2. Control

Chapter Chair	Grahame Grieve Kestral Computing Pty Ltd
Chapter Chair: and Editor:	Anthony (Tony) Julian Mayo Clinic
Chapter Chair	Doug Pratt Siemens Medical Solutions Health Services Corporation
Chapter Chair	Scott Robertson Kaiser Permanente
Chapter Chair	Rene Spronk HL7 The Netherlands
Vocabulary Facilitator	Sandy Stuart Kaiser Permanente
Conformance SIG co-chairs	Lisa Carnahan NIST
	John Lyons Siemens Medical Solutions
	Frank Oemig Agfa HealthCare GmbH

2.1 CHAPTER 2 CONTENTS

Health Level Seven, Version 2.5.1 © 2007. All rights reserved.

2.2 INTRODUCTION

The Control chapter of this Standard defines the generic rules that apply to all messages. Subsequent sections define functionally specific messages to be exchanged among certain applications. The specific aspects of message definition that are addressed herein are:

a) the form to be used in functional chapters for describing messages. This includes their purpose, their contents, and the interrelationships among them. This form is called an abstract message definition because it is purely a level 7 (application) definition.

b) the HL7 encoding rules for converting an abstract message into a string of characters that

comprises an actual message.

c) the programming procedures required to exchange messages using the HL7 specifications

d) the anticipated relationship with lower level protocols.

e) certain message segments that are components of all messages.

f) a single message, the acknowledgment message, that may be used unchanged in multiple applications.

2.3 CONCEPTUAL APPROACH

2.3.1 Trigger events

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**.

Note: No assumption is made about the design or architecture of the application system creating the unsolicited update. The scope of HL7 is restricted to the specification of messages between application systems and the events triggering them.

HL7 allows the use of trigger events at several different levels of data granularity and inter-relationships. For example, most Patient Administration (ADT) trigger events concern single objects (such as an admit event, which creates a message that contains data about a single person and/or account). Other ADT trigger events are concerned with relationships between more than one object (e.g., the merge events, which specify patient or account merges). Some ADT trigger events pertain to a collection of objects that may have no significant inter-relationships (e.g., a record-oriented location-based query, whose response contains data about a collection of inpatients who are related only temporarily, by local geography).

2.3.2 Acknowledgments: original mode

When the unsolicited update is sent from one system to another, this acknowledgment mode specifies that it be acknowledged at the application level. The reasoning is that it is not sufficient to know that the underlying communications system guaranteed delivery of the message. It is also necessary to know that the receiving application processed the data successfully at a logical application level.

The acknowledgment may contain data of interest to the system that initiated the exchange. For example, if a patient care system has processed the trigger event **a lab test is ordered for a patient**, it may send an unsolicited update to a lab application identifying the patient, the test ordered, and various other information about the order. The ancillary system will acknowledge the order when it has processed it successfully. For some pairings of patient care and ancillary department systems the acknowledgment may also include the ancillary identification number that was assigned (HL7 does not require Order Entry and Results Reporting applications to interface in this manner, but it supports those that do).

The HL7 Standard makes no assumptions about the ownership of data. It also makes no requirements of its own on the subsequent action of the recipient of data, nor does it make any assumption about the design or architecture of the receiving application system. The scope of HL7 is restricted to the specification of messages between application systems, and the events triggering them. HL7 does not explicitly support, but

Health Level Seven, Version 2.5.1 © 2007. All rights reserved.

can be used with, systems that support store and forward and data broadcast facilities (see the HL7 Implementation Support Guide).

The HL7 Standard makes no functional interpretation of the requirement that a system commit the data in a message to its database before acknowledging it. All that is required is that the receiving system accept responsibility for the data, providing the same integrity test that it would apply to data from any source. To continue the prior example, the ancillary system may acknowledge the order after placing it in an input queue, expecting to fully process the order into its database at a future time. The only assumption is that the input queue is maintained at the same level of integrity as the database.

2.3.3 Acknowledgments: enhanced mode

The HL7 acknowledgment paradigm has been extended to distinguish both accept and application acknowledgments, as well the conditions under which each is required. 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. 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.

2.3.4 Queries

Query documentation including messages, segments, special protocols, implementation considerations and examples have been moved to chapter 5. The unsolicited display messages were also moved because their message syntax is query-like in nature.

2.4 COMMUNICATIONS ENVIRONMENT

The HL7 Standard defines the messages as they are exchanged among application entities and the procedures used to exchange them. As such, it conceptually operates at the seventh level of the ISO model for Open System Interconnection (OSI). It is primarily concerned with the data content and interrelationship of messages and with communicating certain application-level error conditions.

Since the OSI protocols are not universally implemented, the HL7 Working Group is interested in providing standards that will be useful in the interim. It is also recognized that there is now, and will continue to be, interest in communicating health data among systems operating in communications environments that provide a high level of functionality, but use protocols other than ISO OSI. The universe of environments of interest to HL7 includes, but is not restricted to:

a) ad hoc environments that do not provide even basic transport reliability. Such environments consist of point-to-point RS-232 links, modems, and even LANs, if their connection to host computers is made via RS-232 communications links. Until OSI high level standards become truly prevalent, many healthcare interfaces will be implemented over such links. In such an environment, the HL7 Lower Level Protocols (LLP) may be used between systems to enhance the capabilities of the communications environment. The HL7 Lower Level Protocols are defined in the HL7 Implementation Guide, which is not an official part of the Standard.

b) environments that support a robust transport level, but do not meet the high level requirements. This includes environments such as TCP/IP, DECNET, and SNA.

c) ISO and proprietary networks that implement up to presentation and other high level services. IBM's SNA LU6.2 and SUN Microsystems's NFS are examples of complete proprietary networks.

d) two or more applications running on the same physical and/or logical machine that are not tightly integrated. In these environments, the messaging capabilities may be provided by inter-process communications services (e.g., Pipes in a UNIX System).

The HL7 Standard assumes that the communications environment will provide the following capabilities:

a) error free transmission. Applications can assume that they correctly received all of the transmitted bytes in the order in which they were sent. This implies that error checking is done at a lower level. However, sending applications may not assume that the message was actually received without receiving an acknowledgment message.

b) character conversion. If the two machines exchanging data use different representations of the same character set, the communications environment will convert the data from one representation to the other.

c) message length. HL7 sets no limits on the maximum size of HL7 messages. The Standard assumes that the communications environment can transport messages of any length that might be necessary. In practice, sites may agree to place some upper bound on the size of messages and may use the message continuation protocol, described later in this chapter, for messages that exceed the upper limit.

Note: Just as HL7 makes no assumptions about the design or architecture of the application systems sending and receiving HL7 messages, it makes no assumptions about the communications environment beyond those listed above. In particular, aside from the above assumptions, the communications environment, including its architecture, design and implementation, is outside the scope of HL7.

2.5 MESSAGE FRAMEWORK

This section defines the constituents of messages and provides the methodology for defining abstract messages that are used in later chapters. Message construction rules can be found in section 2.6.

2.5.1 Messages

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 Patient Administration (ADT) data from one system to another. A three-character code contained within each message identifies its type. These are listed in the Message Type list, Appendix A.

The real-world event that initiates an exchange of messages is called a trigger event. See Section *2.3.1*, "*Trigger events*," for a more detailed description of trigger events. Refer to *HL7 Table 0003 – Event type* for a listing of all defined trigger events. These codes represent values such as **A patient is admitted** or **An order event occurred**. There is a one-to-many relationship between message types and trigger event codes. The same trigger event code may not be associated with more than one message type; however a message type may be associated with more than one trigger event code.

All message types and trigger event codes beginning with the letter "Z" are reserved for locally defined messages. No such codes will be defined within the HL7 Standard.

2.5.2 Segments and segment groups

A **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 given a name. For example, the ADT message may contain the following segments: Message Header (MSH), Event Type (EVN), Patient ID (PID), and Patient Visit (PV1).

Each segment is identified by a unique three-character code known as the Segment ID. Although the actual segments are defined in various chapters, the ID codes assigned to the various segments are listed in Appendix A.

Health Level Seven, Version 2.5.1 © 2007. All rights reserved.

All segment ID codes beginning with the letter **Z** are reserved for locally defined segments. No such codes will be defined within the HL7 Standard.

Two or more segments may be organized as a logical unit called a segment group. A segment group may be required or optional and might or might not repeat. As of v 2.5, the first segment in a newly defined segment group will be required to help ensure that unparsable messages will not be inadvertently defined.

A segment group is assigned a name that represents a permanent identifier that may not be changed.

A named segment X may occur more than once in an abstract message syntax. This differs from repetition described earlier in this section. When this occurs, the following rules must be adhered to:

If, within an abstract message syntax, a named segment X appears in two individual or group locations, and

a) Either appearance is optional or repeating in an individual location;

b) or, either appearance is optional or repeating, in a group location

then, the occurrences of segment X must be separated by at least one required segment of a different name so that no ambiguity can exist as to the individual or group location of any occurrence of segment X in a message instance.

Examples of proper segment grouping

Example 1	Example 2	Example 3

{ SEG 1}
SEG2
[SEG1]
	[SEG1]
	{
	SEG2
	[SEG1]
	}
		SEG1
		[SEG2]
		SEG3
		{ SEG1 }

Examples of unparsable segment grouping

Example 1	Example 2	Example 3	Example 4

{ SEG 1}
[SEG1]
	{ SEG1 }
	[SEG2]
	SEG1
		[SEG1]
		{
		[SEG2]
		SEG1
		SEG3
		}
			{ SEG1 }
			[SEG2
			SEG3]
			SEG1

In each of these examples it is not possible to tell which part of the message SEG1 belongs.

2.5.3 Fields

Definition: A field is a string of characters. Fields for use within HL7 segments are defined by HL7. A comprehensive data dictionary of all HL7 fields is provided in Appendix A.

HL7 does not care how systems actually store data within an application. When fields are transmitted, they are sent as character strings. Except where noted, HL7 data fields may take on the null value. Sending the null value, which is transmitted as two double quote marks (""), is different from omitting an optional data field. The difference appears when the contents of a message will be used to update a record in a database

rather than create a new one. If no value is sent, (i.e., it is omitted) the old value should remain unchanged. If the null value is sent, the old value should be changed to null. For further details, see Section *2.6,* "*Message construction rules*".

Version control rules regarding fields can be found in section *2.8*, "*Version compatibility definition*".

Local extension rules regarding fields can be found in section *2.11*, "*Local Extension*".

The various chapters of the Standard contain segment attribute tables. These tables list and describe the data fields in the segment and characteristics of their usage. In defining a segment, the following information is specified about each field:

2.5.3.1 Position (sequence within the segment)

Definition: Ordinal position of the data field within the segment. This number is used to refer to the data field in the text comments that follow the segment definition table.

In the segment attribute tables this information is provided in the column labeled **SEQ**.

2.5.3.2 Maximum length

Definition: Maximum number of characters that one occurrence of the data field may occupy.

In the segment attribute tables this information is in a column labeled **LEN.**

The maximum length is not of conceptual importance in the abstract message or the HL7 coding rules. The length of a field is normative. Changes to the field length may be negotiated by a site agreement such as a conformance profile. See section, *2.12, "Conformance Using Message Profiles"*. When this is done, it shall not render the implementation non-conformant. The receiver must be able to receive up to the maximum field length, and the sender can send up to the maximum field length.

Field length is determined based on the data type lengths, and should fall between the lower and the upper bounds for the corresponding data types. It is calculated to include the component and subcomponent separators. Because the maximum length is that of a single occurrence, the repetition separator is not included in calculating the maximum length See Section *2.5.3.5*, "*Repetition*".

The following conventions have been applied:

a) The maximum length of the data field shall be expressed as a number.

b) If the maximum length needs to convey the notion of a Very Large Number, the number 65536 should be displayed to alert the user. This convention takes the place of the practice in versions prior to 2.4 of abbreviating this expression as 64K.

c) If the maximum length cannot be definitively expressed because the data type for the field is variable, the symbolic number 99999 should be displayed. This convention takes the place of the practice in versions prior to 2.4 of displaying the notation “varies” or some other non-numeric description.

In v 2.5 maximum lengths are being assigned to data types. See *HL7 Table 0440 - Data Types*.

2.5.3.3 Data type

Definition: The basic building block used to construct or restrict the contents of a data field.

In the segment attribute tables this information is provided in the column labeled **DT**. If the data type of the field is variable, the notation “varies” will be displayed.

Health Level Seven, Version 2.5.1 © 2007. All rights reserved.

There are a number of data types defined by HL7. See section *2.16*, "*Data types*"

2.5.3.4 Optionality

Definition: Whether the field is required, optional, or conditional in a segment.

In the segment attribute tables this information is provided in the column labeled **OPT**.

The designations for optionality are:

- R - required
- O - optional
- C - conditional on the trigger event or on some other field(s). The field definitions following the segment attribute table should specify the algorithm that defines the conditionality for this field.
- X - not used with this trigger event
- B - left in for backward compatibility with previous versions of HL7. The field definitions following the segment attribute table should denote the optionality of the field for prior versions.
- W - withdrawn

Note: For Versions 2.3 and higher: the optionality of fields should be explicitly documented in the segment field definitions that follow each segment definition table; if the optionality of fields within a segment changes depending on the trigger event, that optionality should also be explicitly documented.

For version 2.5 and higher, the optionality, table references, and lengths of data type components are supplied in component tables of the data type definition. The component definitions that follow the component table will elaborate on the optionality and table references. Where needed, additional detailed field definitions will follow the formal segment attribute tables. (See also Sections *2.5.4*, "*Message delimiters*," *2.6*, *"Message construction rules*," *2.16*, "*Data types*").

2.5.3.5 Repetition

Definition: Whether the field may repeat. The value that appears in the repetitions column is the maximum number of allowed occurrences, e.g., a value of '3' would mean that the field can have '3 occurrences'; if unspecified, there is only one occurrence, i.e. cannot repeat.

In the segment attribute tables this information is provided in the column labeled **RP/#.**

The designations for Repetition are:

N or blank	-	no repetition
Y	-	the field may repeat an indefinite or site-determined number of times
(integer)	-	the field may repeat up to the number of times specified by the integer

Each occurrence may contain the number of characters specified by the field's maximum length. See Section *2.5.3.2*, "*Maximum length*".

Usage Note: For improved readability some technical committees opt to leave the Repetition fields blank to indicate that the field may NOT repeat. A blank may NOT be construed to mean that the field may optionally repeat.

As of v2.5 the Repetition column is to be left blank if the field may NOT repeat.

2.5.3.6 Table

Definition: The table attribute of the data field definition specifies the HL7 identifier for a set of coded values.

In the segment attribute tables, the table identifier is provided in the column labeled **TBL#.** If this attribute is not valued or blank, there is not a table of values defined for the field.

A number of conventions have been applied to this attribute of the data field definition.

a) If more than one table is applicable, the format xxxx/yyyy will be used to so designate multiple tables. Details on multiple tables will be specified in field or data type notes.

b) If the field is of data type ID or IS a table number will be allocated even if, in the case of IS, there may be a notation "No Suggested values".

c) If the field is of data type CE, CF, CNE or CWE and one or more externally or locally defined tables may be used, the symbolic number 9999 will appear in the column. This is to indicate that table values are used, but no HL7/User-defined table can be allocated. The narrative may constrain which external tables can be used.

d) Tables embedded in field components or subcomponents will not be cited in the attribute column.

   1) Tables embedded in data types are defined, save for the exceptions in point 2 below, in the data type's component table in section 2.A Data Types. They may, however, be constrained in the field note section. The field note definition supercedes the definition in the data type section.

   2) Tables embedded in fields with a data type of CE, CF, CNE, or CWE are only defined in the field notes section.

e) Values for HL7 tables shall not contain the embedded "suggested delimiters" delineated in section, *2.5.4*, "*Message delimiters*".

HL7 defines table values in 3 ways: HL7 defined, user-defined and externally defined.

User-defined Tables: A user-defined table is a set of values that are locally or site defined. This accommodates certain fields, like *PV1-3 - Assigned patient location*, that will have values that vary from institution to institution. Even though these tables are not defined in the Standard, they are given a user-defined table number to facilitate implementations. HL7 sometimes publishes suggested values that a site may use as a starter set (e.g., *table 0001- Sex*). The IS data type is often used to encode values for these tables. Note that some of these tables (e.g., *table 0302 - Point of care*) may reference common master files.

There are some user-defined tables that contain values that might be standardized across institutions but for which no applicable official standard exists. For these a set of **suggested** values may be listed in Appendix A. These suggested values appear in the text in a standard box format (e.g., HL7 *Table 0062 - Event Reason* in Section 3.4.1.4, “Event reason code”). It is recommended that these values be used where applicable within an institution and serve as a basis for extensions as required. These values may, however, be redefined locally. The appropriate functional committee within HL7 solicits suggestions for additional values from institutions that are applying the Standard.

HL7 Tables: An HL7 table is a set of values defined and published by HL7. They are a part of the HL7 Standard because they affect the interpretation of the messages that contain them. These values may not be redefined locally; however, the table itself may be extended to accommodate locally defined values. This is particularly applicable in the case of *HL7 table 0003 – Event Type*. The ID data type is most often used to encode values for HL7 tables. The values are listed in Appendix A. These HL7 tables also appear in the text in a standard box format (e.g., *HL7 table 0003 Event Type*).

Health Level Seven, Version 2.5.1 © 2007. All rights reserved.

External Tables: An external table is a set of coded values defined and published by another standards organization. External tables are used to populate fields like *FT1-19-Diagnosis Code - FT1*. Another example, the encoding of clinical observations using LOINC codes. The CE CF, CNE and CWE data type are used to represent values for these fields.

External tables arise from applications where the concepts and possibly the codes are established by external agencies due to regulatory requirements or agreements between HL7 and other Standards Developing Organizations. They may be published by HL7 on behalf of other organizations. Their contents are not subject to approval by HL7 ballot. Such tables will be published with HL7 Standards. However, they may be updated more frequently than HL7 Standards.

An external table may be imported into the HL7 standard subject to specific license or copyright requirements of the supplier/author. In this case the table will be an HL7-defined table. HL7 users will need to abide by the licensing and copyright requirements of the source when applicable.

The data type for the field will be CWE if 1) other tables are allowed in the field **or** 2) the external table may be locally extended **or** 3) when the code may be replaced by local text.

The data type for the field will be CNE if 1) no other table is allowed in the field **and** 2) the external table may **not** be locally extended **and** 3) text may not replace the code. A CNE field must have an HL7 defined or external table associated with it. It must be specified in the standard.

Local Tables: A local table is a table with a non-HL7 assigned table identifier and which contains a set of locally or site defined values. It may be locally assigned to local fields in Z segments or to HL7 fields having a CWE data type.

2.5.3.7 ID number

Definition: a small integer that uniquely identifies the data item throughout the Standard**.** In the segment definition this information is provided in the column labeled **ITEM #.**

2.5.3.8 Name

Definition: Descriptive name for the data item. In the segment attribute tables this information is provided in the column labeled **ELEMENT NAME**.

When the same name is used in more than one segment, it must have the same data type and semantic meaning in each segment as well as the same ID number. To deal with any ambiguities arising from this convention, whenever a field is referenced herein, the segment name and position must always be included.

2.5.4 Message delimiters

In constructing a message, certain special characters are used. They are the segment terminator, the field separator, the component separator, subcomponent separator, repetition separator, and escape character. The segment terminator is always a carriage return (in ASCII, a hex 0D). The other delimiters are defined in the MSH segment, with the field delimiter in the 4th character position, and the other delimiters occurring as in the field called Encoding Characters, which is the first field after the segment ID. The delimiter values used in the MSH segment are the delimiter values used throughout the entire message. In the absence of other considerations, HL7 recommends the suggested values found in Figure 2-1 delimiter values.

At any given site, the subset of the possible delimiters may be limited by negotiations between applications. This implies that the receiving applications will use the agreed upon delimiters, as they appear in the Message Header segment (MSH), to parse the message.

Note: The binary representation of the delimiter characters will vary with the character set used in the message.

Figure 2-1. Delimiter values

Delimiter	Suggested Value	Encoding Character Position	Usage
Segment Terminator	<cr>	-	Terminates a segment record. This value cannot be changed by implementers.
Field Separator	|	-	Separates two adjacent data fields within a segment. It also separates the segment ID from the first data field in each segment.
Component Separator	^	1	Separates adjacent components of data fields where allowed.
Subcomponent Separator	&	4	Separates adjacent subcomponents of data fields where allowed. If there are no subcomponents, this character may be omitted.
Repetition Separator	~	2	Separates multiple occurrences of a field where allowed.
Escape Character	\	3	Escape character for use with any field represented by an ST, TX or FT data type, or for use with the data (fourth) component of the ED data type. If no escape characters are used in a message, this character may be omitted. However, it must be present if subcomponents are used in the message.

2.6 MESSAGE CONSTRUCTION RULES

Note: These message construction rules define the standard HL7 encoding rules, creating variable length delimited messages. Although only one set of encoding rules has been defined as a standard since HL7 Version 2.3, other encoding rules are possible (but since they are non-standard, they may only be used by a site-specific agreement).

2.6.1 Rules for the sender

2.6.1.1 Message Construction Pseudocode

```
procedure transmit_message ( data ) {
  identify_message_needed;
  validate( data );
  order_segments( data, segment_list );

  foreach segment in ( segment_list ) {

    print segment.name;            /* e.g., MSH */

    /* gather all data for fields */
    foreach field in ( fields_of( segment ) ) {
      print field separator;       /* e.g., |  */
```


Health Level Seven, Version 2.5.1 © 2007. All rights reserved.

A message constituent, except as noted in points c, d and e below, will be withdrawn and removed, no sooner than, after 2 versions in a deprecated state. For example, if a message was originally deprecated in v 2.2, its definition can be removed when v 2.5 is published.

1) A message type and its definition may be removed.

2) A trigger event and its definition may be removed.

3) A segment group in an existing message may be removed.

4) A segment in an existing message may be removed.

A deprecated field in an existing segment may **NOT** be removed from the standard. However, no sooner than, after 2 versions in a deprecated state, the field will be marked as withdrawn and all explanatory narrative will be removed

A deprecated component in an existing data type may **NOT** be removed from the standard. However, no sooner than, after 2 versions in a deprecated state, the component will be marked as withdrawn and all explanatory narrative will be removed.

A deprecated member of an existing HL7 table may **NOT** be removed from the standard. However, no sooner than, after 2 versions in a deprecated state, the table member will be marked as withdrawn and all explanatory narrative will be removed from the description and comment column.

2.8.5 Early adoption of HL7 changes

Early adoption of HL7 changes that have been approved by the technical committee for the next membership ballot is a common practice and is not prohibited, but carries risk. Such changes may be rejected or modified in the balloting process. One example is that the change may pass but may be positioned differently in the segment or data type.

2.8.6 Technical correction rules

Technical corrections may be applied between versions on a case-by-case basis. These corrections will be published on the HL7 website. The following meet criteria for technical correction:

Spelling correction

Incorrect section reference

Transcription error in an imported external table

Correction of an inconsistency between a segment attribute table and the field narrative

Erroneous examples

Erroneous/misleading descriptions

2.9 MESSAGE PROCESSING RULES

The processing rules described here apply to all exchanges of messages, whether or not the HL7 encoding rules or Lower Layer Protocols are used. They represent the primary message processing mode. The user may use either the original processing rules, described in section *2.9.2* or the enhanced processing rules, described in section *2.9.3*.

Note: The MCF – Delayed Acknowledgement message has been removed from the standard. It was deprecated in v 2.2. Accordingly, the narrative notes regarding deferred processing have been removed from this section.

Health Level Seven, Version 2.5.1 © 2007. All rights reserved.

Certain variants exist and are documented elsewhere:

a) an optional sequence number protocol. Refer to section 2.10.1.

b) an optional protocol for continuing a very long message. Refer to section 2.10.2.

Because the protocol describes an exchange of messages, it is described in terms of two entities, the initiating and responding systems. Each is both a sender and receiver of messages. The initiating system sends first and then receives, while the responding system receives and then sends.

In overview this exchange proceeds as follows:

Message Exchange

Step	Process	Comment
Step 1	Initiator constructs an HL7 message from application data and sends it to the responding system	
Step 2	Responder receives message and processes it based on rules	The rules differ based on whether the original acknowledge mode or the enhanced acknowledgement mode is followed
Step 3	Responder sends response message	
Step 4	Initiator processes response message	

2.9.1 Message initiation

The initiating application creates a message with data values as defined in the appropriate chapter of this Standard. The fields shown below should be valued in the MSH segment (as defined under the MSH segment definition of this chapter). The message is encoded according to the applicable rules and sent to the lower level protocols, which will attempt to deliver it to the responding application. (For definitions of the MSH fields see Section *2.15.9*, "*MSH - message header segment*")

Field	Notes
MSH-3-sending application	
MSH-4-sending facility	
MSH-5-receiving application	
MSH-6-receiving facility	
MSH-7-date/time of message	
MSH-9-message type	
MSH-10-message control ID	Unique identifier used to relate the response to the initial message.
MSH-11-processing ID	
MSH-12-version ID	
MSH-13-sequence number	
MSH-14-continuation pointer	Used in implementation of message continuation protocol. See Section *2.10.2*, "*Continuation messages and segments*". Also see chapter 5.

Certain other fields in the MSH segment are required for the operation of the HL7 encoding rules; they will not be relevant if other encoding rules are employed.

The event code in the second component of *MSH-9-message type* is redundantly shown elsewhere in some messages. For example, the same information is in the EVN segment of the ADT message. This is for compatibility with prior versions of the HL7 protocol. Newly defined messages should only show the event code in *MSH-9-message type*.

2.9.2 Message response using the original processing rules

2.9.2.1 Accept and validate the message in responding system

Upon receipt of the message, when the Original Acknowledgement rules are used, the protocol software in the responding system validates it against at least the following criteria:

Note: Both MSH-15-accept acknowledgment type and MSH-16-application acknowledgment type are null or not present.

a) the value in MSH-9-message type is one that is acceptable to the receiver.

b) the value in MSH-12-version ID is acceptable to the receiver.

c) the value in MSH-11-processing ID is appropriate for the application process handling the message.

If any of these edits fail, the protocol software rejects the message. That is, it creates an ACK message with AR in MSA-1-acknowledgment code.

If successful, the process moves to the next step.

2.9.2.2 Accept and validate/process the message in the receiving application

Upon successful validation by the responding system, the message is passed to the receiving application, which performs one of these functions:

a) process the message successfully, generating the functional response message with a value of AA in MSA-1-acknowledgment code.

b) send an error response, providing error information in functional segments to be included in the response message with a value of AE in MSA-1-acknowledgment code.

c) fail to process (reject) the message for reasons unrelated to its content or format (system down, internal error, etc.). For most such problems it is likely that the responding system will be able to accept the same message at a later time. The implementers must decide on an application-specific basis whether the message should be automatically sent again. The response message contains a value of **AR** in *MSA-1-acknowledgment code*.

The MSH segment in the response is constructed anew following the rules used to create the initial message described above. In particular, *MSH-7-date/time of message* and *MSH-10-message control ID* refer to the response message; they are not echoes of the fields in the initial message. *MSH-5-receiving application*, *MSH-6-receiving facility*, and *MSH-11-processing ID* contain codes that are copied from *MSH-3-sending application*, *MSH-4-sending facility* and *MSH-11-processing ID* in the initiating message.

In all the responses described above, the following values are put in the MSA segment. Note that the field definitions for the MSA segment fields are in Section *2.15.8*.

Field	Notes
MSA-1-acknowledgment code	As described above.
MSA-2-message control ID	MSH-10-message control ID from MSH segment of incoming message.
MSA-4-expected sequence number	As described in Section *2.10.1*, "*Sequence number*

Health Level Seven, Version 2.5.1 © 2007. All rights reserved.

	protocol," (if the sequence number protocol is being used).
ERR segment fields	Refer to section 2.15.5.

The receiving application then passes the response message back to the responding system for the next step in the process.

2.9.2.3 Transmit the response message

Upon receiving the response message from the receiving application, the responding system transmits it to the initiating system.

The initiator processes the response message.

2.9.3 Response using enhanced acknowledgement

a) the responding system receives the message and commits it to safe storage. This means that the responding system accepts the responsibility for the message in a manner that releases the sending system from any obligation to resend the message. The responding system now checks the message header record to determine whether or not the initiating system requires an accept acknowledgment message indicating successful receipt and secure storage of the message. If it does, the accept acknowledgment message is constructed and returned to the initiator.

b) at this point, the requirements of the applications involved in the interface determine whether or not more information needs to be exchanged. This exchange is referred to as an application acknowledgment and includes information ranging from simple validation to a complex application-dependent response. If the receiving system is expected to return application-dependent information, it initiates another exchange when this information is available. This time, the roles of initiator and responder are reversed.

2.9.3.1 Accept and validate the message in responding system

Upon receipt of the message, when the Enhanced Acknowledgement rules are used, the protocol software in the responding system makes an initial determination as to whether or not the message can be accepted, based on factors such as:

Note: At least one of MSH-15-accept acknowledgment type or MSH-16-application acknowledgment type is not null.

a) the status of the interface

b) the availability of safe storage onto which the message can be saved

c) the syntactical correctness of the message, if the design of the receiving system includes this type of validation at this phase

d) the values of MSH-9-message type, MSH-12-version ID, and MSH-11-processing ID, if the design of the receiving system includes this type of validation at this phase

It then examines the Message Header segment (MSH) to determine whether or not the initiating system requires an accept acknowledgment.

2.9.3.2 Transmit general acknowledgement message

A general acknowledgement message is not always required by the initiating system, but if it is the responding system sends one of the following:

a) a commit accept (CA) in MSA-1-acknowledgment code if the message can be accepted for processing

b) a commit reject (CR) in MSA-1-acknowledgment code if the one of the values of MSH-9-message type, MSH-12-version ID or MSH-11-processing ID is not acceptable to the receiving application

c) a commit error (CE) in MSA-1-acknowledgment code if the message cannot be accepted for any other reason (e.g., sequence number error)

The MSH segment in the response is constructed anew following the rules used to create the initial message described above. In particular, *MSH-7-date/time of message* and *MSH-10-message control ID* refer to the response message; they are not echoes of the fields in the initial message. *MSH-5-receiving application*, *MSH-6-receiving facility*, and *MSH-11-processing ID* contain codes that are copied from *MSH-3-sending application*, *MSH-4-sending facility* and *MSH-11-processing ID* in the initiating message.

For this response, the following values are put in the MSA segment. Note that the field definitions for the MSA segment fields are in Section *2.15.8*, '*MSA - message acknowledgment segment*":

Field	Notes
MSA-2-message control ID	MSH-10-message control ID from the incoming message.
MSA-1-acknowledgment code	As described above.
MSA-4-expected sequence number	As described in Section *2.10.1*, "*Sequence number protocol*" (if the sequence number protocol is being used).
ERR segment fields	Refer to section 2.15.5.

Note: MSH-15-accept acknowledgment type and MSH-16-application acknowledgment type are not valued (not present or null). At this point, the accept portion of this message exchange is considered complete.

2.9.3.3 Transmit application acknowledgement

If the message header segment indicates that the initiating system also requires an application acknowledgment, this will be returned as the initial message of a later exchange.

For this message, the receiving system acts as the initiator. Since the message it sends is application-specific, the layouts of these application-level response messages are defined in the relevant application-specific chapter. If needed, this application acknowledgment message can itself require (in *MSH-15-accept acknowledgment type*) an accept acknowledgment message (MSA). *MSH-16-application acknowledgment type*, however, is always null, since the protocol does not allow the application acknowledgment message to have an application acknowledgment.

For this response, the following values are put in the MSA segment. Note that the field definitions for the MSA segment fields are in Section *2.15.8*, "*MSA - message acknowledgment segment*".

Field	Notes
MSA-2-message control ID	Identifies the initial message from the original initiating system as defined in Section *2.9.1*, "*Message initiation*".
MSA-1-acknowledgment code	Uses the application (processing) acknowledgment codes as described in Section *2.15.8.1*.
MSA-3-text message	Text description of error.
ERR-1-Error Code and Location	As described in section *2.15.5.1*. Populated if an error condition is found.

At this point, the application acknowledgment portion of this message exchange is considered complete.

Health Level Seven, Version 2.5.1 © 2007. All rights reserved.

If the processing on the receiving system goes through multiple stages, chapter-defined messages may be used to relay status or informational changes to other systems (including the original initiating system). Such messages are not part of the acknowledgment scheme for the original message, but are considered to be independent messages triggered by events on the (original) responding system.

Note: The original acknowledgment protocol is equivalent to the enhanced acknowledgment protocol with MSH-15-accept acknowledgment type = NE and MSH-16-application acknowledgment type = AL, and with the application acknowledgment message defined so that it never requires an accept acknowledgment (MSH-15-accept acknowledgment type = NE).

2.10 SPECIAL HL7 PROTOCOLS

This section contains several extensions to the basic HL7 message protocol. These extensions represent implementation choices, and are to be used on a site-specific and application-specific basis as needed.

2.10.1 Sequence number protocol

For certain types of data transactions between systems the issue of keeping databases synchronized is critical. An example is an ancillary system such as lab, which needs to know the locations of all inpatients to route stat results correctly. If the lab receives an ADT transaction out of sequence, the census/location information may be incorrect. Although it is true that a simple one-to-one acknowledgment scheme can prevent out-of-sequence transactions between any two systems, only the use of sequence numbers can prevent duplicate transactions.

Note: Although this sequence number protocol is limited to the use of sequence numbers on a single transaction stream between two applications, this sequencing protocol is sufficiently robust to allow the design of HL7-compatible store-and-forward applications.

a) initial conditions:

   1) the system receiving the data stream is expected to store the sequence number of the most recently accepted transaction in a secure fashion before acknowledging that transaction. This stored sequence number allows comparison with the next transaction's sequence number, and the implementation of fault-tolerant restart capabilities.

   2) the initiating system keeps a queue of outgoing transactions indexed by the sequence number. The length of this queue must be negotiated as part of the design process for a given link. The minimum length for this queue is one.

   3) the sequence number is a positive (non-zero) integer; and it is incremented by one (by the initiating system) for each successive transaction.

b) starting the link:

   1) the value of 0 (zero) for a sequence number is reserved: it is allowed only when the initiating system (re-)starts the link.

   2) if the receiving system gets a transaction with a 0 (zero) in the sequence number field, it should respond with a general acknowledgment message whose MSA contains a sequence number one greater than the sequence number of the last transaction it accepted in the Expected Sequence Number field. If this value does not exist (as on the first startup of a given link), the MSA should contain a sequence number of -1, meaning that the receiving system will use the positive, non-zero sequence number of the first transaction it accepts as its initial sequence number (see resynching the link, item e below).

   3) the initiating system then sends the transaction indexed by the expected sequence number (if that expected transaction is still on its queue). Otherwise the link is frozen until an operator intervenes.

2.10.4.2 Action code/unique identifier mode update definition

In the "action code/unique identifier" mode, each member of a repeating group of segments must have a unique identifier (equivalent to the filler number in observational reports messages). The choice of delete/update/insert is determined by the action code (equivalent to the result status in observational reports messages). Refer to *HL7 Table 0206 - Segment action code* for valid values.

HL7 Table 0206 - Segment action code

Value	Description	Comment
A	Add/Insert	
D	Delete	
U	Update	

The *unique identifier* is defined in a general manner as follows: it uniquely identifies one of multiple repetitions of the primary entity defined by the repeating segment in a way that does not change over time. It is not dependent on any particular message identifier level (MSH) fields; it functions across messages, not just within a message. The *unique identifier* will be chosen on a segment-specific basis, depending on the primary entity referenced by the segment. For some cases, such as a diagnosis code, it may be a CE data type. For others, such as a person identifier, it may be a CX data type. For others it may be an EI (entity identifier) data type.

Note: This mode is available for use only for new segments for Version 2.3 and for new segments in future versions

2.11 LOCAL EXTENSION

The following section specifies where local extensions to a message and its constituent parts are allowed, where they are not, and where they are ill-advised. Inter-version compatibility rules must be followed plus there are certain restrictions and prohibitions outlined in the sections that follow. In general, basic structures should not be altered.

The reader is advised to review the Conformance mechanism defined in section *2.12*, "*Conformance Using Message Profiles*" before applying local extensions. Using the conformance mechanism may eliminate the need for local extension.

2.11.1 Messages

Messages may be locally extended as follows:

a) Users may develop local Z messages to cover areas not already covered by existing HL7 messages. These should be composed of HL7 segments where possible.

b) A local Z message may consist entirely of Z segments except that it must begin with a MSH segment.

c) A local Z Acknowledgement message must begin with an MSH segment followed by an MSA segment, an optional SFT segment and a conditional ERR segment.

d) Users may develop Z segments and add them to Z messages.

e) Users may develop Z segments and add them to HL7 messages. The trigger event may remain the same if the intent of the message has remained unchanged.

f) The practice of adding additional HL7 segments, like NTE, to existing HL7 messages locally is ill-advised. HL7 may move or change the segment in a future release; this will render the message unparsible.

2.11.2 Trigger events

Users may develop local Z trigger events for messages.

2.11.3 Segment groups

a) The practice of turning a single segment or segments into a segment group locally **is not** allowed within an HL7 event. It will have a negative impact on XML and any component-based encoding schemes. Note that HL7, on other hand, can do this.

b) A segment group may not be ungrouped locally.

Fox example, if there is an HL7 group as follows:

{

ABC

[DEF

[GHI]]

}

one cannot change it in a local implementation to be as follows:

{[ABC]}

[DEF]

[GHI]

Example 2:

If the original definition was:

GROUP1 ::= ABC, GROUP2?

GROUP2 ::= DEF, GHI?

and someone wished to constrain the segments in GROUP2 to be mandatory

(i.e. the HL7 grammar would look like:

{[

ABC

DEF

[GHI]

]}

Health Level Seven, Version 2.5.1 © 2007. All rights reserved.

Their message instance would need to still look like:

<GROUP1>

<ABC/>

<GROUP2>

<DEF/>

<GHI/>

</GROUP2>

</GROUP1>

It would be an error if they instead sent it as:

<GROUP1>

<ABC/>

<DEF/>

<GHI/>

</GROUP1>

c) A segment group can repeat locally. The 1st repetition needs to mean what it does in HL7

d) The practice of incorporating a Z segment into a segment group locally **is** allowed.

2.11.4 Segments

2.11.4.1 Local extension rules for segments

Users may not modify an existing segment, except as specified in section *2.8.2*.

Locally defined fields may be defined for use in locally defined segments, although HL7 defined fields are a better choice when available. The practice of extending an HL7 segment with locally defined fields, while not prohibited, is ill-advised.

HL7 also recognizes that sites may have locally defined fields where the users believe the enhancement may be of interest to the HL7 community as a whole and are moving forward with a proposal to HL7.

2.11.4.2 Caveats for locally extending segments

Locally extending an HL7 segment with locally defined fields will likely cause conformance problems with the next release of the HL7 standard. There are, however, certain circumstances where HL7 has, itself, directed the membership to add Z fields as an interim measure between versions to accommodate

regulatory agency requirements. These are fields that HL7 has reserved for official introduction in the next release.

If the local site intends to add a proposed field early, there is a risk that it may collide with another field when HL7 officially approves or rejects the proposed additions. Some sites have employed the practice of assigning a high sequence number locally i.e. leaving a gap between the last official HL7 field and the proposed new field. The user–defined fields should be deleted or deprecated when HL7 officially approves or rejects the proposed additions so that the fields do not collide. It must be understood that the local implementation will have to adjust if a collision occurs and they want to conform.

2.11.5 Data types

The following rules apply for locally extending data types:

a) Locally defined data types may be defined for use in locally defined segment fields, although HL7 defined data types are a better choice when available.

b) Locally redefining existing data type components, e.g., changing a component from NM to ST, is prohibited.

c) Data types may be locally extended by adding new components at the end. This action creates a Z data type.

Note: The practice of extending an HL7 data type with locally defined components is particularly ill-advised and may cause conformance problems with the next release of the HL7 standard.

2.11.6 Tables

Rules for locally extending tables are the same as discussed in section *2.5.3.6*, "*Table*":

a) Users may redefine suggested values in User-defined tables.

b) Local tables may be defined for Z fields.

c) Local tables may be assigned to HL7 fields with data type CWE.

2.12 CONFORMANCE USING MESSAGE PROFILES

Previous sections in this chapter define the rules and conventions for constructing and communicating a message including the parts of a message structure. Messages that adhere to those rules of a specific version of a standard are **compliant** to that version of the standard.

Compliance to the HL7 Standard has historically been impossible to define and measure in a meaningful way. To compensate for this shortcoming, vendors and sites have used various methods of specifying boundary conditions such as optionality and cardinality. Frequently, specifications have given little guidance beyond the often-indefinite constraints provided in the HL7 Standard.

This section presents the methodology for producing a precise and unambiguous specification called a ***message profile***. Messages that adhere to the constraints of a message profile are said to be **conformant** to the profile. For conformance to be measurable, the message profile must specify the following types of information:

- What data will be passed in a message.
- The format in which the data will be passed.
- The acknowledgement responsibilities of the sender and receiver.

Health Level Seven, Version 2.5.1 © 2007. All rights reserved.

2.13 CHAPTER FORMATS FOR DEFINING HL7 MESSAGES

Subsequent chapters of this document describe messages that are exchanged among applications in functionally-specific situations. Each chapter is organized as follows:

a) purpose. This is an overview describing the purpose of the chapter, general information and concepts.

b) trigger events and messages. There is a list of the trigger events.

c) message segments. The segments defined in a chapter are then listed in a functional order designed to maximize conceptual clarity.

d) examples. Complete messages are included.

e) implementation considerations. Special supplementary information is presented here. This includes issues that must be addressed in planning an implementation.

f) outstanding issues. Issues still under consideration or requiring consideration are listed here.

2.13.1 Message representation

For each trigger event the messages that are exchanged when the trigger event occurs are defined using the HL7 abstract message syntax as follows:

Each message is defined in special notation that lists the segment IDs in the order they would appear in the message. Braces, { . . . }, indicate one or more repetitions of the enclosed group of segments. Of course, the group may contain only a single segment. Brackets, [. . .], show that the enclosed group of segments is optional. If a group of segments is optional and may repeat it should be enclosed in brackets and braces, [{...}].

Note: [{...}] and {[...]} are equivalent.

Whenever braces or brackets enclose more than one segment ID a special stylistic convention is used to help the reader understand the hierarchy of repetition. For example, the first segment ID appears on the same line as the brace, two columns to the right. The subsequent segment IDs appear under the first. The closing brace appears on a line of its own in the same column as the opening brace. This convention is an optional convenience to the user. If there is conflict between its use and the braces that appear in a message schematic, the braces define the actual grouping of segments that is permitted.

A choice of one segment from a group of segments is indicated by using angle brackets to delimit the group and vertical bar delimiters between the several segments.

Example: The ORM^O01, as described in chapter 4 section 4.4.1, allows a choice of order detail segments. The choice would be represented as follows:

```
<OBR|RQD|RQ1|RXO|ODS|ODT>
```

Consider the hypothetical triggering event **a widget report is requested**. It might be served by the Widget Request (WRQ) and Widget Report (WRP) messages. These would be defined in the Widget chapter (say Chapter XX). The Widget Request message might consist of the following segments: Message Header (MSH), Software Segment (SFT) and Widget ID (WID). The Widget Report message might consist of the following segments: Message Header (MSH), Software Segment (SFT), Message acknowledgment (MSA),

The Australian Healthcare Messaging Laboratory (AHML) is a not-for-profit venture of the University of Ballarat providing a developer test-bed available to HL7 members as well as validation of the static profile. Contact AHML (http://www.ahml.com.au.) for more information.

Error Segment (ERR) and one or more Widget Description (WDN) Segments each of which is followed by a single Widget Portion segment (WPN) followed by zero or more Widget Portion Detail (WPD) segments.

2.13.2 HL7 abstract message syntax example

The schematic form for this hypothetical exchange of messages is shown in Figure 2-5:

Figure 2-5. Hypothetical schematic message

Trigger Event: WIDGET REPORT IS REQUESTED

WRQ	Widget Request	Status	Chapter
MSH	Message Header		2
[{SFT}]	Software Segment		2
WID	Widget ID		XX

WRP	Widget Report	Status	Chapter
MSH	Message Header		2
[{SFT}]	Software Segment		2
MSA	Message Acknowledgment		2
[{ERR}]	Error Segment		2
{	---Widget begin		
WDN	Widget Description		XX
WPN	Widget Portion		XX
}	---Widget end		

The WID, WDN, WPN, and WPD segments would be defined by the widget committee in the widget chapter, as designated by the Arabic numeral XX in the right column. The MSH and MSA segments, although included in the widget messages, are defined in another chapter. They are incorporated by reference into the widget chapter by the chapter number XX.

On the other hand, the widget committee might decide that the WPN and WPD segments should appear in pairs, but the pairs are optional and can repeat. Then the schematic for the WRP message would be as shown in Figure 2-6.

Figure 2-6. WPN and WPD segments in pairs

WRF	Widget Report	Status	Chapter
MSH	Message Header		2
MSA	Message Acknowledgment		2
{	--Widget begin		
WDN	Widget Description		XX
[{	---WidgetDetailA begin		
WPN	Widget Portion		XX
WPD	Widget Portion Detail		XX
}]	---WidgetDetailA end		
}	---Widget end		

Health Level Seven, Version 2.5.1 © 2007. All rights reserved.

If the widget committee determined that at least one pair of WPN and WPD segments must follow a WDN, then the notation would be as shown in Figure 2-7.

Figure 2-7. At least one pair of WPN and WPD

WRP	Widget Report	Group Name	Status	Chapter
MSH	Message Header			2
MSA	Message Acknowledgment			2
{	--Widget begin			
WDN	Widget Description			XX
{	---WidgetDetailB begin			
WPN	Widget Portion			XX
WPD	Widget Portion Detail			XX
}	---WidgetDetailB begin			
}	---Widget end			

2.14 ACKNOWLEDGMENT MESSAGES

Acknowledgment messages may be defined on an application basis. However the simple general acknowledgment message (ACK) may be used where the application does not define a special message (application level acknowledgment) and in other cases as described in Section *2.9*, "*Message Processing Rules*".

2.14.1 ACK - general acknowledgment

The simple general acknowledgment (ACK) can be used where the application does not define a special application level acknowledgment message or where there has been an error that precludes application processing. It is also used for accept level acknowledgments. The details are described in Section *2.9*, "*Message Processing Rules*".

ACK^varies^ACK	General Acknowledgment	Status	Chapter
MSH	Message Header		2
[{ SFT }]	Software segment		2
MSA	Message Acknowledgment		2
[{ ERR }]	Error		2

Note: For the general acknowledgment (ACK) message, the value of MSH-9-2-Trigger event is equal to the value of MSH-9-2-Trigger event in the message being acknowledged. The value of MSH-9-3-Message structure for the general acknowledgment message is always ACK.

2.14.2 MCF - delayed acknowledgment

Note: The MCF message was deprecated in v2.2 and has been withdrawn and removed from the standard as of v 2.5.

2.15 MESSAGE CONTROL SEGMENTS

The following segments are necessary to support the functionality described in this chapter.