could add a message to the batch. Also, you must construct the batch message in a location such as ^XTMP or another translated global that each process can access.

   For each message that you want to add to a batch:

   a. Call:

      ```
      INIT^HLFNC2("SERVER PROTOCOL",.HLARRAY)
      ```

      This initializes variables to build the individual message.

   b. Retrieve the batch message id (HLMID), batch message counter and location of the under-construction batch message in ^TMP or ^XTMP.

   c. Increment the batch message counter (the message sequence should start at 1 and increment sequentially.)

   d. Create a new message header segment for the individual message to append to the batch. Call:

      ```
      MSH^HLFNC2(.HLARRAY, HLMID_"-n",.HLRES,SECURITY)
      ```

where n is the batch message counter.

e. Append the text for the new message to the message header (returned in HLRES above) to fully construct the new message.

f. Append the fully constructed new message to your "under construction" batch message in ^XTMP or other translated global.

3. Transmit the batch message when it is completed.

   When all messages have been added to the batch message you are constructing, you can pass it to **V*IST*A** HL7 for delivery.

   a. Load the fully constructed batch message from its current location into ^TMP("HLS",$J,I), where I is sequential, starting at 1.

   b. Call:

      ```
      GENERATE^HLMA("SERVER PROTOCOL","GB",1,.MYRESULT,MTIEN,.MYOPTNS)
      ```

      where MTIEN is the IEN of the batch message entry in the HL7 Message Text file.

11.7.2 Receive/Process Batch Message Procedure

When a *batch* HL7 message is received by **V*IST*A** HL7:

1. The Batch Header segment (BHS) is checked for validity. If an error is found, an HL7 acknowledgment message is sent to the sending application to report the error ("Commit Reject" or "Commit Error" for enhanced mode, "Application Error" for original mode).
2. If the header is valid, the BATCH PROCESSING ROUTINE for the transaction is invoked.

You have complete control over processing for each message contained within a batch message. **V*IST*A** HL7 does not pre-process the header, nor link the messages to any particular protocol or processing routine.

In some cases, you may need to send a batch acknowledgment when you receive a batch message. A batch acknowledgment is a single batch message that, within itself, contains individual acknowledgments to messages within an original batch message.

11.7.3 To send a batch acknowledgment:

While processing a batch message, you can send a batch acknowledgment. To do this:

1. Create the batch acknowledgment message (Do once per batch acknowledgment.)

   This is the single message that will be used to transmit the entire batch acknowledgment.

   a. Call:

      ```
      CREATE^HLTF(.HLMID,.MTIEN,.HLDT,.HLDT1)
      ```

      This creates a message stub in the HL7 Message Text file (IEN is returned in MTIEN) and reserves a batch message id (returned in HLMID).

2. Create and store each acknowledgment that will be in the batch.

   You must fully construct the batch acknowledgment message before you can send it to HL7.

   For each message that you want to add to a batch:

   a. Increment the batch message counter (the message sequence should start at 1 and increment sequentially.)

   b. Create a new message header segment for the individual message to append to the batch. Call:

      ```
      MSH^HLFNC2(.HLARRAY, "HLMID-n",.HLRES,SECURITY)
      ```

      where n is the batch message counter.

   c. Append the text for the acknowledgment to the message header (returned in HLRES above) to fully construct the acknowledgment.

   d. Append the fully constructed acknowledgment to your "under construction" batch acknowledgment message in ^XTMP or other translated global.

3. Transmit the batch acknowledgment message when it is completed.

   When all acknowledgments have been added to the batch acknowledgment message you are constructing, you can send it.

   a. Load the fully constructed batch message from its current location into ^TMP("HLS",$J,I), where I is sequential, starting at 1.

   b. To send the acknowledgment, call GENACK^HLMA1:

      ```
      GENACK^HLMA1(EID,HLMTIENS,EIDS,ArrayType,1,.MYRESULT,
      MTIEN,.MYOPTNS)
      ```

      where MTIEN is the IEN of the batch message entry returned by the initial CREATE^HTLF call.

      Note: Because GENACK^HLMA1 changes variables, we recommend calling it as the last variable-dependent action in a processing routine.

   c. Quit to pass control back to **V*IST*A** HL7.

11.8 Z Entities and V*IST*A HL7's "Standard" Tables

11.8.1 HL7 Entities

The following files are maintained by the **V*IST*A** HL7 package for HL7 entities:

File Name	Current Contents	Updateable for Z Entities?
HL7 Message Type	ACK, ADR, ADT, etc.	Yes; contact **V*IST*A** HL7 team
HL7 Event Type Code	A01, A02, A03, etc.	Yes; contact **V*IST*A** HL7 team

11.8.2 V*IST*A HL7 and Z Entities

You can have Z entities for Segments, Events, Message Types and Fields. If you need to use Z entities in a **V*IST*A** HL7 interface, you need to:

1. Register with Z entities with the **V*IST*A** Data Systems and Integration (VDSI) Messaging Administrator.
2. If the Z entity is a Message or Event type, contact the **V*IST*A** HL7 team to have them update the HL7 Message Type or HL7 Event Type Code file with the Z Message or Z Event Type, and issue the update as a **V*IST*A** HL7 patch.

In addition, the following files are also provided by **V*IST*A** HL7 for HL7 entities, but are not checked by **V*IST*A** HL7 at this time (so Z entities do not need to be placed in these files):

- HL7 Data Type
- HL7 Field
- HL7 Segment Type

11.8.3 HL7 Tables

The following files are maintained within the **V*IST*A** HL7 package to hold HL7 tables:

File Name	Current Contents	Local Entries Allowed?
Country Code	"US", "USA"	No
HL7 Accept/Application Ack Condition	AL, NE, ER, SU	No
HL7 Acknowledgment Code	AA, AE, AR, CA, CE, CR	No
HL7 Version	2.1, 2.2, 2.3, 2.3.1	No

11.9 HL7 Queries

The HL7 standard outlines several query message types:

Query Type	Msg Type	Description
Original Mode	QRY	Generalized query
Embedded Query Language	EQQ	Supports free-form selects statements, using a query language, e.g., SQL
Virtual Table	VQQ	Supports queries against database tables
Stored Procedure Query	SPQ	Enables execution of a stored procedure on another system
Event Replay Request	ERQ	Request data formatted as an event replay response

The standard also outlines several query response message types:

Response Type	Msg Type	Description
Generalized Display Response	DSR	Responding system formats data for output to a display device
Tabular Response	TBR	Responding system formats data in relational format (rows and columns)
Event Replay Response	RQQ	Responding system formats data on the basis of an application-specific segment-oriented message

According to the HL7 standard,

> The variety of potential queries is almost unlimited. There was no attempt here to define a Standard that would cover every possible query. Instead, the Standard embraces the most common queries that are likely to occur in a hospital. For each common query, there is a corresponding … [query response]. [9]

11.9.1 Level of Query Support

V*IST*A HL7 does not provide explicit support any of these query types. For example, although software implementing SQL queries against VA FileMan data has been locally installed at many **V*IST*A** facilities, there is no generic query parser for VA FileMan data, such as SQL, in place at *all* **V*IST*A** facilities.

[9] Health Level Seven, *Health Level Seven, Version 2.3.1*, copyright 1999, p. 2-81.

11.9.2 Implementing Application-Specific Queries

The key feature to consider if implanting your own application-specific queries is that the response message type/event type pair will be different from the original message's response type/event type pair. **V*IST*A** HL7 supports this ability.

When setting up event driver protocols and subscriber protocols for a query, the following key fields model the message type and event type of the query and response messages:

Event Driver Protocol

Transaction Message Type:	QRY	*(query message type)*
Event Type:	Q02	*(query event type)*

Subscriber Protocol

Response Message Type:	DSR	*(response message type)*
Event Type:	Q03	*(response event type)*

The example above shows how to set up an original mode query (QRY/Q02) and a display response (DSR/Q03) in a **V*IST*A** HL7 interface. If the query is between two **V*IST*A** systems, install the same event driver and subscriber protocols, with the same settings, on both systems.

V*IST*A HL7 will delivery query transactions; at the current time, it is up to you to implement the processing routines that will process query requests, and parse query results.

11.10 Continuation Pointers

Continuation pointers, as defined by the HL7 standard, allow you to "break up" a message into several separate messages, and use a continuation pointer in each subsequent message's header to link the message to its predecessor.

You might need to use a continuation pointer if, for example, a large message (e.g., batch) is being sent over a link LLP type which has a size limitation, such as HLLP (whose message text size limit, including control characters required for the protocol, is 99999 bytes). TCP links don't have an inherent size limitation.

To break up a transmission into multiple messages using continuation pointers:

1. You need to keep track of how you break up each message. Each time you send a message with GENERATE^HLMA or DIRECT^HLMA, get its Message ID from the result parameter.
2. When you send another message that is part of the chain, pass in the previous message's message ID to be the continuation pointer in input variable HLP("CONTPTR").
3. When you receive messages and expect that there may be continuation pointers, check for the presence of variable HL("CONTPTR"). If present, use it with each incoming message to assemble the whole message from the individual messages received.

11.11HL7 Sequence Number Protocol

V*IST*A HL7 does not support HL7 Sequence Number Protocol.

11.12 Writing Interface Specifications

When constructing an interface, it is useful to formalize and document the interface by writing an interface specification. The specification is valuable for maintenance purposes, and as a reference for any changes you need to make to the interface later.

In addition, interface specifications are essential for documenting negotiated elements of transactions between **V*IST*A** and COTS systems, as well as for registering Z entities with the **V*IST*A** Messaging Administrator.

Examples of existing **V*IST*A** HL7 interface specifications include:

- *Laboratory Electronic Data Interchange (LEDI) V. 5.2 Installation Guide*
- *Laboratory and UI LEDI HL7 Interface Specification (*http://www.vista.med.va.gov/softserv/clin_nar.row/lab/visnnb/ledihlst.htm*)*
- *Outpatient Pharmacy V. 7.0 Technical Manual/Security Guide*
- *PIMS V. 5.3 Technical Manual*
- *Radiology/Nuclear Medicine V. 5.0 Technical Manual*

11.13 Securing Your Interface

11.13.1 Security and Link Types

Link Type	Security Issues
HLLP and X3.28	There are very few if any security issues associated with HLLP and X3.28 interfaces over serial lines. The security of the interface is as secure as the physical security of the serial connection and network connecting the serial device to the host system.
TCP	There *are* some security issues to consider with TCP/IP HL7 interfaces. The security of the interface is as secure as the network over which messages are being exchanged. Security for VA's internal network is maintained by VA's firewall. **V*IST*A** HL7 currently assumes that systems within the firewall are trusted systems.
MailMan	The most significant security issues arise from the use of MailMan interfaces. The security of the interface is *only as secure as is email*. Any email-connected system is capable of conducting an HL7 message transaction with a **V*IST*A** system. We do not recommend the use of MailMan links for interfaces at this time.

11.13.2 Sending Facility and Receiving Facility in Message headers

Consider using the "Sending Facility Required?" and "Receiving Facility Required?" fields in subscriber protocols. Used properly, they help ensure that:

- Messages intended for other systems aren't inadvertently processed on the wrong system.
- Messages are only processed if the return path (based on domain and institution in the message header) matches an active link on the receiving system.

For more information about these fields, please see "Sending Facility Field in the Message Header" and "Receiving Facility Field in the Message Header", earlier in this chapter.

11.13.3 Guarding Against Processing Messages from Inappropriate Accounts

V*IST*A HL7 supports the HL7 standard's codes for Processing ID. You can use the Processing ID field in message headers to guard against processing messages from inappropriate accounts. The **V*IST*A** HL7 site parameter field Default Processing ID ("Is this a Production or Test Account?") can be set to Production or Training. This value is placed in the Processing ID message header field for all outgoing messages.

The event driver protocol for a given transaction can override the site parameter setting, but it can only override with a value of DEBUG.

When processing an inbound message, **V*IST*A** HL7 checks the processing ID, if found in the incoming message's header, against that account's site parameter setting (if the message header is

Training or Production) or against the event driver protocol (if the value is Debug). If there is a mismatch, an error will be logged and the message will *not* be processed. Used properly, this feature can prevent messages from test accounts from being inadvertently processed in production accounts.

11.13.4 "Security Required?" Field in Subscriber Protocols

On sending systems, the security field in outgoing message headers is populated from the HLP("SECURITY") input variable used in various **V*IST*A** HL7 entry points for sending messages.

On receiving systems, if the "Security Required?" field in a subscriber protocol is set to YES, this means that the security field is required in the message header segment on incoming messages. If set to YES, and an incoming message does not have that information (or if the access code passed is invalid), the message is rejected.

In **V*IST*A** interfaces, the message header Security field is divided into three components, which must be separated by the HL7 component separator:

Component	Purpose
1	(required) Valid access code of a user in the New Person file (#200) on the **V*IST*A** system that receives the message. When a message is received, the user's DUZ corresponding to a valid access code is available in HL("DUZ") during message processing.
2	Reserved for future use.
3	(optional) Valid electronic signature code in the New Person file (#200) on the **V*IST*A** system that receives the message. This code must be for the user identified by component 1 of the Security field. When a message is received, the signature block printed name for valid users is available in HL("ESIG") during message processing.

Note: Encryption of values in the Security field is not currently supported.

When processing received messages, information from the security field can be used to perform specific actions on the server. For example, Kurzweil interfaces have passed an access code and electronic signature so that doctors can dictate and authenticate exam data directly from Kurzweil systems.

11.14 Testing Your Interface

11.14.1 Preventing Test Messages from Being Processed in Production Accounts

Before testing each transaction in your interface, make sure your test account's HL7 site parameter for "Is this a Production or Test Account" is set to the appropriate value; e.g., if this is a test account, set it to Test. The Processing ID field in message headers generated by your test interface will then contain TEST, and will not be processed if it is inadvertently sent to a production account.

11.14.2 LLP-Level Troubleshooting

TCP Interfaces

If you are sufficiently versed in the internal mechanisms of the MLLP protocol, you can establish a telnet session to the host IP address and port of the HL7 service on your target system. This lets you directly test the TCP communications link to the target system, by entering in message lines in conformance with the MLLP protocol.

Other approaches to troubleshooting TCP/IP interfaces include using network sniffers, and, on OpenVMS systems, using the TCP/IP trace utility. Using sniffers or tracing works well in conjunction with using the retransmit entry point ($$MSGACT^HLUTIL) to repeatedly send the same message over the interface.

HLLP Interfaces

If you are sufficiently versed in the internal mechanisms of the HLLP protocol, you can interactively log on to your target system and directly test the serial communications link to the target system by entering in message lines in conformance with the HLLP protocol.

On an OpenVMS system, to connect directly to another system over a serial line (VMS device LTA9904), execute the following DCL command:

```
SET HOST /DTE /ESCAPE=[ LTA9904
```

To break out of the session, enter the selected escape character.

11.15 Exporting and Deploying Your Interface

Using the Kernel Installation and Distribution System (KIDS), you can distribute the following (file-based) components of an interface:

- Applications: HL7 Application Parameter (File #771)
- Protocols: Protocol (File #101)
- Links: HL Logical Link (File #870)

Together, these three elements comprise the scope of an exportable HL7 application. Manual adjustment to the configuration of an interface after installation is usually needed, however.

11.15.1 HL7 Applications

To stop newly installed interfaces from transmitting messages (until the site wants to), it is recommended that you set your application's ACTIVE/INACTIVE flag to "Inactive" for export. It can be changed to active during your post-init, or by the site itself subsequent to the post-init. This prevents the application from becoming immediately active, without either your installation's intervention or the site's intervention.

11.15.2 HL7 Protocols

Relationships between Subscriber and Event Driver protocols are preserved by KIDS, if you explicitly send both the subscriber and event driver protocols, and depending on the KIDS installation action you choose for each protocol:

- Send to Site
- Delete at Site
- Use as Link for Menu Items
- Merge Menu Items.

For more information on KIDS, please see the *Kernel Systems Manual*.

Also, you should export event driver protocols with Processing ID set to null. This leaves control over what value is placed in outgoing message headers' Processing ID field in the hands of installing sites' HL7 site parameters.

11.15.3 Links

If you export links, you should:

- Export outgoing links with settings such as IP Address, Port, etc. null unless they are fixed in place regardless of the installation site.

Note: KIDS does *not* export Device pointers, Autostart and Startup Node fields, Time Started or Stopped, Task Number, Shutdown LLP?, or the "In" and "Out" queues and their related fields.

11.15.4 Exporting Interfaces after Patch HL*1.6*57

Patch HL*1.6*57 introduces file structure changes for protocols and links. Because these are both items that the Kernel Installation and Distribution System (KIDS) has special code to handle installations for, KIDS itself requires a patch update in order to install interfaces developed using post-HL*1.6*57 systems.

Kernel patch XU*8*131 updates KIDS to support HL*1.6*57 interfaces. File structure changes:

- Links are consolidated from two files (File #870 [HL Logical Link] and File #869.2 [HL Lower Level Protocol Parameter]) into a single file (File #870).
- Changes in the use of certain fields in the Protocol file allow reuse of the same set of protocols for both sending and receiving, for any given transaction.
- Protocol subscriptions move from File #101's Items multiple to a new Subscribers multiple.

Interfaces developed on systems at a patch level lower than HL*1.6*57 *can* be installed on systems that have been patched at level HL*1.6*57 and above. An automatic conversion is performed to upgrade components in those interfaces to the new format introduced by HL*1.6*57.

11.15.5 Exporting Only One Side of an Interface

The situation may come up where you have a new transaction to add to **V*IST*A**, i.e., a message you are generating for a new event point, but you don't know of any interested subscribers to that event point yet.

In this situation, you *can* send out just one side of the interface. Send out the application with the application disabled. When the site has a subscriber, at that point they should enable the sending application. Otherwise, each time the event point was reached, an HL7 message would be generated, but **V*IST*A** HL7 would log it as a "no subscribers" error, which would fill up the HL7 message files with error status messages. Your code should check the return value from INIT^HLFNC2 prior to sending a message; it will report failure if the application is disabled.

11.15.6 V*IST*A HL7 Applications Installed by Nationally Released Packages

When a **V*IST*A** package makes use of **V*IST*A** HL7 messaging functionality, it exports its **V*IST*A** HL7 application, protocols and links as part of its KIDS build.

National packages may or may not include their own links, or simply use existing links. TCP links come pre-defined in patch HL*1.6*39. Other link type entries (i.e., MailMan) are usually set up, but require some site settings before being made operational. Packages may instruct you to set up and enable a certain MailMan domain.

CHAPTER

12. API Reference

Category	Entry Point	Description
Message Creation and Transmission	INIT^HLFNC2	Initialize variables before building HL7 message
	GENERATE^HLMA	Sends a message (asynchronously)
	GENACK^HLMA1	Sends acknowledgment message (asynchronously)
	DIRECT^HLMA	Sends a message *synchronously* to a single subscriber over a TCP link
	$$MSGACT^HLUTIL	Cancels or requeues message transmission
	$$MSGSTAT^HLUTIL	Returns status of incoming or outgoing message
Links	LINK^HLUTIL3	Return the link for an Institution, VISN or domain
	CHKLL^HLUTIL	Verify if there is an operational outgoing TCP link for a given institution
Exception Processing	$$GETAPP^HLCS2	Returns the mail group for an HL7 application.
	$$DONTPURG^HLUTIL	Sets a message's "Don't Purge" flag
	$$TOPURG^HLUTIL	Clears a message's "Don't Purge" flag
	$$SETPURG^HLUTIL	Sets or clears a message's "Don't Purge" flag
	$$REPROC^HLUTIL	Reprocesses a message
Subscription Registry	$$ACT^HLSUB	Create a subscription registry entry
	GET^HLSUB	Get a subscription record
	UPD^HLSUB	Update a subscription record
Batch Message Creation	CREATE^HLTF	Batch messages: Creates a message stub and reserves a batch message id
	MSH^HLFNC2	Batch messages: builds MSH segments
Conversion	$$FMDATE^HLFNC	Convert HL7 date to DHCP
	$$FMNAME^HLFNC	Convert HL7 name to DHCP
	$$HLADDR^HLFNC	Convert DHCP address to HL7
	$$HLDATE^HLFNC	Convert DHCP date to HL7
	$$HLNAME^HLFNC	Convert DHCP name to HL7
	$$HLPHONE^HLFNC	Convert DHCP phone to HL7
	$$M10^HLFNC	HL7-compliant M10 checksum algorithm
	$$M11^HLFNC	HL7-compliant M11 checksum algorithm
	$$OLDM10^HLFNC	Non-HL7-compliant M10 checksum algorithm
	$$OLDM11^HLFNC	Non-HL7-compliant M11 checksum algorithm
	$$UPPER^HLFNC	Convert string to uppercase

12.1 Message Creation and Transmission

12.1.1 INIT^HLFNC2

Sets up HL7 environment variables needed when you build HL7 messages.

Usage	`INIT^HLFNC2(EID, .HL, [internal])`	
Input	EID	(required) The IEN of the event driver protocol from the Protocol file (#101) that will be used to send the message.
	HL	(required) *Pass by reference.* The array in which to return output parameters (typically you should kill this beforehand.)
	internal	(optional) To return only the values needed for an internal, same-system interface, pass 1 for this parameter.
Output	HL	If the call succeeds, the top-level node of HL will be undefined. If the call fails, $G(HL) will be true, and HL will be set to the error message/reason for failure. Check this variable and only proceed with message generation if it succeeds.
	HL("ACAT")	The accept acknowledgment type from the specified event driver protocol, if defined.
	HL("APAT")	The application acknowledgment condition from the specified event driver protocol, if defined.
	HL("CC")	The country code of the sending application, if defined in the application's entry in the HL7 Application Parameter file.
	HL("ECH")	4-character string of HL7 encoding characters to use (e.g., "~|\&"). The four encoding characters are the component separator, repetition separator, escape character, and sub-component separator, in that order. If not specified in the protocol's application, **V*IST*A** HL7 defaults to ~|\&.
	HL("ETN")	3-character event type name from the specified event driver protocol (e.g., A01, O01, etc.).
	HL("FS")	The HL7 field separator character to be used for building HL7 segments. The field separator is a single character (e.g., ^).
	HL("MTN")	3-character message type name from the specified event driver protocol (e.g., ADT, QRY, ORU, etc.).
	HL("PID")	Processing ID (P for production, T for Training, D for Debug).
	HL("Q")	Two quotation marks (""). Use to insert null field values in segments.
	HL("SAF")	The name of the sending facility.
	HL("SAN")	The name of the sending application (e.g., Radiology).
	HL("VER")	Version number of the HL7 protocol to use to build the message being sent.

Example

```
> K HL D INIT^HLFNC2("RH ADT SERVER",.HL)
> ZW HL

HL("ACAT")=AL
HL("APAT")=AL
HL("CC")=US
HL("ECH")=~|\&
HL("ETN")=A01
HL("FS")=^
HL("MTN")=ADT
HL("PID")=P
HL("Q")=""
HL("SAF")=RCP
HL("SAN")=EDR-MAS
HL("VER")=2.2
```

12.1.2 GENERATE^HLMA

Delivers a message to all subscribers. The message is delivered asynchronously; this is a non-blocking call.

Usage	`GENERATE^HLMA(hleid, hlarytyp, 1, .result, [hlmtien], [.options])`	
Input	hleid	(required) The name or IEN of the event driver protocol for which the message is being generated.
	hlarytyp	(required) Array type. One of the following codes: LM local array containing a single message LB local array containing a batch of messages GM global array containing a single message GB global array containing a batch of messages
	1	(required) Specifies whether the HLA array is pre-formatted in HL7 format. At this time, always set it to 1.
	result	(required) *Pass by reference*. The message ID assigned to this message and/or an error message will be returned in this parameter.
	hlmtien	(optional) For batch messages only. The IEN of the stub message in File #772 created by the prior call to the entry point CREATE^HLTF.
	options	(optional) *Pass by reference*. The Options array and each of the following nodes within it is optional: • Options("CONTPTR"): Set to a value to go in the continuation pointer field of the Message Header segment for the message being sent. • Options("SECURITY"): Set to any security information to include in the Security field (#8) of the HL7 MSH or BHS (batch header) segment.
Output	Result	If the call is successful, and the message was delivered to a single recipient, the first piece of the Result parameter is returned equal to the message id assigned to the message that was created. If the message was delivered to additional recipients, the second message ID is returned as Result(1), the third as Result(2), and so forth. If the call was *not* successful, Result is returned with the following three pieces of data: **Piece** **Contains** 1 message id (or 0 if no message id was assigned) 2 error code 3 error message If +$P(RESULT,U,2), there was an error.

Call INIT^HLFNC2 before making this call, to set up HL7 environment variables needed to build your message and needed by GENERATE^HLMA.

GENERATE^HLMA expects the segments for the message to be already loaded in the HLA("HLS") local array or the ^TMP("HLS") global array. Set a node for each segment for the message, subscripted numerically, starting at subscript 1. To accommodate segments greater than 245 characters, each segment node can define sub-nodes to hold the rest of the segment that exceeds the 245-character limit.

For example (global array):

```
S ^TMP("HLS",$J,1)="EVN"_HL("FS")_"A01"
S ^TMP("HLS",$J,2)="PID"_HL("FS")_... start of very long segment
S ^TMP("HLS",$J,2,1)= continuation of segment
S ^TMP("HLS",$J,2,2)= continuation of segment
```

For example (local array):

```
S HLA("HLS",1)="EVN"_HL("FS")_"A01"
S HLA("HLS",2)="PID"_HL("FS")_... start of very long segment
S HLA("HLS",2,1)= continuation of segment
S HLA("HLS",2,2)= continuation of segment
```

Internally, **V*IST*A** HL7 stores the message is such a way that no ambiguity exits for segments greater than 245.

Example

```
D GENERATE^HLMA("EVENT DRIVER PROTOCOL","GB",1,.MYRESULT)
I +$P(MYRESULT,U,2) D ERR
```

12.1.3 GENACK^HLMA1

Sends an acknowledgment message when receiving and processing an incoming message. Only call this entry point from a routine invoked by an HL7 subscriber protocol's PROCESSING ROUTINE.

Usage	`GENACK^HLMA1(EID, HLMTIENS, EIDS, ArrayType, 1, .Result, [MTIEN], [.Options])`	
Input	EID	(required) The IEN of the sending (event driver) protocol in the Protocol file (#101). Get this value from HL("EID").
	HLMTIENS	(required) The IEN of the message for the receiving (subscriber) application. This can be pulled from current environment; it is defined when processing a message.
	EIDS	(required) IEN of receiving (subscriber) protocol from the Protocol file. Get this value from HL("EIDS").
	ArrayType	(required) Array type. One of the following codes: LM local array containing a single message LB local array containing a batch of messages GM global array containing a single message GB global array containing a batch of messages
	1	(required) This parameter specifies whether the HLA array is pre-formatted in HL7 format. At this time, always set it to 1.
	Result	(required) *Pass by reference*. Potential holder of returned value.
	MTIEN	(optional) For batch acknowledgments only. The IEN of the entry in the HL7 Message Text file (#772) created by the call to the entry point CREATE^HLTF.
	Options	(optional) *Pass by reference*. The Options parameter and each of the following array nodes within it are optional: • Options("CONTPTR"): Set to a value to go in the continuation pointer field of the Message Header segment for the message being sent. • Options("SECURITY"): Set to any security information to include in the Security field (#8) of the HL7 MSH segment.
Output	Result	Several formats may be returned in this variable/parameter. Any string that contains a value in a second caret piece indicates that the generation of an acknowledgement message was NOT successful. Successful calls are indicated by the possible return values: -NULL -MESSAGE ID Values indicating no acknowledgement was generated include: -MESSAGE ID^ERROR (IN SEVERAL DIFFERENT FORMATS) -0^IEN 771.7^ACTUAL ERROR MESSAGE

-IEN 771.7^ACUTUAL ERROR MESSAGE
-^ACTUAL ERROR MESSAGE

Because of the format variations possible in the return string, it is recommended that calling applications not try to interpret or rely on the specific values returned in the string and only use the existence or absence of a second piece to determine whether the call was successful.

Note: Because GENACK changes variables, we recommend calling it as the last variable-dependent action in a processing routine.

GENACK^HLMA1 expects the segments for the acknowledgment message to be already loaded in the HLA("HLA") local array or the ^TMP("HLA") global array. Set a node for each segment for the message, subscripted numerically, starting at subscript 1. To accommodate segments greater than 245 characters, each segment node can define sub-nodes to hold the rest of the segment that exceeds the 245-character limit.

For example (global array):

```
S ^TMP("HLA",$J,1)="MSA"_HL("FS")_"AA"_HL("FS")_HL("MID")
```

For example (local array):

```
S HLA("HLA",1)=)="MSA"_HL("FS")_"AA"_HL("FS")_HL("MID")
```

The format for segments longer than 245 characters is explained in the description for GENERATE^HLMA.

Example

```
D GENACK^HLMA1(HL("EID"),HLMTIENS,HL("EIDS"),"LM",1,.MYRESULT)
I +$P(MYRESULT,U,2) D ERR
```

12.1.4 DIRECT^HLMA

Delivers a message:

- To a single subscriber only;
- Over an outgoing TCP link only;
- Immediately. Message is delivered synchronously; this is a *blocking* call.

Use this entry point when you need an immediate response, for example, when making a real-time query to another system. It transmits the message immediately, and waits for a response before returning to the calling routine.

Usage	`DIRECT^HLMA(hleid, hlarytyp, 1, .result, [hlmtien], [.options])`	
Input	hleid	(required) The name or IEN of the event driver protocol for which the message is being generated. For the asynchronous call, it must be a protocol that uses a TCP link, and should have only one subscriber.
	hlarytyp	(required) Array type. One of the following codes: LM local array containing a single message LB local array containing a batch of messages GM global array containing a single message GB global array containing a batch of messages
	1	(required) Specifies whether the HLA array is pre-formatted in HL7 format. At this time, always set it to 1.
	result	(required) *Pass by reference*. Results are returned in this parameter (see Output below.)
	hlmtien	(optional) For batch messages only. The IEN of the message stub entry in File #772 created by the prior call to CREATE^HLTF.
	options	(optional) *Pass by reference*. The Options array and each of the following nodes within it is optional: • Options("CONTPTR"): Set to a value to go in the continuation pointer field of the Message Header segment for the message being sent. • Options("SECURITY"): Set to any security information to include in the Security field (#8) of the HL7 MSH or BHS (batch header) segment.
Output	Result	If the call is successful (message sent), the Result parameter is returned with piece 1 equal to the message ID of the message created. If the call failed (message was not sent), the Result parameter is returned with the following three pieces of data: message ID(0 if none assigned)^error code^error description If the call failed, +$P(RESULT,U,2) returns True.
	HLMTIEN	If HLMTIEN is positive, a response was returned, and it is ready for processing. Otherwise, no response was returned.

Call INIT^HLFNC2 before making this call, to set up HL7 environment variables needed to build your message and needed by DIRECT^HLMA.

This call has the same input parameters as GENERATE^HLMA. Like GENERATE^HLMA, it expects segments for the message to be already loaded in the HLA("HLS") local array or the ^TMP("HLS") global array. For more information on the expected format of HLA("HLS") or ^TMP("HLS"), please see the GENERATE^HLMA call.

To select the subscriber to transmit to, this call first checks the contents of the HLL("LINKS") array; it will use first record it finds. If that array is empty, it uses the first subscriber protocol it finds in the Subscribers multiple of the event driver protocol.

Upon return, DIRECT^HLMA does not invoke the event driver protocol's GENERATE/PROCESS ACK ROUTINE to processes the acknowledgment. When control is returned to the calling routine, the environment is left in the same condition as if you were in the "processing routine environment" when receiving an acknowledgment; the variables HLNEXT, HLNODE, and HLQUIT are defined so that you can process the response.

Note that DIRECT^HLMA supports:

- All acknowledgment modes (including deferred acknowledgments)
- Batch message transmission

The timeout for the synchronous transmission is determined by the settings of the TCP used.

Example

```
D DIRECT^HLMA("EVENT DRIVER PROTOCOL","GB",1,.MYRESULT)
I +$P(MYRESULT,U,2) D ERR Q  ; message was not transmitted
I HLMTIEN D PROCESS ; response was returned from target system
```

12.1.5 $$MSGACT^HLUTIL (TCP Only)

Cancels or requeues an outgoing message, over TCP links only:

- Canceling a message cancels its transmission.
- Requeuing a message clears its error status, if any, and requeues it for transmission. If the message has not been sent yet, it is requeued which still leaves it ready for transmission. If it has already been sent, it is requeued for retransmission over the same link.

Usage	`$$MSGACT^HLUTIL(messageID, action)`	
Input	messageID	(required) Message ID assigned by **V*IST*A** HL7 to the outgoing message.
Output	action	(required) Desired action: 1 to cancel, or 2 to requeue.
	return value	Returns 1 if the action succeeded; 0 if the action failed. Canceling or requeuing a message will fail if the message is locked (typically, if the message is in the act of transmission). Requeuing a message can also fail if the message is either not an outgoing message, or does not have a defined link.

Example

```
S X=$$MSGACT^HLUTIL(A6AMSGID,1)
```

12.1.6 $$MSGSTAT^HLUTIL (TCP Only)

Returns the status of a specified incoming or outgoing message, over TCP links only.

Usage	`$$MSGSTAT^HLUTIL(messageID)`	
Input	messageID	(required) Message ID (from the message's header).
Output	return value	Five-piece string, in the following format: status^status updated^error msg^error type^retransmission "Status" is one of the following: 0: message doesn't exist 1: pending transmission 1.5: being transmitted 2: awaiting application acknowledgment 3: successfully completed 4: error 8: being generated 9: awaiting processing "Status updated" is the date and time the Status was updated. "Error message", if there was an error transmitting the message, contains a text error description. "Error type", if there was an error transmitting the message, contains a pointer to the relevant error in the HL7 Error Message File (#771.17). "Retransmissions" contains the number of retransmission attempts, if any, for an outgoing message.

Note: Message ID should be unique for both incoming and outgoing messages, if interfaces are negotiated correctly. For **V*IST*A**'s side of an interface, the message ID is composed of the current domain's institution number concatenated with a unique number. For the other side of the interface, a similar convention should be used, with a unique prefix representing the other system.

Example

```
S X=$$MSGSTAT^HLUTIL(A6AMSGID)
```

12.2 Links

12.2.1 LINK^HLUTIL3

Returns link(s) associated with the Institution, VISN or domain you specify.

Usage	`LINK^HLUTIL3(inst, .out_array, [flag])`	
Input	inst	(required) Institution, VISN or Domain (pass either Name or IEN).
	out_array	(required) Output array. Pass by reference.
	flag	(optional) Flag indicating reference type passed in "inst" parameter. Can be one of the following: D: "inst" is Domain, either as an IEN, or as any value that a FileMan multiple-index lookup can match to a single Domain file entry, including Name I: "inst" is any value that a FileMan multiple-index lookup can match to a single Institution file entry, including Name, Official Name or Station Number as defined in the Institution file null: "inst" is IEN of a record in Institution file
Output	out_array	If one or more links are associated with the institution, VISN or domain, the IEN and name of each link are returned in the output array. The format of each entry returned in the output array is: `ARRAY(LINK_IEN)=LINK NAME`

Institutions and VISNs are entries in the Institution File (#4). Domains are entries in the Domain file (#4.2). The association between a link and an institution or domain is created by setting the link's Institution and Domain fields.

One example of using this entry point is to set up the HLL("LINKS") array to dynamically address a message.

Note: The ability to group other institutions within a VISN institution entry was introduced by Kernel patch XU*8*43.

Examples

```
>;by station number
>K ZZLL D LINK^HLUTIL3("516",.ZZLL,"I") ZW ZZLL
ZZLL=
ZZLL(14)=VA516

>; by institution name
>K ZZLL D LINK^HLUTIL3("BAY PINES",.ZZLL,"I") ZW ZZLL
ZZLL=
ZZLL(14)=VA516
```

```
>; by institution IEN
>K ZZLL D LINK^HLUTIL3(516,.ZZLL) ZW ZZLL
ZZLL=
ZZLL(14)=VA516
```

12.2.2 $$CHKLL^HLUTIL

Verify if there is an operational outgoing TCP link for a given institution. The link must be a) an outgoing (client) TCP link, and b) operational (not shut down).

Usage	`$$CHKLL^HLUTIL(institution)`	
Input	institution	(required) Institution Name or Number associated with a link.
Output	return value	Returns one of the following values: 1: If the specified link is an outgoing (client) TCP link, and the link is not shut down. 0: If the link does not satisfy the above conditions.

This entry point is useful when you are dynamically routing messages, and therefore choosing links on the fly to deliver those messages. If the function returns true, the link should be suitable for use to transmit messages to another VA facility.

Example

```
> W $$CHKLL^HLTUIL("TAMPA")
1
```

12.3 Exception Processing

12.3.1 $$GETAPP^HLCS2

Returns the name of a mail group associated with a particular HL7 application (from the application's Mail Group field).

Usage	`$$GETAPP^HLCS2`	
Input	application	(required) Name or IEN of an application in the HL7 Application Parameter file (#771).
Output	return value	If the application is found, a two-piece value is returned: `mailgroup^status` Mailgroup: Name of the mail group associated with the application, or null if none. Status: "a" if the application is active, "i" if the application is inactive, or null if neither. If the application is not found, null is returned.

Example

```
>W $$GETAPP^HLCS2("JC TEST SERVER")

 JOHN^a
```

12.3.2 $$DONTPURG^HLUTIL

Use this entry point to set a message's DON'T PURGE flag (#773,10). This prevents the message from being purged by the HL PURGE TRANSMISSIONS option. As of patch HL*1.6*19, use only for messages sent or received over TCP links.

Usage	`$$DONTPURG^HLUTIL()`	
Input	HLMTIENS	(required) Must be defined in the environment, not passed as a function parameter. Set to the IEN of the message in File #773. This variable should already be defined if in a message processing context.
Output	return value	1: The message's "Don't Purge" flag has been set to "Don't Purge". -1: The call has failed; nothing has been changed.

Example

```
>S HLMTIENS=21 I '$$DONTPURG^HLUTIL() W "Could not set DON'T PURGE flag."
```

12.3.3 $$TOPURG^HLUTIL

Use this entry point to clear a message's DON'T PURGE flag (#773,10), allowing the message to be purged by the HL PURGE TRANSMISSIONS option. As of patch HL*1.6*19, use only for messages sent over TCP links.

Usage	`$$TOPURG^HLUTIL()`	
Input	HLMTIENS	(required) Must be defined in the environment, not passed as a function parameter. Set to the IEN of the message in File #773. This variable should already be defined if in a message processing context.
Output	return value	0: The message's "Don't Purge" flag has been cleared. -1: The call has failed; nothing has been changed.

Example

```
>S HLMTIENS=21 I $$TOPURG^HLUTIL()<0 W "Could not clear DON'T PURGE flag."
```

12.3.4 $$REPROC^HLUTIL

Use this entry point to *reprocess* a message.

Usage	$$REPROC^HLUTIL(IEN, routine)	
Input	IEN	(required) IEN of the message in File #773 to reprocess.
	routine	(required) The routine that should be called to reprocess the message.
Output	return value	0: The message was reprocessed. -1: The message was not reprocessed.

Reprocessing is essentially the same as when, in an interface, the **V*IST*A** HL7 calls an applications PROCESS ROUTINE (for incoming messages) or PROCESS ACKNOWLEDGMENT routine (for incoming acknowledgments). As of patch HL*1.6*19, use only for messages sent over TCP links (you must know this in advance).

Example

```
>W $$REPROC^HLUTIL(34,"A6ARADPR")
0
```

12.3.5 $$SETPURG^HLUTIL

$$SETPURG^HLUTIL is an alternate entry point that duplicates the functionality of both $$TOPURG^HLUTIL and $$DONTPURG^HLUTIL.

Usage	`$$SETPURG^HLUTIL(status)`	
Input	status	(required) 1: Sets the message's "Don't Purge" flag such that the message won't be purged (equivalent to $$DONTPURG^HLUTIL). 0: Sets the message's "Don't Purge" flag such that the message will be purged (equivalent to $$TOPURG^HLUTIL).
	HLMTIENS	(required) Must be defined in the environment, not passed as a function parameter. Set to the IEN of the message in File #773. This variable should already be defined if in a message processing context.
Output	return value	1: The message's "Don't Purge" flag has been set to "Don't Purge". 0: The message's "Don't Purge" flag has cleared. -1: The call has failed; nothing has been changed.

12.4 Subscription Registry

12.4.1 GET^HLSUB

Returns the list of active subscriber subrecords for a particular subscription registry record. **V***IST***A** HL7 can use this array as the list of recipients for dynamic addressing.

Usage	`GET^HLSUB(subnum, [subtype], [subs_p], .out_array)`	
Input	subnum:	(required) Subscription control number.
	subtype:	(optional) Subscription type. Use this to return subscribers based on their TYPE field. If null, all subscribers for this subscription registry entry are returned. If 0, types 0 and 1 are returned. If 1, only type 1 is returned. If 2, only type 2 is returned.
	subs_p:	(optional) Valid HL7 subscriber protocol name for message routing. GET^HLSUB uses this as piece 1 of each record it adds to the output array. If your are making this call specifically to dynamically route a message, pass this parameter so that each record added to the output array is added in the format required for message routing -- the format of the HLL("LINKS") array. If null, piece 1 of each record in the output array will also be null.
	out_array:	(required) Used by GET^HLSUB for output; pass it by reference. GET^HLSUB adds any matching subscribers to this array as output. If it already exists, it will be appended to. Typically, you would pass the HLL array, because HLL("LINKS") can then be used directly for routing by **V***IST***A** HL7.
Output	ARRAY("LINKS",n)	For each qualifying destination record found in the subscription registry entry, one array record is added to the output array (passed by reference as an input parameter): ARRAY("LINKS",n) = Subscription Record Each subscription record is in the format:

Piece	Description
1	Subscriber Protocol name (if passed)
2	Link name
3	Destination (rec.app@link)
4	Receiving Application (IEN)
5	(reserved for future use)
6	Subscription Link (IEN)
7	Subscription Type
8	Subscription Creation Date/Time
9	Subscription Activation Date/Time
10	Subscription Termination Date/Time

You can make multiple calls to GET^HLSUB. For example, to route a message to both Clinical and Descriptive subscribers of a particular subscription registry entry, first call GET^HLSUB with sub_type set to '0', and then call it with sub_type set to '1.' If the output array already exists, it will be appended to.

Examples

1. Return all subscriber information for Subscription Control record #3.

```
>K HLL D GET^HLSUB(3,,,.HLL) ZW HLL

HLL("LINKS",1)=^REDACTED^RADIOLOGY@REDACTED^29^^24^1^2991204.101744^2991204.
101744
```

2. Return all subscribers of type 1 for Subscription Control record #3, specifying the subscriber protocol for message routing:

```
>K HLL D GET^HLSUB(3,1,"RH ADT SUBS SEND",.HLL) ZW HLL

HLL("LINKS",1)=RH ADT SUBS SEND^REDACTED^RADIOLOGY@REDACTED^29^^24^1^
  2991204.101744^2991204.101744
```

12.4.2 $$ACT^HLSUB

Use this entry point to add a new top-level record to the subscription registry. The only field populated in the new record is NUMBER (subscription control number). To add subscribers for this record, use the UPD^HLSUB call. Your application is responsible for tracking each subscription registry record it adds.

Usage	$$ACT^HLSUB	
Input	none	
Output	return value	A unique number that can be used to reference this registry entry in the future.

Example

```
>S X=$$ACT^HLSUB W X
2992
```

12.4.3 UPD^HLSUB

Creates or edits an existing subscription subrecord in a subscription registry entry. Subrecords are keyed on the link name: If an existing subscription subrecord matches the specified link name, that subrecord is updated; otherwise, a new subrecord is added.

Usage	`UPD^HLSUB(sub_num, llname, [sub_type], [act_date], [term_date], [rec_app],[.err_array])`	
Input	sub_num	(required) Subscription Control Number for which to create or edit a subscription subrecord.
	llname	(required) Link name in File #870 for the subscription subrecord.
	sub_type	(optional) Subscription Type (arbitrary; types 0 and 1 mainly for use by CIRN). If null, defaults to type 0. Current types supported are: 0=patient descriptive only 1=patient clinical and descriptive 2=other (locally defined)
	act_date	(optional) Activation date/time. Time is required in addition to date. Defaults to 'now'. Subscribers are considered inactive before this date. GET^HLSUB won't retrieve inactive records.
	term_date	(optional) Termination date/time. Defaults to open-ended. Time is optional; if not included, will be appended to the date as ".0001". Passing null deletes the current termination date. Subscribers are considered inactive after this date; GET^HLSUB won't retrieve inactive records.
	rec_app	(optional) Name (.01 field) or IEN of the receiving application from the HL7 Application Parameter file; this names the subscribing application.
	err_array	(optional) Used to return error messages resulting from this call. Pass by reference.

Output	err_array	Error messages are returned in this variable (passed by reference as an input variable). Possible errors are:
		`HLER(1)="Missing subscription control number"`
		`HLER(2)="Missing network address."`
		`HLER(4)="Invalid Subscription Control number"`
		`HLER(5)="Invalid Logical Link"`

Example

```
D UPD^HLSUB($$ACT^HLSUB,"REDACTED",1,,,"RADIOLOGY",.A6AERR)
```

resulting subscription subrecord:

```
NUMBER: 2993
DESTINATION: RADIOLOGY@REDACTED            RECEIVING APPLICATION: RADIOLOGY
  LOGICAL LINK: REDACTED                   TYPE: Patient Clinical and
Descriptive
  CREATION DATE/TIME: DEC 04, 1999@10:17:44
  ACTIVATION DATE/TIME: DEC 04, 1999@10:17:44
```

12.5 Batch Message Creation

12.5.1 CREATE^HLTF

Use this call to create a message stub for the batch message and reserve a batch message id. If you are creating a *batch* of HL7 messages (more than one), your application processing routine should:

1. Call INIT^HLFNC2 to initialize variables.
2. Call CREATE^HLTF to create a message stub for the batch message.
3. Create and store each message that will be in the batch (requires calls to INIT^HLFNC2 and MSH^HLFNC2).
4. Transmit the completed batch message using GENERATE^HLMA.

If you are sending only one HL7 message (i.e., you're not sending a batch message), don't call this entry point. For single messages, **V*IST*A** HL7 creates the MSH segment for you.

For more information on sending batch messages using **V*IST*A** HL7, please see the "Advanced Interface Issues" chapter.

Usage	`CREATE^HLTF(.HLMID, .MTIEN, .HLDT, .HLDT1)`	
Input	HLMID	(required) *Pass by reference*. The variable in which to return the batch message ID.
	MTIEN	(required) *Pass by reference*. The parameter in which to return the IEN of the message stub in the HL7 Message Text file (#772).
	HLDT	(required) *Pass by reference*. The parameter in which to return the message date/time in VA FileMan format.
	HLDT1	(required) *Pass by reference*. The parameter in which to return the message date/time in HL7 format.
Output	HLMID	Message ID of the "stub" message that has been created.
	MTIEN	IEN in the File #772 for the stub message that has been created.
	HLDT	Message creation date/time in VA FileMan format.
	HLDT1	Message creation date/Time in HL7 format.

Example

```
D CREATE^HLTF(.HLMID,.MTIEN,.HLDT,.HLDT1)
```

12.5.2 MSH^HLFNC2

Use this call to build MSH segments for each individual message that you're adding to a batch message. If you are creating a *batch* of HL7 messages (more than one), your application processing routine should:

1. Call INIT^HLFNC2 to initialize variables.
2. Call CREATE^HLTF to create a message stub for the batch message.
3. Create and store each message that will be in the batch (requires calls to INIT^HLFNC2 and MSH^HLFNC2).
4. Transmit the completed batch message using GENERATE^HLMA.

If you are sending only one HL7 message (i.e., you're not sending a batch message), don't call this entry point. For single messages, **V*IST*A** HL7 creates the MSH segment for you.

For more information on batch messages, please see the "Advanced Interface Issues" chapter.

Usage	`MSH^HLFNC2(.HL, MID, .result, [security])`	
Input	HL	(required) *Pass by reference*. The array of values returned by the call to INIT^HLFNC2.
	MID	(required) The message ID to be included in the MSH segment. It should be created for each segment by concatenating together the following: 1. Batch message ID returned by the call to CREATE^HLTF 2. Hyphen 3. Sequential batch message counter, a whole number whose sequence starts at 1 and increments by 1 For example, "12345-1" would be the first message in a batch with a message ID of 12345.
	result	(required) *Pass by reference*. Variable in which to return the message header.
	security	(optional) Security information (1-40 characters) to be included in field #8 of the MSH segment.
Output	result	Based on the input parameters, an HL7 Message Header (MSH) segment is returned in the input parameter. If the MSH segment is longer than 245 characters, the continuation is returned in subscript 1 of this parameter, e.g., Result(1). If the required input parameters HL or MID are missing, Result is returned equal to null.

Example

```
S COUNT=COUNT+1 D MSH^HLFNC2(.HL, HLMID_"-"_COUNT,.MYRESULT,SECURITY)
```

12.6 Conversion Utilities

12.6.1 HL7/VA FileMan Data Types Conversion Chart

The following table lists HL7 data types and their corresponding VA FileMan data types. Listed under the first column titled "Function Call" are entry points in the HLFNC routine that can be called to convert VA FileMan data to HL7 format. Listed under the second column titled "Function Call" are entry points that can be called to convert HL7 data to VA FileMan format.

HL7 Data Type	Function Call	VA FileMan Data Type	Conversion Function
(ST) String	*none needed*	Free Text	None Needed
(TX) Text	*none needed*	Word Processing	None Needed
(FT) Formatted Text (See note below)	*none*	*no equivalent*	None
(NM) Numeric	*none needed*	Numeric	None Needed
(DT) Date	$$HLDATE	Date Only	$$FMDATE
(TM) Time	$$HLDATE	Time Only	$$FMDATE
(TS) Time Stamp	$$HLDATE	Date/Time	$$FMDATE
(PN) Person Name	$$HLNAME	Free Text	$$FMNAME
(TN) Telephone No.	$$HLPHONE	Free Text	None Needed
(AD) Address	$$HLADDR	Free Text	None Needed
(ID) Coded Value	*none needed*	Set of Codes or Pointer	None Needed
(SI) Sequence ID	*none needed*	Numeric	None Needed
(CM) Composite	*none needed*	*no equivalent*	None Needed
(CE) Coded Element	*none needed*	*no equivalent*	None Needed
(CF) Coded Element with Formatted Value	*none needed*	*no equivalent*	None Needed
(CK) Composite ID with Check Digit	$$M10, $M11	*no equivalent*	None Needed
(CN) Composite ID and Name	*none needed*	*no equivalent*	None Needed
(CQ) Composite Quantity with Units	*none needed*	*no equivalent*	None Needed
(MO) Money	*none needed*	*no equivalent*	None Needed
(RP) Reference Pointer	*none needed*	*no equivalent*	None Needed
(TQ) Timing Quantity	*none needed*	*no equivalent*	None Needed

Note: The formatted text (FT) data type is not required in any HL7 fields at this time. The formatted text data type includes formatting instructions (escape codes) in the data string. It is recommended that locally created fields not be assigned the formatted data type. Use the string (ST) or text (TX) data types instead.

12.6.2 $$FMDATE^HLFNC

Provided for backwards compatibility. Use Kernel's HL7TFM^XLFDT entry point instead.

Converts a date, date/time, or time in HL7 format to a date, date/time, or time in internal VA FileMan format.

Usage	`$$FMDATE^HLFNC(hldate)`	
Input	hldate	(required) Date, date/time or time in HL7 format.
Output	return value	Date converted to VA FileMan format.

Example

```
>W $$FMDATE^HLFNC("19980506173216") <RET>
2980506.173216
```

Note: HL7's internal representation for midnight is 0000. VA FileMan's internal representation for midnight is 2400. Therefore, the end of one day (2400) coincides with (0000) at the start of the next day (e.g., 2400 on 12 April 1995 is the same as 0000 on 13 April 1995.). Both are valid representations for midnight and the function returns the correct time and date accordingly.

12.6.3 $$HLDATE^HLFNC

Provided for backwards compatibility. Use Kernel's FMTHL7^XLFDT entry point instead.

Converts a date, date/time, or time in internal VA FileMan format to a date, date/time, or time in HL7 format.

Usage	`$$HLDATE^HLFNC(fmdate, [format])`	
Input	fmdate	(required) VA FileMan date/time.
	format	(optional) The specificity of date/time that is returned (date only, date/time or time only) is, by default, the same specificity as is provided in the fmdate input parameter. You can force a particular return format by passing one of the following two-character strings for the format parameter: • DT: Date Only • TM: Time Only • TS: Timestamp (Date and Time)
Output	return value	Date, date/time or time in HL7 format.

Examples

```
>D NOW^%DTC W %,!,$$HLDATE^HLFNC(%) <RET>
2980506.173216
19980506173216
```

```
>D NOW^%DTC W %,!,$$HLDATE^HLFNC(%,"DT") <RET>
2980506.173259
19980506
```

Note: VA FileMan's internal representation for midnight is 2400. HL7's internal representation for midnight is 0000. Therefore, the end of one day (2400) coincides with (0000) at the start of the next day (e.g., 2400 on 12 April 1995 is the same as 0000 on 13 April 1995.). Both are valid representations for midnight and the function returns the correct time and date accordingly.

12.6.4 $$FMNAME^HLFNC

Converts a name in HL7 format to a name in DHCP format.

Usage	`$$FMNAME^HLFNC(hlname, [hlecde])`	
Input	hlname	(required) HL7-formatted name to convert to a name in DHCP format.
	hlecde	(optional) 4-character string of HL7 encoding characters to use (e.g., "~|\&"). If this parameter is not passed, the HLECH variable must be defined already with those characters. If passed, this parameter overrides HLECH.
Output	return value	Converted name in DHCP format.

Example

```
>W $$FMNAME^HLFNC("PAUL~ADRIAN~H.") <RET>
PAUL,ADRIAN H.

>W $$FMNAME^HLFNC("DOE~JOHN~""""~H.") <RET>
DOE,JOHN SR
```

12.6.5 $$HLNAME^HLFNC

Converts a name in the format typically used in **V*IST*A** (e.g., lastname,firstname) to a name in HL7 format. The variable X is the input variable equal to the DHCP name.

Usage	`$$HLNAME^HLFNC(name, [hlecde])`	
Input	name	(required) DHCP name.
	hlecde	(optional) 4-character string of HL7 encoding characters to use (e.g., "~|\&"). If this parameter is not passed, the HLECH variable must be defined already with those characters. If passed, this parameter overrides HLECH.
Output	return value	Name in HL7 format.

Example

```
>W $$HLNAME^HLFNC("PAUL,ADRIAN H.") <RET>
PAUL~ADRIAN~H.
```

12.6.6 $$HLADDR^HLFNC

Converts address information in DHCP format to an address in HL7 format.

Usage	`$$HLADDR^HLFNC(address, geo_loc, [hlecde])`	
Input	address	(required) The address can be one to four pieces representing lines of a street address. The pieces must be delimited with carets (e.g., VA Medical Center^123 Anywhere Street^Suite 101^Building 36).
	geo_loc	(required) The geo_loc parameter can be one to four pieces contain geographic location information in the following order: city, state or province, zip code, country code. The pieces must be delimited with carets (e.g., city^state or province^zip code^country code) and defined as follows: 1. The first component of the GL variable (city) will be truncated to no more than 30 characters. 2. The second component of the GL variable (state or province) must be the internal entry number of an entry in the State file (#5). 3. The third component of the GL variable (zip code) must pattern match 5N, 9N or 5N1"-"4N. 4. The fourth component of the GL variable (country code) will be set to "USA" if it is not defined.
	hlecde	(optional) 4-character string of HL7 encoding characters to use (e.g., "~|\&"). If this parameter is not passed, the HLECH variable must be defined already with those characters. If passed, this parameter overrides HLECH.
Output	return value	Address in HL7 format. The maximum length of the formatted string of six address components that is returned (including component separators) is 106 characters.

Example

```
>W $$HLADDR^HLFNC("301 Howard St.^Suite 600","San
Francisco^6^94105")
301 Howard St.~Suite 600~San Francisco~CA~94105~USA
```

12.6.7 $$HLPHONE^HLFNC

Converts a telephone number in various formats to a telephone number in HL7 format.

Usage	`$$HLPHONE^HLFNC(string, [b], [c])`	
Input	string	(required) Phone number in various formats. This means: • Seven-digit phone number at a minimum. • Optional 3-digit area code • Optional 2-digit country code • Optional formatting characters (e.g., dashes, spaces and/or parentheses)
	b	(optional) Beeper number.
	c	(optional) Comment.
Output	return value	Phone number in HL7 format. The maximum length of the formatted string is 40 characters.

Example

```
> W $$HLPHONE^HLFNC("01 501 333 3333","18003335555","Dr. HL7Doctor")
01 (501)333-3333B18003335555CDr. HL7Doctor
```

12.6.8 $$UPPER^HLFNC

Converts a string to uppercase.

Usage	`$$UPPER^HLFNC(string)`	
Input	string	(required) String to convert to uppercase.
Output	return value	String, converted to uppercase.

Example

```
> W $$UPPER^HLFNC("cat")
CAT
```

12.6.9 $$M10^HLFNC

Use for data types of CK (composite ID with check digit). Given an ID number, this function uses the HL7-compliant M10 algorithm to generate a check digit. For more information, please see the "Control/Query" chapter of the *Health Level Seven* standard.

Note: The algorithm used by this function was corrected to be HL7-compliant in patch HL*1.6*38. The previous, non-HL7-compliant version of this function is still provided, in $$OLDM10^HLFNC.

Note: Patch HL*1.6*42 adds the ability to return non-null values if the ID number provided is not completely numeric.

Usage	`$$M10^HLFNC(ck_id, [cmp_sep])`	
Input	ck_id	(required) First component (the ID *number*) for an HL7 field whose data type is composite id with check digit (CK). Must be numeric.
	cmp_sep	(optional) Component separator to use. Pass a 1-character string containing the component separator only, or pass the 4-character string containing the HL7 encoding characters as defined in HL("ECH") (i.e., the component separator is the first character). The only rule is that $E(cmp_sep) must evaluate to the component separator character to use. If this parameter is not passed, the HLECH variable must contain the HL7 encoding characters already, in the format defined by **V*IST*A** HL7. If passed, this parameter overrides HLECH.
Output	return value	If ck_id is numeric, the first 3 CK data type components are returned, separated by specified component separator. If ck_id contains any non-numeric characters, it is incompatible with the HL7 standard; ck_id is returned concatenated with 2 component separators. If no component separator is available, ck_id is returned unaltered.

Example

```
>W $$M10^HLFNC("9993","~^\&")          (valid input)
9993~7~M10

>W $$M10^HLFNC("9993","~")             (valid input)
9993~7~M10

>W $$M10^HLFNC("9993a","~^\&")         (incompatible input)
9993a~~
```

12.6.10$$M11^HLFNC

Given an ID number, this function generates a check digit using the HL7-compliant M11 algorithm. Use for data types of CK (composite ID with check digit). For more information, please see the "Control/Query" chapter of the *Health Level Seven* standard.

Note: The algorithm used by this function was corrected to be HL7-compliant in patch HL*1.6*38. The previous, non-HL7-compliant version of this function is still provided, in $$OLDM11^HLFNC.

Note: Patch HL*1.6*42 adds the ability to return non-null values if the ID number provided is not completely numeric.

Usage	`$$OLDM11^HLFNC(ck_id, [hlecde])`	
Input	ck_id	(required) First component (the ID *number*) for an HL7 field whose data type is composite id with check digit (CK). Must be numeric.
	hlecde	(optional) Component separator to use. Pass a 1-character string containing the component separator only, or pass the 4-character string containing the HL7 encoding characters as defined in HL("ECH") (i.e., the component separator is the first character). The only rule is that $E(cmp_sep) must evaluate to the component separator character to use. If this parameter is not passed, the HLECH variable must contain the HL7 encoding characters already, in the format defined by **V*IST*A** HL7. If passed, this parameter overrides HLECH.
Output	return value	If ck_id is numeric, the first 3 CK data type components are returned, separated by specified component separator. If ck_id contains any non-numeric characters, it is incompatible with the HL7 standard; ck_id is returned concatenated with 2 component separators. If no component separator is available, ck_id is returned unaltered.

Example

```
>W $$M11^HLFNC("9993","~^\&")          (valid input)
9993~7~M11

>W $$M11^HLFNC("9993","~")             (valid input)
9993~7~M11

>W $$M11^HLFNC("9993a","~^\&")         (incompatible input)
9993a~~
```

12.6.11$$OLDM10^HLFNC

This function, $$OLD10^HLFNC, is provided for backwards compatibility. It provides the non-compliant, pre-HL*1.6*38 functionality of $$M10^HLFNC.

Patch HL*1.6*38 provides an HL7-compliant check digit calculator using the M10 algorithm.

Usage	`$$OLDM10^HLFNC(ck_id, [hlecde])`	
Input	ck_id	(required) First component (the ID *number*) for an HL7 field whose data type is composite id with check digit (CK). Must be numeric.
	hlecde	(optional) Component separator to use. Pass a 1-character string containing the component separator only, or pass the 4-character string containing the HL7 encoding characters as defined in HL("ECH") (i.e., the component separator is the first character). The only rule is that $E(cmp_sep) must evaluate to the component separator character to use. If this parameter is not passed, the HLECH variable must contain the HL7 encoding characters already, in the format defined by **V*IST*A** HL7. If passed, this parameter overrides HLECH.
Output	return value	First three CK data type components, separated by specified component separator.

12.6.12$$OLDM11^HLFNC

This function, $$OLD11^HLFNC, is provided for backwards compatibility. It provides the non-compliant, pre-HL*1.6*38 functionality of $$M11^HLFNC.

Patch HL*1.6*38 provides an HL7-compliant check digit calculator using the M11 algorithm.

Usage	`$$OLDM11^HLFNC(ck_id, [hlecde])`	
Input	ck_id	(required) First component (the ID *number*) for an HL7 field whose data type is composite id with check digit (CK). Must be numeric.
	hlecde	(optional) Component separator to use. Pass a 1-character string containing the component separator only, or pass the 4-character string containing the HL7 encoding characters as defined in HL("ECH") (i.e., the component separator is the first character). The only rule is that $E(cmp_sep) must evaluate to the component separator character to use. If this parameter is not passed, the HLECH variable must contain the HL7 encoding characters already, in the format defined by **V*IST*A** HL7. If passed, this parameter overrides HLECH.
Output	return value	First three CK data type components, separated by specified component separator.

A P P E N D I X

A. V*IST*A HL7 Process Flows

A.1 Sending a Message (GENERATE^HLMA or DIRECT^HLMA)

A. M application calls INIT^HLFNC2 to set up HL7 environment variables.

B. M application builds message body in local or global array.

C. M application calls GENERATE^HLMA or DIRECT^HLMA, passing the event driver protocol to invoke as a parameter, and the location of the message body as an input variable.

1. **Steps for Every Outgoing Message. V*IST*A** HL7 performs the following steps:
   a. Validate the event driver protocol and other parameters.
   b. Create a parent entry to hold the message body in File #772.
   c. Execute the event driver protocol's entry action.
   d. Set the parent entry's status in File #772 to "BEING GENERATED".
   e. File the message body in the parent entry in File #772.
   f. Build a supplemental recipients array, based on the contents (if any) of the HLL("LINKS") array.
2. **Synchronous.** If the application call is to DIRECT^HLMA, there must be at least one recipient. If there is, **V*IST*A** HL7 performs the following steps, and then goes to step D:
   a. Try to open a connection to the target system. If this fails, return error and quit.
   b. If open succeeds, create a new child entry in File #773, with a new message ID.
   c. Create the message header and file it in the new child entry.
   d. Send the message (header from File #773, body from File #772) over the open channel to the target system.
   e. Receive the expected response from the target system (store the response message body in a new entry in File #772, and response header in a new entry in File #773).
   f. Set the special HL variables to enable the calling routine to process the response (HLQUIT, HLNODE and HLNEXT).
   g. Return success and quit.
3. **Asynchronous.** If the application call is to GENERATE^HLMA, **V*IST*A** HL7 obtains the list of subscriber protocols. It then loops through each subscriber protocol as follows:
   a. **Router.** If the subscriber protocol is a router protocol, **V*IST*A** HL7 executes the following steps, which conclude the processing for this subscriber protocol:
      i. Execute the Routing Logic code of the protocol (for more information, please see the "Dynamic Addressing" section of the "Advanced Interface Issues" chapter).

ii. Add any new entries placed in HLL("LINKS") by the routing logic to V*IST*A HL7's supplemental recipients array.

b. **Outgoing.** If the subscriber protocol has a link, V*IST*A HL7 performs the following steps, which conclude the processing for this subscriber protocol:

i. If the link type is TCP, create a new child entry for the outgoing message in File #773 with a new message ID; create a message header; store the message header in the File #773 child entry, set the Logical Link field (this also sets the "AC","O" cross reference subscript in File #773).

ii. If the link type is HLLP, MailMan or X3.28, create a new child entry in File #772 including a new message ID; set the Logical Link field to the appropriate link (this also sets the "A-XMIT-OUT" cross reference subscript in File #772.)

iii. Set the status of the child entry to "Pending Transmission".

c. **Same-System.** If the subscriber protocol has no link, but is not a router protocol, V*IST*A HL7 treats the message delivery as a same-system delivery. It performs the following steps, which conclude the processing for this subscriber protocol:

i. Create a child entry in File #772 for the "outgoing" message, including a new message ID.

ii. Create a message header for the "outgoing" message.

iii. Create new parent and child entries in File #772 for the "incoming" message.

iv. Copy the message (header and body) into the new "incoming" parent entry.

v. Set the status of the "outgoing" message's child entry to "Awaiting Acknowledgment", and the status of the "incoming" message's parent entry to "Awaiting Processing".

vi. Set the special HL variables for the processing routine (HLQUIT, HLNODE and HLNEXT).

vii. Execute the processing routine of the subscriber protocol. If the processing routine calls GENACK^HLMA1, the creation, delivery and processing of the acknowledgment is conducted within this process, which may then set the status of the "outgoing" message to "Successfully Completed".

viii. Set the status of the "incoming" message to "Successfully Completed".

4. **Dynamic Addressing.** V*IST*A HL7 checks for any dynamically addressed recipients added to the supplemental recipients array, from before and during subscriber protocol processing. For each recipient found, V*IST*A HL7 performs the following steps:

a. If the link type is TCP, create a new child entry for the outgoing message in File #773; create a message header; store the message header in the File #773 child entry; set the Logical Link field (this also sets the "AC","O" cross reference subscript in File #773).

b. If the link type is HLLP, MailMan or X3.28, create a new child entry in File #772; set the Logical Link field to the appropriate link (this also sets the "A-XMIT-OUT" cross reference subscript in File #772.)

c. Set the status of the child entry to "Pending Transmission".

D. V*IST*A HL7 executes the event driver protocol's exit action.

E. **V***IST***A** HL7 sets the status of the message's parent entry in File #772, based on the result of the processing executed in prior steps.

F. Control is returned to the M application.

A.2 Outgoing Filer

A. The outgoing filer (^HLCSOUT) loops continuously through the "A-XMIT-OUT" cross reference of File #772. Each entry in this cross-reference comes from a child entry added to File #772, and represents one message to transmit over one particular link. For each entry in the cross-reference:

   1. Obtain the outgoing link and child message pointer from the "A-XMIT-OUT" cross-reference of the outgoing message's child entry in File #772.
   2. Build a message header to use for the outgoing message.
   3. Create an entry in the appropriate HL Logical Link (File #870) entry's "Out" queue. Set its status to "Stub Record".
   4. Write the message body (from the parent entry in File #772) and message header to the new entry in the "Out" queue.
   5. Set the status of the new entry in the "Out" queue to "Pending".
   6. Set the status of the outgoing message's child entry in File #772 corresponding to "Awaiting Acknowledgment".

B. Return to Step A.

Note: The "A-XMIT-OUT" cross-reference is only set for outgoing messages over X3.28, HLLP and MailMan links only (not TCP).

Note: Outgoing filers (and the entire process of copying a message body from the Message files to links' "Out" queues and adding a header) are not used for TCP links. Instead, the header is already formed, the message is ready to deliver, and the TCP links' background processes pick up the outbound message directly from the "AC","O" cross reference in File #773.

A.3 HLLP Link

A. Check if the link has a valid device (HLLP Device field). If not, quit.

B. Set the link's "Time Started" field to now, "Time Stopped" to null, "Shutdown LLP?" to No, "Device Type" to "SH", and "Task Number" equal to the TaskMan number of the job that started the link.

C. Set the link's status to "Open", and open the link's device. If the open fails, set the link's state to "OpenFail", wait 5 seconds, and repeat step C.

D. Retrieve the link's HLLP settings (start character, end character, LLP version number, etc.)

E. **Read Incoming Messages.**

1. Perform a read on the device. If the read times out, go to step G.
2. If the read succeeds, set the link's status to "Reading".
3. Purge all messages with a status of "Done" from the beginning of the "In" queue to the front pointer of the queue, minus the link's "Queue Size" setting.
4. Add a new entry to the link's "In" queue.
5. Increment the "In" queue's back pointer by 1.
6. Repeat the reads to gradually read in a complete message into the new "In" queue entry; stop when the LLP End Block character is encountered. If a timeout is encountered at any point, write a NAK to the device and repeat step E.
7. If the incoming block is a NAK character, set the error field of the last transmitted message in the "Out" queue, based on the error contained in the NAK. Increment the retry count by 1. Go to step F.
8. Check the validity of the message; set the link's status to "Validate". If the message is invalid, send a NAK back to the other system and repeat step E.

F. **Write Outgoing Messages.**

1. Purge all messages with a status of "Done" from the beginning of the link's "Out" queue to the front pointer of the queue, minus the link's "Queue Size" setting.
2. Check to see if there is a message in the link's "Out" queue to deliver. If there is:
3. Increment the "Out" queue's front pointer by 1.
4. Set the link's status to "Writing".
5. Write the message (header and body) to the link's device.
6. If the retry count has been exceeded, stop attempting to write the message, and go to step G.
7. Set the status of the message's entry in the link's "Out" queue to "Done".
8. Go to step E.

G. Go to Step F.

A.4 X3.28 Link

V*IST***A** HL7's implementation of X3.28 processing precisely mirrors the expected behavior described in the *Health Level Seven Implementation Support Guide*. For more information, please see "Appendix C: Lower Layer Protocols" in the *Health Level Seven Implementation Support Guide*.

A.5 Outgoing MailMan Link

A. Set the link's "Time Started" field to now, "Time Stopped" to null, "Shutdown LLP?" to No, "Device Type" to "MM", and "Task Number" equal to the TaskMan number of the job that started the link.

B. Set the link's status to "Idle".

C. Purge all messages with a status of "Done" from the beginning of the "Out" queue to the front pointer of the queue, minus the link's "Queue Size" setting.

D. Look in the link's "Out" queue to see if there is a message to go out. If there is:

   1. Increment the "Out" queue's front pointer by 1.
   2. Set the link's status to "Writing".
   3. Check that the link has a valid mail group. If not, set an error condition and go to step E.
   4. Create a subject for the outgoing mail message: "HL7 Message [date/time] from Station [name]".
   5. Load the message text (header and body) from the "Out" queue entry into a new mail message.
   6. Send the message to the mail group specified in the link's "Mail Group" field.
   7. Set the status of the message's entry in the link's "Out" queue to "Done".

E. If an error condition was encountered, send a MailMan message indicating the error to the Postmaster. The message says, "Unable to transmit HL7 message due to the following Application Error: …"

F. Go to Step B.

A.6 TCP Link (Outgoing)

A. If the link is persistent, open a connection to the target system. If the open fails, set the link's status to "OpenFail" and repeat step A.

B. Check the "Out" queue for TCP messages. For TCP messages, the "Out" queue is the "AC","O" cross-reference of File #773. Set the link's status to "Check Out". If there is a message to send in the "Out" queue:

   1. If the link is non-persistent, open a connection to the target system. If the open fails, end the processing for this message and go to step B.
   2. Set the link's status to "Send". Write the message to the target system. If the write fails, end the processing for this message and go to step B.
   3. Update the link's "Message Sent" counter.
   4. Determine the type of response expected. If no response is needed (Commit Ack and Application Ack are both set to "NE"), set the status of the outgoing message in File #773 to "Successfully Completed". Go to step B.
   5. Read the response from the target system:
      a. Set the link's status to "Reading".
      b. Create new parent entry in File #772 for the incoming response message body and child entry in File #773 for the incoming response message header.
      c. Read the response into the response's entries in File #772 and File #773.
      d. If the response is a commit acknowledgment:
         i. If only a commit ack is expected, set the status of the outgoing message in File #773 to "Successfully Completed". Go to step B.
         ii. If an application ack is also expected (making this a deferred two-phase commit), set the status of the outgoing message in File #773 to "Awaiting Application Acknowledgment". Go to step B.
      e. If the response is an application acknowledgment, and that is what is expected, set the status of the outgoing message in File #773 to "Successfully Completed". Go to step B.
      f. If the response is not the expected response, set the status of the outgoing message in File #773 to "Error". Go to step B.
   6. If the retry count for the current message exceeds the link parameter:
      a. Send an alert to the HL7 mail group for alerts notifying it that the HL7 message exceeds the retry count.
      b. Perform the appropriate action based on the link's "Exceed Re-Transmit Action" setting (ignore, shut down the link, or restart the link).

C. If there are no more messages in the TCP "Out" queue, set the link's status to "Idle". If the link is non-persistent:

   1. Set the link's status to "Retention".
   2. Check the TCP "Out" queue for the duration of the retention time, as found in the link's "Retention" setting or the "Default Retention Time" site parameter. If no message is found in

that time, leave the link in a state of "Retention" and quit/halt (terminate processing). Please see the "TCP Link Manager" process flow for how non-persistent TCP links are restarted.

D. Go to step B.

A.7 TCP Link Manager

The TCP Link Manager starts up background processes as needed for non-persistent outgoing TCP links. *Persistent* TCP links still require one process per running link.

A. The "AC" cross-reference in File #773 contains all TCP links over which outgoing messages are waiting to transmit. The TCP Link Manager loops through this cross-reference. For every TCP link found, the TCP Link Manager performs the following actions:

   1. Check that there are messages waiting to go out over the link. If not, end processing for this link.
   2. Look to see if there is an entry in the Task Number field for the link in File #870. If there is, the link is considered to be running. End processing for this link.
   3. Look in the local array maintained by TCP Link Manager to see if the TCP Link Manager has requested a task to be started for the link. If a task for the link has been requested to start, and that request is less than 30 minutes in the past, end processing for this link.
   4. If a task has been requested to be started, and the time requested is more than 30 minutes ago, there is a problem. Perform the following steps, and then end processing for this link:
      a. Set the "Shutdown?" field of the link to "Yes"
      b. Attempt to shut down the link
      c. Send an alert to the HL7 mail group for alerts, notifying it that the link has been shutdown due to TaskMan being unable to process a task request.
   5. Start a new task for the link. Save the requested task number in the local array maintained by the TCP Link Manager.

B. Return to Step A.

A.8 TCP Listeners (All Types)

There are three types of TCP listeners: Single-threaded, Multi-threaded for Cache NT, and UCX Multi-threaded for DSM for OpenVMS. While each of these listeners is started in a different way, once they begin listening on a TCP port, the processing for incoming messages is the same. Each listens for new, incoming messages and for deferred acknowledgments.

A. The listener process sets the status of its link to "Open".

B. The listener goes into a loop to read new incoming messages. For each new message it reads, it executes the following steps:

   1. Set the link's status to "Reading".
   2. Create new parent entry in File #772 for the incoming response message body and child entry in File #773 for the incoming response message header.
   3. Read the message into the new entries in File #772 and File #773.
   4. Update the counters for the listener's links.
   5. Validate the incoming message's header. If the header is invalid, reject it with a commit reject or application reject acknowledgment and end processing for this message.
   6. Check for duplicate message already received, using the receiving application, message ID, and complete message header for the comparison. If the message is a duplicate:
      a. Set the message's status to "Error", and Error Type to "Duplicate Message".
      b. Find the original response, and re-sends it to acknowledge the duplicate message.
   7. If the message is a deferred acknowledgment, return a commit acknowledgment if necessary, set the message status to "Awaiting Processing", and set the "AC", "I" cross reference in File #773. End processing for this message. **Note:** The inbound filer will pick up the message for processing.
   8. If the message is a new message:
      a. Set the special HL variables for the processing routine (HLQUIT, HLNODE and HLNEXT).
      b. Parse the message header for message type, event type, version ID and sending application and match these to find the appropriate event driver protocol. Then use the receiving application to find the appropriate subscriber protocol.
      c. Execute the entry action of the subscriber protocol.
      d. Call the processing routine of the subscriber protocol to process the deferred acknowledgment.
      e. The processing routine may call GENACK^HLMA1 to return an Ack. Check if it did, in which case there is a response to send back. If so, write the response back over the open link.
      f. Execute the exit action of the subscriber protocol.
      g. Set the status of the incoming message in File #773 to "Successfully Completed".

C. Set the link's status to "Idle."

D. Go to step B.

A.9 Incoming Message (MailMan)

A. The incoming message must be addressed to "S.HL V16 SERVER" on your system.

B. The server option processes the incoming message by executing each of the following steps:

   2. Create a parent entry in File #772 for the incoming message.
   3. Copy the message (header and body) from the MailMan message to the new File #772 entry.
   4. Validate the message header.
   5. Create a child entry in File #772 for the incoming message.
   6. Send a commit ack if required.
   7. Set the special HL variables for the processing routine (HLQUIT, HLNODE and HLNEXT).
   8. If the incoming message is an acknowledgment (containing an MSA segment):
      a. Obtain the Message ID of the original message from the MSA segment
      b. Update the status of the original message's child entry to "Successfully Completed".
      c. Find the appropriate event driver protocol from the original message.
      d. Execute the event driver protocols' entry action, response processing routine, and exit action.
   9. If the incoming message is a new message:
      a. Parse the message header for message type, event type, version ID and sending application and match these to find the appropriate event driver protocol.
      b. Then use the receiving application to find the appropriate subscriber protocol.
      c. Execute the subscriber protocols' entry action, response processing routine, and exit action.
   10. Update the status of the incoming message's child entry to "Successfully Completed".

C. The server option deletes the MailMan message from the server mailbox.

A.10 Incoming Filer

A. Loop through each link in File #870. For each link:

1. Make sure another incoming filer is not processing the same link. End the processing for this link if another one is.
2. If this is a TCP link, and there are one or more deferred acknowledgments waiting for it that have been received by TCP listeners (stored in parent and child entries in File #772 and #773), perform the following steps for each deferred acknowledgment, and then end the processing for this link:
    a. Set the special HL variables for the processing routine (HLQUIT, HLNODE and HLNEXT).
    b. Obtain the Message ID of the original message from the MSA segment
    c. Update the status of the original message's child entry to "Successfully Completed".
    d. Find the appropriate event driver protocol from the original message.
    e. Execute the event driver protocols' entry action, response processing routine, and exit action.
    f. Set the status of the incoming message to "Successfully Completed".
3. If this is not a TCP link, check the link's "In" queue for messages with a status of "Pending". For each received pending message in the "In" queue, perform the following steps:
    a. Purge all messages with a status of "Done" from the beginning of the link's "In" queue to the front pointer of the queue, minus the link's "Queue Size" setting.
    b. Increment the front queue pointer by 1.
    c. Create a parent entry in File #772 for the incoming message.
    d. Copy the message (header and body) from the "In" queue entry to the new File #772 entry.
    e. Validate the message header.
    f. Create a child entry in File #772 for the incoming message.
    g. Send a commit ack if required.
    h. Set the special HL variables for the processing routine (HLQUIT, HLNODE and HLNEXT).
    i. If the incoming message is an acknowledgment (containing an MSA segment):
        i. Obtain the Message ID of the original message from the MSA segment
        ii. Update the status of the original message's child entry to "Successfully Completed".
        iii. Find the appropriate event driver protocol from the original message.
        iv. Execute the event driver protocols' entry action, response processing routine, and exit action.
    j. If the incoming message is a new message:
        i. Parse the message header for message type, event type, version ID and sending application and match these to find the appropriate event driver protocol.

ii. Then use the receiving application to find the appropriate subscriber protocol.

iii. Execute the subscriber protocols' entry action, response processing routine, and exit action.

k. Set the status of the incoming message to "Successfully Completed".

l. Set the status of the message in the link's "Out" queue to Done.

B. Go to Step A.

A P P E N D I X

B. Version 1.5 Options Distributed in V. 1.6

The **V*IST*A** HL7 V. 1.5 package consisted of 7 files and 10 routines. It supported two Lower Layer Protocols (DHCP MailMan and the Hybrid Lower Layer Protocol HLLP). For its time, it provided a mechanism for applications to build and transmit data in the HL7 V. 2.1 standard. This was a breakthrough as it provided a common tool for all applications follow a standard development path.

While version 1.6 is now the current release of **V*IST*A** HL7, it still exports options, files and routines supporting **V*IST*A** HL7 V. 1.5 interfaces. This is so that existing interfaces dependent on the behavior of **V*IST*A** V. 1.5 would not be broken.

B.1 All Remaining V. 1.5 Interfaces Can Be Converted to V. 1.6

With the release of patch HL*1.6*57, **V*IST*A** HL7 1.6 is now fully backwards compatible with version 2.1 of the HL7 Standard. In particular, event types for version 2.1 messages are no longer required. All HL7 V. 2.1 transactions currently interfaced using **V*IST*A** HL7 V. 1.5 can now be converted to **V*IST*A** HL7 V. 1.6, and should be upgraded as soon as possible. There should be no new development using the 1.5 interface method.

All current work to enhance our HL7 capabilities is building on V. 1.6 of **V*IST*A** HL7. In order to advance the overall messaging in **V*IST*A** to functionality levels beyond those offered by V. 1.6, we need to finish migrating V. 1.5 interfaces to V. 1.6.

B.2 Differences between V. 1.5 and V. 1.6

The following table lists the major architectural differences between the two versions of **V*IST*A** HL7:

V. 1.5 Interface Method	V. 1.6 Interface Method
One sender and one receiver per message.	One sender, one or more receivers.
Sender and receiver must be on different systems.	Sender and receiver can be on the same or different systems.
Messages must go through a communications protocol.	Messages sent to applications on the same system do not have to go through a communications protocol.
No support for event types.	Event types are supported.
Applications for "other" systems are stored in File #770 (Non-DHCP Application Parameter).	Applications for "other" systems are stored in File #771 (HL7 Application Parameter).
Communications parameters are stored along with the application in File #770.	Communications parameters are stored in links, in File #870. They can be shared by multiple applications.
Supported LLPs: HLLP and MailMan.	Supported LLPs: HLLP, MailMan, X3.28, and MLLP over TCP/IP .
MailMan Server option: HL SERVER.	MailMan Server option: HL V16 SERVER.
Interface Editing: through VA FileMan.	Interfaces are edited through a set of ListMan options (1.6 upon release) or through a streamlined set of ScreenMan options (after HL*1.6*57).

For more information on V. 1.5 interfaces, please the documentation for the V. 1.5 HL7 package. The V. 1.5 documentation is archived on the **V*IST*A** HL7 home page, at:

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

B.3 V. 1.5 Options

The following vestigial V. 1.5 options are exported by **V*IST*A** HL7 V. 1.6

- Non-DHCP Application Parameter Enter/Edit [HL Edit Site Param]
- Initiate Background Task [HL Task]
- Non-DHCP Application Parameters Print/Display [HL Print Site Param]
- Start/Stop Log of HL7 Transmissions [HL Transmission Log]

B.3.1 Non-DHCP Application Parameter Enter/Edit

Use the Non-DHCP Application Parameter Enter/Edit option to

- Enter V.1.5 non-DHCP (in V.1.5 terminology, this means on a different system) applications — applications that will be communicating with the DHCP system via the HL7 Protocol — in the HL7 Non-DHCP Application Parameter file (#770).
- Enter or edit parameters associated with non-DHCP applications.
- Delete a non-DHCP application.

The applications entered in this file are referred to as non-DHCP applications simply as a way of distinguishing them from the DHCP system with which they will be communicating. These non-DHCP applications could also be other DHCP applications (e.g., Radiology sending/receiving HL7 messages to/from MAS) on a different system. A non-DHCP application might have more than one entry in this file depending on how many DHCP applications it communicates with and how many versions of the HL7 Protocol it utilizes.

Example

```
Select HL7 Main Menu Option:  Non-DHCP Application Parameter Enter/Edit

Select HL7 NON-DHCP APPLICATION PARAMETER NAME: AMCH-RADIOLOGY
  ARE YOU ADDING 'AMCH-RADIOLOGY' AS
    A NEW HL7 NON-DHCP APPLICATION PARAMETER (THE 3RD)? Y  (YES)
   HL7 NON-DHCP APPLICATION PARAMETER DHCP STATION NUMBER: 500         ALBANY
   HL7 NON-DHCP APPLICATION PARAMETER NON-DHCP FACILITY NAME: AMCH XRAY
   HL7 NON-DHCP APPLICATION PARAMETER DHCP APPLICATION: EDR-MAS          ACTIVE
NAME: AMCH-RADIOLOGY//
NON-DHCP FACILITY NAME: AMCH XRAY//
DHCP STATION NUMBER: 500//
MAXIMUM BLOCK SIZE: 245//
NUMBER OF RETRIES: 3//
HL7 DEVICE: HLLP
HL7 VERSION NUMBER: 2.1//
DHCP APPLICATION: EDR-MAS//
LOWER LEVEL PROTOCOL TIMEOUT: 30//
MAIL GROUP:
HL7 PROCESSING ID: P  PRODUCTION
PURPOSE:
  1>Data exchange between Albany VAMC and Albany Medical Center Hospital
EDIT Option: <RET>
```

B.3.2 Initiate Background Task

Use the Initiate Background Task option to create a background task to start up the lower level protocol routine for a specific V. 1.5, non-DHCP application. *You must select a non-DHCP application for which an HL7 device has been defined. Non-DHCP applications that use MailMan as their lower level protocol do not need a background task.* This option also checks to make sure a background task is not already queued in TaskMan. *If a task is already queued, a second task will not be started.*

Example

```
Select HL7 Main Menu Option:  Initiate Background Task

Note:  You must select a Non-DHCP Application for which an HL7 Device has
been defined.
Select HL7 NON-DHCP APPLICATION PARAMETER NAME:    AMCH-RADIOLOGY         500
  AMCH XRAY      EDR-MAS
```

B.3.3 Start/Stop Log of HL7 Transmissions

The Start/Stop Log of HL7 Transmissions option is a diagnostic tool for V. 1.5 applications. Use it to:

- Test the HL7 interface when the HL7 HLLP is used as the communications protocol.
- Check that HL7 messages are being properly exchanged with a particular non-DHCP application.

This option should not be used if your application uses DHCP MailMan as the lower level protocol.

- If logging for a particular application is on, the option prompts you to turn it off.
- If login is off and you choose to turn it on, you're asked if you wish to purge existing log entries. *You should purge existing entries when you start the transmission log to ensure a clean start.*

Log data is stored descendant from the ^TMP("HL",HLION) global node where HLION is equal to the name of the HL7 device in field #6 of the HL7 Non-DHCP Application Parameter file entry. *Use the operating system routine that lists a global (%G, %GL, etc.) to view the contents of the log.*

When your testing is complete, stop the log to save disk space and reduce global sets. It can be purged at any time by killing the ^TMP("HL",HLION) global.

B.3.4 Non-DHCP Application Parameters Print/Display

Use the Non-DHCP Application Parameters Print/Display option to list the V. 1.5 non-DHCP applications in the HL7 Non-DHCP Application Parameter file (#770).

Example

```
HL7 Non-DHCP Application Parameters            NOV 28,1994  13:45    PAGE 1
-------------------------------------------------------------------------------

NAME: AMCH-RADIOLOGY                       DHCP STATION NUMBER: 500
  NON-DHCP FACILITY NAME: AMCH XRAY        MAXIMUM BLOCK SIZE: 245
  NUMBER OF RETRIES: 3                     HL7 DEVICE: HLLP
  HL7 VERSION NUMBER: 2.1                  DHCP APPLICATION: EDR-MAS
  LOWER LEVEL PROTOCOL TIMEOUT: 30         HL7 PROCESSING ID: PRODUCTION
 PURPOSE:   Data exchange between Albany VAMC and Albany Medical Center
 Hospital
  START/STOP TRANSMISSION LOG: STOP LOG

NAME: EDR-MAS                              DHCP STATION NUMBER: 500
  NON-DHCP FACILITY NAME: BDC              MAXIMUM BLOCK SIZE: 245
  NUMBER OF RETRIES: 3                     HL7 VERSION NUMBER: 2.2
  DHCP APPLICATION: EDR-MAS                LOWER LEVEL PROTOCOL TIMEOUT: 30
  MAIL GROUP: EDR-MAS                      HL7 PROCESSING ID: DEBUG

NAME: IVM CENTER                           DHCP STATION NUMBER: 500
  NON-DHCP FACILITY NAME: 724              MAXIMUM BLOCK SIZE: 245
  HL7 VERSION NUMBER: 2.1                  DHCP APPLICATION: IVM
  MAIL GROUP: IVM TRANSMISSIONS            HL7 PROCESSING ID: PRODUCTION
```

G L O S S A R Y

Accept Acknowledgment	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."[10]
Acknowledgment	A software handshake method used by HL7. When a system receives a message, it may send back an "accept acknowledgment" message to confirm the data was received. It may also send back an "application acknowledgment" message to confirm if the data received was appropriate. For more information, please see the "Acknowledgments" chapter.
Application	In the terminology of **V*IST*A** HL7, an application implements the interface between two systems. A **V*IST*A** HL7 application is stored in **V*IST*A** HL7 file entries. The application consists of one entry in the HL7 Application Parameter file, and *protocols* representing each transaction (i.e., HL7 message type), set up to be exchanged in the interface.
Application Acknowledgment	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."[11]
Commit Acknowledgment	See Accept Acknowledgment
EAI	See Enterprise Application Integration.
Enterprise Application Integration	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 **V*IST*A** HL7, an event driver protocol represents the sending application's side of a transaction for a particular message type/event type, whether the message originates on the **V*IST*A** side of the interface, or on the other side of the interface.
Event Type	Trigger event code (one component of a message header's Message Type field) defined by *HL7 table 0003 - Event typ*e.
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.

[10] Health Level Seven, *Health Level Seven, Version 2.3.1*, copyright 1999, p. E-1.
[11] Health Level Seven, *Health Level Seven, Version 2.3.1*, copyright 1999, p. E-2.

Filer	One of **V*IST*A** HL7's background processes. *Incoming* filers move received messages over **V*IST*A** HL7 HLLP and X3.28 links to **V*IST*A** HL7's messages files, and process the message. *Outgoing* filers deliver messages generated by **V*IST*A** programs to HLLP, X3.28 and MailMan links for delivery to other systems.
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.
Hybrid Lower Layer Protocol (HLLP)	A *Lower Layer Protocol* for communications over a LAN that may involve RS-232 connections. It is a more efficient LLP than X3.28, but still suitable for use over generally error-free RS-232 connections.
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 **V*IST*A** HL7 to a particular destination. The destination may be a device (defined in the Device file), a TCP/IP address and port, or a system that is reached through VA MailMan.
Logical Link	See Link.
Lower Layer Protocol (LLP)	Defined by the *HL7 Implementation Guide*. LLPs provide for a degree of lower layer communications functionality, such as flow control and error recovery, needed between systems in a networked environment. Examples include *Hybrid Lower Layer Protocol* (HLLP) and *Minimal Lower Layer Protocol* (MLLP).
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."[12]
Message Type	Message type code (one component of a message header's Message Type field) defined by *HL7 table 0076 - Message typ*e.
Minimal Lower Layer Protocol (MLLP)	A very simple *Lower Layer Protocol* that simply delimits messages. It is used ir LAN-based networking environments such as TCP/IP that provide a reliable byte stream (flow control and error recovery are provided by the network), but insufficient session control to support HL7.
Persistent	When a link is persistent, the connection between the two systems is kept open even when no messages are being exchanged. When a link is non-persistent, the connection is closed when there are no messages to exchange.

[12] Health Level Seven, *Health Level Seven, Version 2.3.1*, copyright 1999, p. E-18.

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 entry on your system is used to represent the receiving application (subscriber protocol).
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."[13]
Subscriber	An application that subscribes to a particular event point, registering its interest so it can receive unsolicited updates.
Subscriber Protocol	For **V*IST*A** HL7, a subscriber protocol represents the receiving application's side of a particular transaction for a particular message type/event type, whether the message is being received by the **V*IST*A** side of the interface, or by the other side of the interface.
Queue	Incoming and outgoing queues are used by **V*IST*A** HL7 to manage message delivery over links. Each link has an "In" and an "Out" queue.
Unsolicited Update	A message representing an event that is initiated by the application in whose domain the event occurs.
X3.28	A *Lower Layer Protocol* based on ANSI X3.28, for use over simple RS-232 circuits where flow control and error recovery are major issues.

[13] Health Level Seven, *Health Level Seven, Version 2.3.1*, copyright 1999, p. E-28.

I N D E X