<hapi/>

HAPI Conformance Classes System

User Manual

By Wendy Hall, Technical Writer,

and

James Agnew, Paul Brohman, Mitch Delachevrotiere,
Shawn Dyck and Cory Metcalf,

Software Developers

First Edition: Published April 2003

Table of Contents

Getting Started

Using the System

Reference Material

This document is available in Microsoft Word, PDF and HTML formats from the HAPI Web site at http://hl7api.sourceforge.net.

List of Tables

List of Figures

<hapi/>

HAPI Conformance Classes System

User Manual

Getting Started

HAPI Conformance Classes System

User Manual

Introduction

> "In order to reduce cost and improve the quality of health care, computer-based information systems are being used more and more in health care. However, there is no comprehensive information system that can completely cover all the needs of hospitals and other areas of the health care system. Therefore, it will be necessary to employ information systems from various suppliers. These systems must be functionally coupled with each other for cooperative data processing: communication must be possible."

--Kai U. Heitmann, Bernd Blobel and Joachim Dudeck
HL7: Communication Standard in Medicine

Health Level 7

Health Level 7 (HL7) is an organization that develops standards in the health care field, particularly electronic messaging standards. Accredited by the American National Standards Institute (ANSI), HL7 is coordinating an international project aimed at facilitating electronic data exchange within and among hospitals. The goal is to design standards that will allow different clinical information systems to communicate with each other. With these standards, the communication could occur within a single hospital or between hospitals, whether they are in the same city or on opposite sides of the globe.

For example, an electronic thermometer could take a patient's temperature. Using the HL7 standards, that information could be sent from the thermometer to a computer anywhere in the world, and that computer would understand. Without HL7, the computers and devices may not be able to communicate.

For more information on HL7, visit www.hl7.org.

Health Level 7 Application Programming Interface

There are several commercial and open-source programming libraries that implement the HL7 standards.

HAPI (HL7 application programming interface) at http://hl7api.sourceforge.net is an open-source HL7 library for Java. Its major features include:

- Message parsing and encoding functionality for both standard HL7 encodings (ER7 and XML).
- A library of 2.x message classes.
- Various message transport and handling functions.
- Various forms of message validation, including enforcement of HL7 data type rules, and checking messages against conformance profiles (see below).

Conformance

The HL7 standard defines hundreds of message formats, each of which is used to communicate information about a few specific types of events that occur in healthcare (for example, patient admission to hospital or a lab result recorded by a technician). HL7 defines the structure of each message (including segments, fields, data types, maximum lengths) and the meaning of each part of the structure. Message structures, and parts thereof, are re-used in many different situations in slightly different ways. This means that the specifications for these are necessarily “loose” (many fields are optional, and field lengths are often longer than required) to accommodate a variety of uses. Unfortunately this optionality leaves much room for interpretation when implementing an HL7 interface between two systems, making this process difficult, costly, and error-prone.

HL7 introduced a conformance framework in HL7 2.5. This framework presents a method for users of the standard to write a more detailed, less loose message specification, called a conformance profile. A profile must be based on one of the standard HL7 message specifications, but it is much more specific, in that it tightly constrains the optionality of fields, length of data, constant values, and number of repetitions of segments in messages. Profiles are written in XML. This feature enables the comparison of multiple profiles (for example, to find a common profile that satisfies two systems) and the automatic validation of a message against a profile (as opposed to the manual checking against paper specifications that was common in the past). HAPI provides a tool for message validation against profiles.

This manual describes an advanced HAPI tool that compiles a profile into Java code. A programmer can write an HL7 application against the resulting code, and use it to create HL7 messages. The generated code ensures that the message conforms to the original profile, through compile-time, type-checking and runtime Java exceptions.

Message Profiles

In HL7, a message profile is a refinement of a standard HL7 message structure. It is an XML document that defines additional constraints beyond the standard. Such additional constraints are often needed to communicate with a specific clinical information system or product line. The message profile does not contain any clinical data; if it did, it would be called a message and not a message profile.

HL7 message profiles have a hierarchical format. At the top of the hierarchy is information about message structure. A message can have one or more segments or segment groups. Each segment can have one or more fields. Each field can have one or more components. Each component may or may not have sub-components.

System Description

HAPI Conformance Classes System has three parts: the Messaging Workbench, the generator system and the compiled profile Java archive. The Messaging Workbench is freeware that creates, maintains and displays message profiles. The generator system is Java software that generates standardized messages in Java from XML code (which comes from another part of HAPI). The compiled profile Java archive is software that allows a programmer to access the messages that the generator system produced. The messages can then be used as part of a program that the programmer writes.

System Requirements

Operating System

HAPI Conformance Classes System operates on any operating system that supports the Java 1.4.1 platform, including Windows, Macintosh and UNIX platforms. To find out whether your operating system supports the required Java platform, see http://java.sun.com, or search the Web for Java 2 standard edition (J2SE) for your platform.

Hardware

HAPI Conformance Classes System operates on any hardware that supports the Java platform, including personal computers, Macintosh and UNIX-based systems.

Software

HAPI Conformance Classes System requires:

- A computer with Internet access
- A Java compiler and (optionally) a Java Integrated Development Environment (IDE)
- All of HAPI, including HAPI Conformance Classes System, which can be downloaded from HAPI at http://hl7api.sourceforge.net
- Apache Ant, a Java-based build tool used to compile classes and build package files. This is available from Apache at http://ant.apache.org
- Messaging Workbench—freeware that creates, maintains and displays electronic message profiles. This software is available from HL7 at http://hl7.org.
- Xerces, an XML parser, available at http://xml.apache.org

Installation

Perform the following steps to install the HAPI Conformance Classes System.

1. Download the archive containing the HAPI Conformance Classes System from the HAPI Web site at http://hl7api.sourceforge.net.

2. Extract the contents of the archive to a local directory.

3. Download the remaining software as listed under System Requirements.

Verification and Validation

This section tells you how to verify that you have downloaded and installed the HAPI Conformance Classes System correctly.

How to Validate Installation

1. Open a command line system such as DOS or UNIX.

2. Run the test system by entering the following command:

   ```
   Java ConfGen -t
   ```

3. If all tests succeed, the screen will read as follows:

   ```
   ConfGen: system test enabled
   Checking for XML Parser:              PASS
   Checking for Xerces:                  PASS
   Checking for Conformance Classes:     PASS
   Checking for Apache Ant:              PASS
   ```

   Installation is correct. You can now use the HAPI Conformance Classes System.

4. If a test fails, the screen will read FAIL. Installation is incorrect. Proceed to Troubleshooting on page 25.

HAPI Conformance Classes System

User Manual

Using the System

Procedure 1

Creating a Message Profile with Messaging Workbench

This procedure introduces the process of creating message profiles. More detailed information is available from the Messaging Workbench help. Messaging Workbench is freeware that creates, maintains and displays message profiles. This software allows rapid prototyping of new profiles, and quick and easy alteration of existing profiles to meet new requirements. To download Messaging Workbench, see System Requirements.

How to create a message profile with the Messaging Workbench

1. Start Messaging Workbench.

2. Click File>Load Message Structure.

3. Pick a message structure from the list (see Figure 1) and click OK.

4. Select a structure from the Structure textbox (see Figure 2) and press Enter.

   A message tree appears in the bottom left corner of the Messaging Workbench.

   Note: For a description of the structures listed in the Structure textbox, see http://hl7.org/Library/Generat/v231.zip

5. To add constraints to each piece of the message tree, click on one of the nodes in the message tree (see Figure 1) and add constraints as follows:

   a. Select the Repeatable check box (see Figure 1).

   b. Set the usage list box to either Required or Not Supported (see Figure 1). For definitions of Required and Not Supported, see HL7 2.5 Section 2.12 or visit the bookstore at www.hl7.org.

   c. Change the maximum and minimum cardinality as desired (see Figure 1).

      Note: A maximum cardinality of zero means that there is no limit.

   d. Set the maximum length of the node(see Figure 1).

   e. Set an example value if desired (see Figure 1).

   f. Add other information if desired (see Figure 1)).

6. Generate the XML profile as follows:

   a. Click on the Display/Reports tab at the top of the dialog box (see Figure 3).

   b. Select the XML specification from the left drop-down menu (see Figure 3).

   An XML representation of the message tree appears.

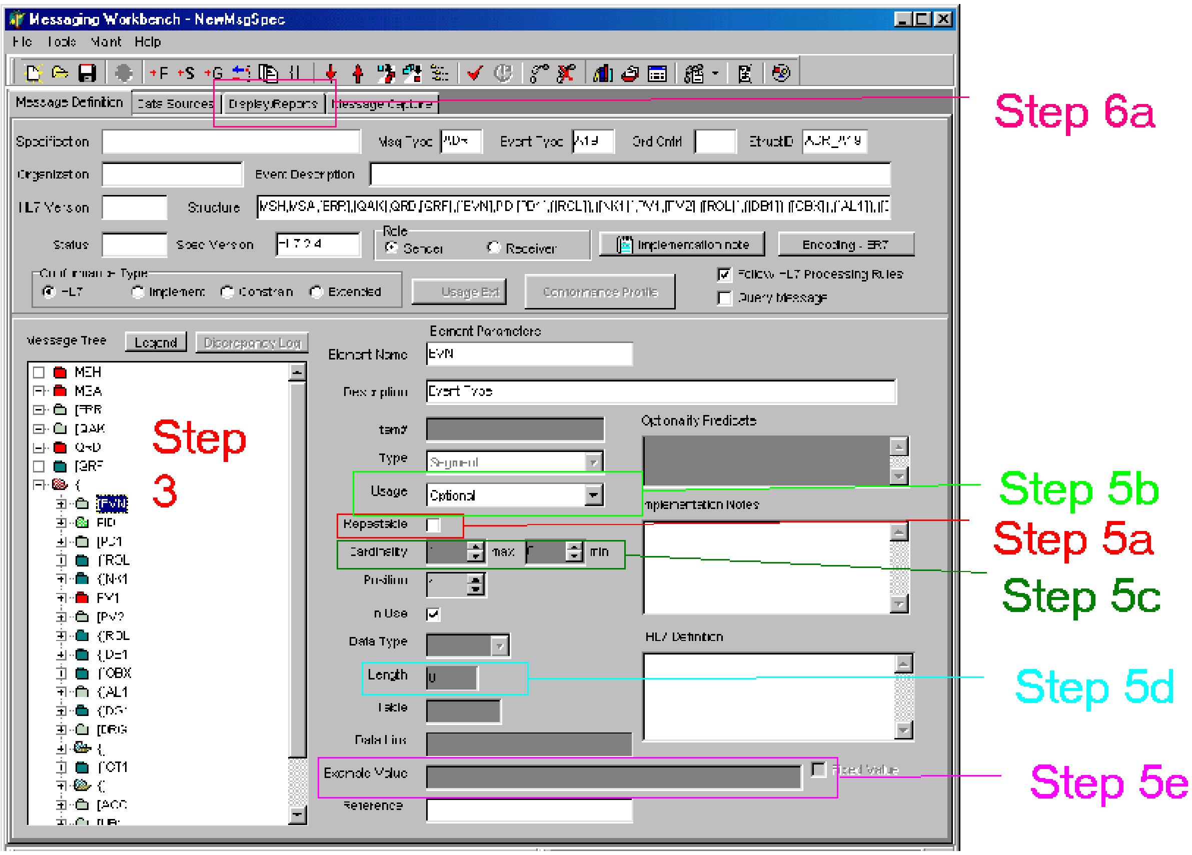
   c. Click on the Generate HL7 Registry Profile button on the top right corner of the Messaging Workbench (see Figure 3).

   A dialog box asks whether you want to register this profile with HL7.

   d. Since registering a profile is beyond the scope of this documentation, click No.

7. To save this profile, click File>Save.

   This profile is now ready for the generator system (see Procedure 2).



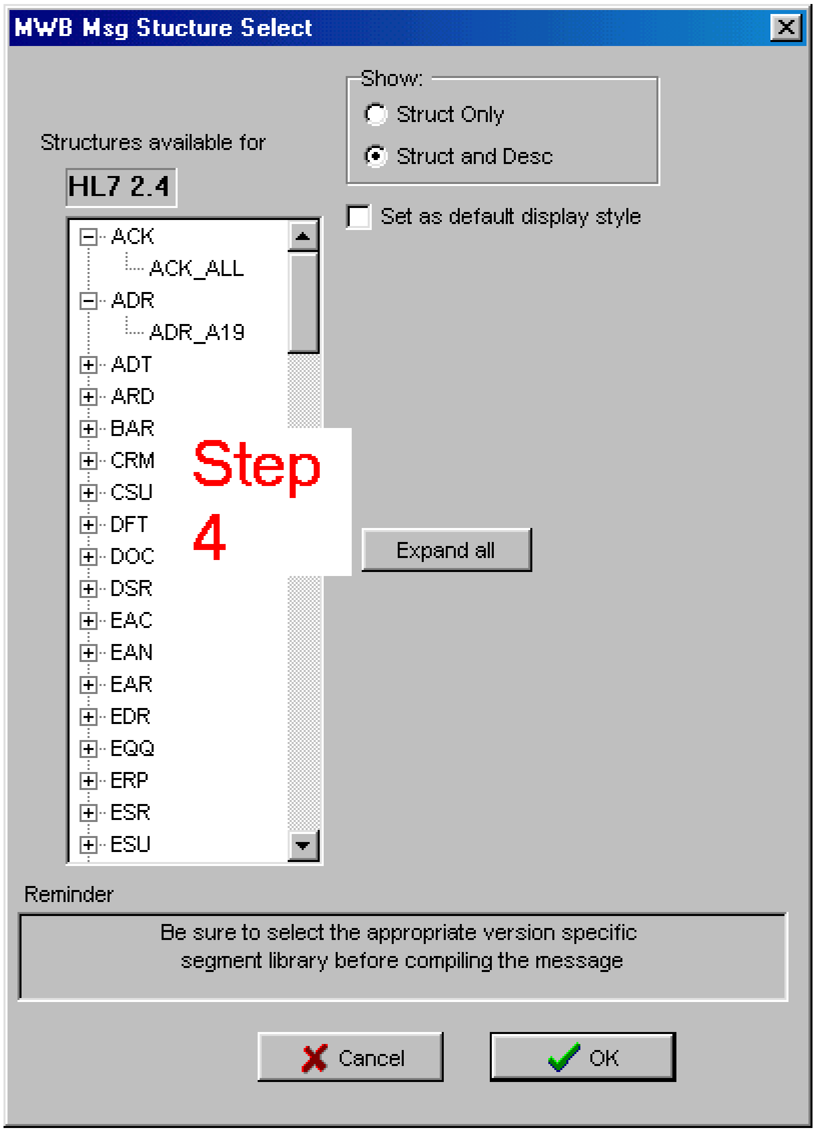
Specification
Msg Type ADR
Event Type A19
Ord Cntrl
StructID ADR_A19
Organization
Event Description
HL7 Version
Structure MSH,MSA,[ERR],[QAK],QRD,[QRF],{[EVN],PID,[PD1],[{ROL}],[{NK1}],PV1,[PV2],[{ROL}],[{DB1}],[{OBX}],[{AL1}],[{
Status
Spec Version HL7 2.4
Role ● Sender ○ Receiver
Conformance Type ● HL7 ○ Implement ○ Constrain ○ Extended
☑ Follow HL7 Processing Rules
□ Query Message

Message Tree

- □ MSH
- ⊟ MSA
- ⊟ [ERR
- ⊟ [QAK
- ⊟ QRD
- □ [QRF
- ⊟ {
  - ⊞ [EVN
  - ⊞ PID
  - ⊞ [PD1
  - ⊞ [{ROL
  - ⊞ [{NK1
  - ⊞ PV1
  - ⊞ [PV2
  - ⊞ [{ROL
  - ⊞ [{DB1
  - ⊞ [{OBX
  - ⊞ [{AL1
  - ⊞ [{DG1
  - ⊞ [DRG
  - ⊞ {
  - ⊞ [{GT1
  - ⊞ {
  - ⊞ [ACC
  - ⊞ [UB

Element Parameters
Element Name EVN
Description Event Type
Item#
Type Segment
Usage Optional
Repeatable □
Cardinality max min
Position
In Use ☑
Data Type
Length 0
Table
Data Link
Example Value
□ Fixed Value
Reference
Optionality Predicate
Implementation Notes
HL7 Definition

Figure 1 New Message Specification Dialog Box



MWB Msg Stucture Select

Structures available for HL7 2.4

Show:
○ Struct Only
● Struct and Desc

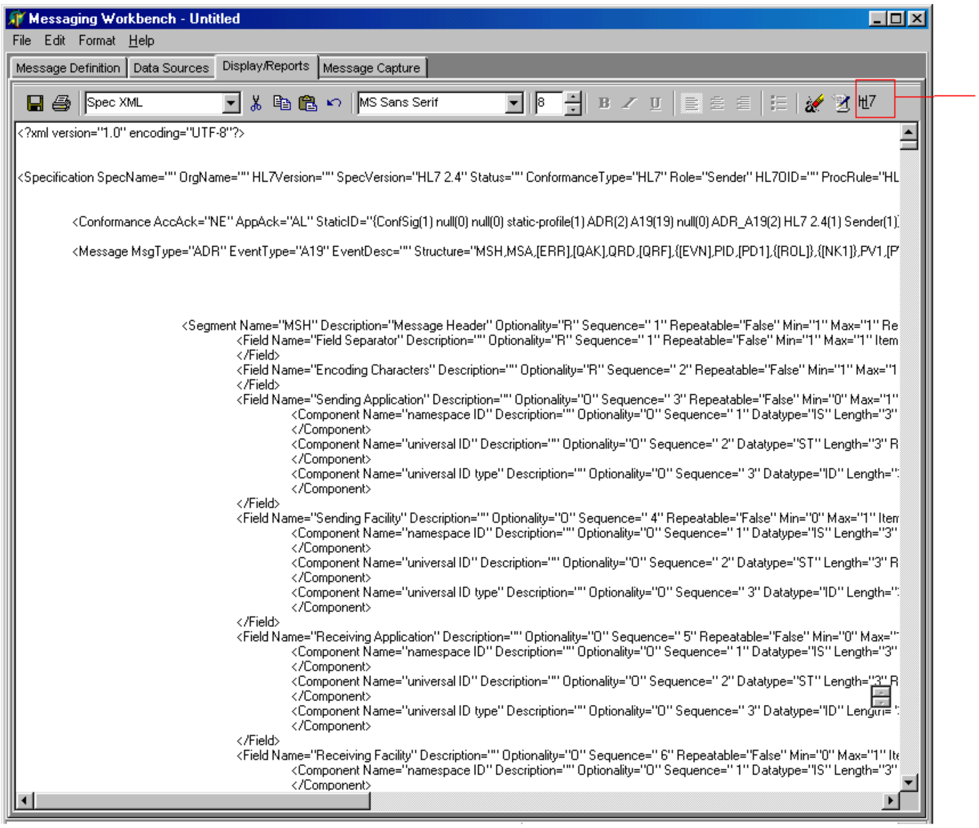
☐ Set as default display style

Step 4

Reminder

Be sure to select the appropriate version specific segment library before compiling the message

Figure 2 Message Structure Select Dialog Box



```
<?xml version="1.0" encoding="UTF-8"?>


<Specification SpecName="" OrgName="" HL7Version="" SpecVersion="HL7 2.4" Status="" ConformanceType="HL7" Role="Sender" HL7OID="" ProcRule="HL


          <Conformance AccAck="NE" AppAck="AL" StaticID="{ConfSig(1) null(0) null(0) static-profile(1) ADR(2) A19(19) null(0) ADR_A19(2) HL7 2.4(1) Sender(1)

          <Message MsgType="ADR" EventType="A19" EventDesc="" Structure="MSH,MSA,[ERR],[QAK],QRD,[QRF],{[EVN],PID,[PD1],{[ROL]},{[NK1]},PV1,[P



                              <Segment Name="MSH" Description="Message Header" Optionality="R" Sequence=" 1" Repeatable="False" Min="1" Max="1" Re
                                        <Field Name="Field Separator" Description="" Optionality="R" Sequence=" 1" Repeatable="False" Min="1" Max="1" Item
                                        </Field>
                                        <Field Name="Encoding Characters" Description="" Optionality="R" Sequence=" 2" Repeatable="False" Min="1" Max="1
                                        </Field>
                                        <Field Name="Sending Application" Description="" Optionality="O" Sequence=" 3" Repeatable="False" Min="0" Max="1"
                                                  <Component Name="namespace ID" Description="" Optionality="O" Sequence=" 1" Datatype="IS" Length="3"
                                                  </Component>
                                                  <Component Name="universal ID" Description="" Optionality="O" Sequence=" 2" Datatype="ST" Length="3" R
                                                  </Component>
                                                  <Component Name="universal ID type" Description="" Optionality="O" Sequence=" 3" Datatype="ID" Length="
                                                  </Component>
                                        </Field>
                                        <Field Name="Sending Facility" Description="" Optionality="O" Sequence=" 4" Repeatable="False" Min="0" Max="1" Item
                                                  <Component Name="namespace ID" Description="" Optionality="O" Sequence=" 1" Datatype="IS" Length="3"
                                                  </Component>
                                                  <Component Name="universal ID" Description="" Optionality="O" Sequence=" 2" Datatype="ST" Length="3" R
                                                  </Component>
                                                  <Component Name="universal ID type" Description="" Optionality="O" Sequence=" 3" Datatype="ID" Length="
                                                  </Component>
                                        </Field>
                                        <Field Name="Receiving Application" Description="" Optionality="O" Sequence=" 5" Repeatable="False" Min="0" Max="
                                                  <Component Name="namespace ID" Description="" Optionality="O" Sequence=" 1" Datatype="IS" Length="3"
                                                  </Component>
                                                  <Component Name="universal ID" Description="" Optionality="O" Sequence=" 2" Datatype="ST" Length="3" R
                                                  </Component>
                                                  <Component Name="universal ID type" Description="" Optionality="O" Sequence=" 3" Datatype="ID" Length="
                                                  </Component>
                                        </Field>
                                        <Field Name="Receiving Facility" Description="" Optionality="O" Sequence=" 6" Repeatable="False" Min="0" Max="1" Ite
                                                  <Component Name="namespace ID" Description="" Optionality="O" Sequence=" 1" Datatype="IS" Length="3"
                                                  </Component>
```

Step 6c

Figure 3 Display/Reports Tab

Procedure 2

Compiling a Message Profile with the Generator System

The generator system is Java software that generates standardized messages in Java from XML code.

Before you begin, you will need the XML file that the Messaging Workbench generated in Procedure 1. You will also need to start up a command line interface such as the Windows command prompt or a Linux shell.

How to Compile a Message Profile with the Generator System

1. To operate the generator system, type the following command:

```
java -classpath conf.jar ConfGen [flags] {source} {dest}
{package-name}
```

This will produce a JAR file that contains a Java version of the XML profile. You can use this JAR file in Procedure 3.

The meaning of “flags”, “source”, “dest” and “package-name” is as follows.

Flags, which are optional, are explained in Table 1.

Table 1 Flags and their Meaning

Flag	Meaning
-v	Displays all created files
-t	Runs an installation self test
-h	Displays how to run the program

“Source”, “dest” and “package-name” are mandatory. They are explained in Table 2.

Table 2 Meaning of Mandatory Parameters

Parameter	Meaning
source	Gives the location and filename of the XML file that creates the combined profile Java archive.
dest	Gives the location of the Java archive (JAR) that will be created.
package-name	Gives the name of the package to be created.

Procedure 3

Writing an HL7 Application with a Compiled Profile

This procedure shows how to use a compiled profile (produced by the generator system) to write an application. The compiled profile in this example is based upon an acknowledgement (ACK) message. There are many types of programs that you can write with the code from the compiled profile Java archive. For example, your program could listen for lab results and send a pager message to a physician when a result is abnormal.

How to Build a Program with the Compiled Profile Java Archive

1. Open a Java Integrated Development Environment (IDE).

2. Download the example JAR from http://hl7api.sourceforge.net/docs/profile_example/ack.jar. Make sure it is in your classpath.

3. Review the JavaDocs at http://hl7api.sourceforge.net/docs/profile_example/. Look for the class, constructor and method that you require.

4. Call the constructor by entering the following code:

   ```
   ConformanceMessage_XX msg = new ConformanceMessage_XX();
   ```

   Note: The constructor creates an underlying HAPI message object in memory to store the message. This message object can then be passed to other portions of the HAPI library for transmission.

5. Retrieve an object reference for the segments and the segment groups by entering the following code:

   ```
   ConformanceSegment_XX seg = msg.getSegment_XX( n );
   ```

   where the n in parentheses is the repetition number, if one is needed.

6. Retrieve an object reference for a field within a segment by typing the following code:

   ```
   ConformanceField_XX field = seg.getField_XX( n );
   ```

where the n in parentheses is the repetition number, if one is needed.

7. Retrieve the code for components within a field by entering the following code:

   ```
   ConformanceComponent_XX comp = field.getComponent_XX( );
   ```

8. Retrieve the code for sub-components within a component, if any, by entering the following code:

   ```
   ConformanceSubComponent_XX subc = comp.getSubComponent();
   ```

   You have completed this procedure.

HAPI Conformance Classes System

User Manual

Reference Material

Sample JavaDocs

A sample of the JavaDocs for an acknowledgement (ACK) message is located at http://hl7api.sourceforge.net/docs/profile_example/. This documentation is produced by the generator system. Note that this material is only a sample, since different material is generated from different profiles.

Sample Universal Markup Language Diagram

Figure 4 is a condensed sample of a Universal Markup Language (UML) diagram for a conformed ACK message. Only the methods needed to get to the data contained in subComponentIdentifier are present in the classes in the diagram.

This diagram is produced by the generator system. Note that this material is only a sample, since different code is generated from different profiles.

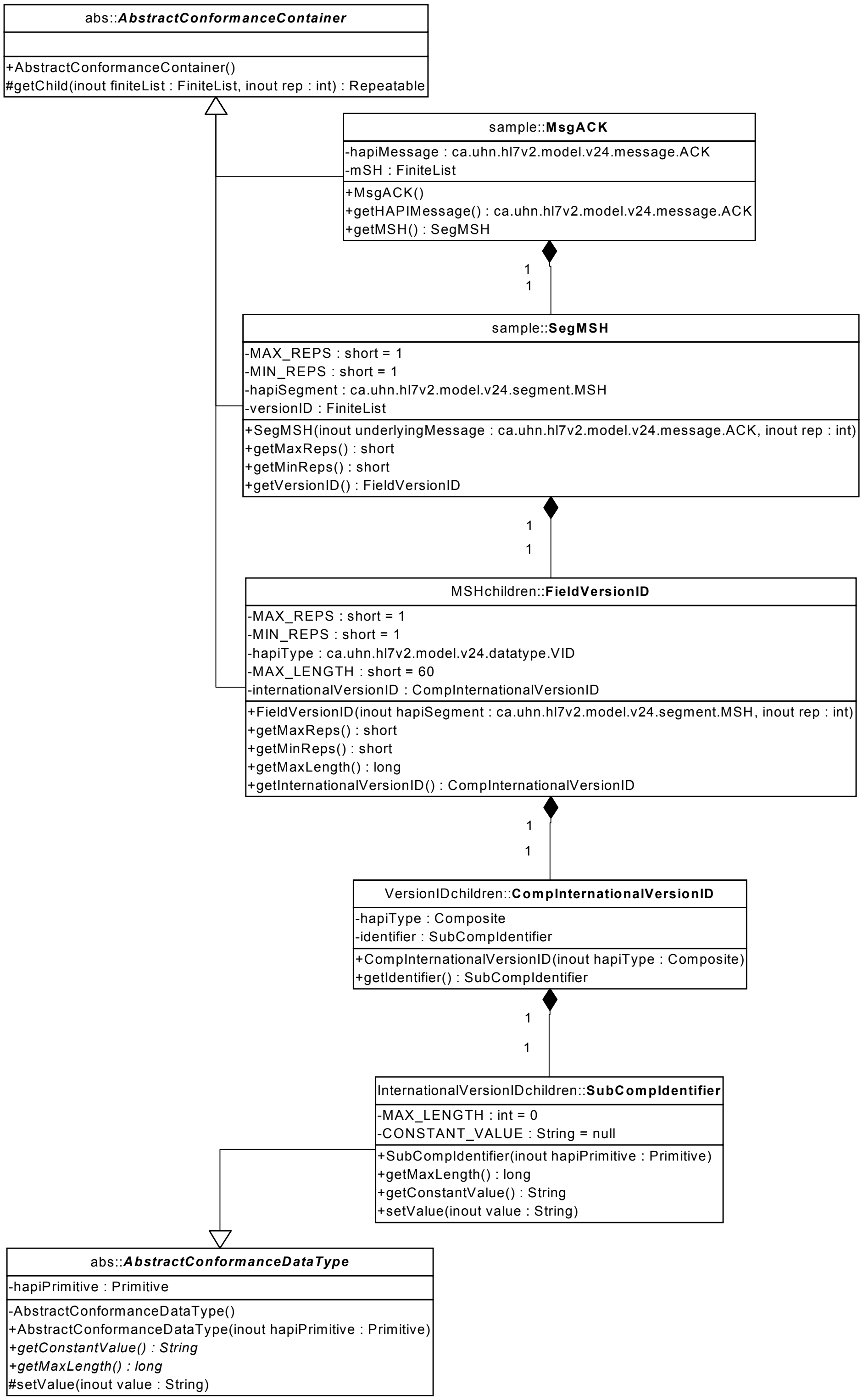

abs::AbstractConformanceContainer
+AbstractConformanceContainer()
#getChild(inout finiteList : FiniteList, inout rep : int) : Repeatable
sample::MsgACK
-hapiMessage : ca.uhn.hl7v2.model.v24.message.ACK
-mSH : FiniteList
+MsgACK()
+getHAPIMessage() : ca.uhn.hl7v2.model.v24.message.ACK
+getMSH() : SegMSH
1
1
sample::SegMSH
-MAX_REPS : short = 1
-MIN_REPS : short = 1
-hapiSegment : ca.uhn.hl7v2.model.v24.segment.MSH
-versionID : FiniteList
+SegMSH(inout underlyingMessage : ca.uhn.hl7v2.model.v24.message.ACK, inout rep : int)
+getMaxReps() : short
+getMinReps() : short
+getVersionID() : FieldVersionID
1
1
MSHchildren::FieldVersionID
-MAX_REPS : short = 1
-MIN_REPS : short = 1
-hapiType : ca.uhn.hl7v2.model.v24.datatype.VID
-MAX_LENGTH : short = 60
-internationalVersionID : CompInternationalVersionID
+FieldVersionID(inout hapiSegment : ca.uhn.hl7v2.model.v24.segment.MSH, inout rep : int)
+getMaxReps() : short
+getMinReps() : short
+getMaxLength() : long
+getInternationalVersionID() : CompInternationalVersionID
1
1
VersionIDchildren::CompInternationalVersionID
-hapiType : Composite
-identifier : SubCompIdentifier
+CompInternationalVersionID(inout hapiType : Composite)
+getIdentifier() : SubCompIdentifier
1
1
InternationalVersionIDchildren::SubCompIdentifier
-MAX_LENGTH : int = 0
-CONSTANT_VALUE : String = null
+SubCompIdentifier(inout hapiPrimitive : Primitive)
+getMaxLength() : long
+getConstantValue() : String
+setValue(inout value : String)
abs::AbstractConformanceDataType
-hapiPrimitive : Primitive
-AbstractConformanceDataType()
+AbstractConformanceDataType(inout hapiPrimitive : Primitive)
+getConstantValue() : String
+getMaxLength() : long
#setValue(inout value : String)

Figure 4 Sample UML Diagram

Sample Code

The following system uses a compiled profile Java archive (JAR). When creating an HL7 application, a programmer will write something like this code when he or she wants to use conformance classes.

```
/*
 * Created on Mar 16, 2003
 *
 * Example HL7 program, which uses the "Compiled Profile JAR"
 *
 */
import test.*;
import ca.uhn.hl7v2.conf.classes.exceptions.*;
/**
 * @author Shawn Dyck
 */
public class Sample_Using_Conformance_Classes {

     public static void main(String[] args){

           MsgACK msgAck = new MsgACK();

           try{
                System.out.println("Here is the max length for "
+
                             "the Sub-Component,Identifier\n" +
                             "that is contained within the
Component, "+
                             "VersionID \nthat is contained
within the Field, "+
                                      "VersionID \n"+
                             "that is contained within the
segment MSH: "+
                  msgAck.getMSH().getVersionID()
                                 .getInternationalVersionID()
                                 .getIdentifier()
                                 .getMaxLength());

                System.out.println("Constant value for
Identifier:" +
                msgAck.getMSH().getVersionID()
                                  .getInternationalVersionID()

.getIdentifier().getConstantValue());
```

```
                System.out.println("\nAbout to set the value for
the Identifyer to: 555");
                msgAck.getMSH().getVersionID()
                                   .getInternationalVersionID()

.getIdentifier().setValue("555");
     System.out.println("\nAbout to read back the value...");

                try {
                   ca.uhn.hl7v2.parser.XMLParser xmlParser = new
ca.uhn.hl7v2.parser.DefaultXMLParser();
                   String ackMessageInXML =
xmlParser.encode(msgAck.getHAPIMessage());
                   System.out.println("Identifier: " +
ackMessageInXML);
                } catch (Exception e) {
                   e.printStackTrace();
                   System.out.println("\nERR: " + e.toString() );
                }

                System.out.println("\nHere is the max length for
" +
                                    "the Sub-
Component,AlternateText\n" +
                                    "that is contained within the
Component, "+
                                    "VersionID \nthat is
contained within the Field, "
                                      "VersionID \n"+
                                    "that is contained within the
segment MSH: "+
                msgAck.getMSH().getVersionID()
                                   .getInternationalizationCode()
                                   .getAlternateText()
                                   .getMaxLength());
                // will throw an exception because the length is
tool long
                System.out.println("\nAbout to set the value for
the AlternateText to: 5555");
                System.out.println("please note, an exception
will be thrown");
                msgAck.getMSH().getVersionID()
                                   .getInternationalVersionID()

.getAlternateText().setValue("5555");


           }catch(ConfRepException cre){
                System.err.println("ConformanceRepException" +
                           cre.toString());
           }catch(ConfDataException cde){
                System.err.println("ConformanceDataException" +
cde.toString());
           }
      }

}
```

Troubleshooting

This section provides a description of error codes. See Table 3 for details.

Table 3 Command Line Troubleshooting

Problem	Probable Cause	Solution
File already exists error	There already is a generated file in the directory that was specified.	Either change the destination directory on the command line, or remove all previous generated files in the directory.
Invalid command line option error	An invalid flag was specified	Type `ConfGen -h` to get a list of all valid command line options.
Source path required error	The path to the XML profile was not specified.	Add the XML source path as the first argument on the command line.
Destination path required error	The path for the generated source was not specified.	Add a valid path where the generated source will be put. Add the path as the second argument on the command line.
Package name required error	The name of the package was not specified.	Add a package name as the last argument on the command line.
Invalid destination path error	The destination path that was specified doesn't exist.	The destination path must be a valid system path, and it must exist.
Invalid source file error	The XML source file doesn't exist or isn't in the directory specified	Make sure that the XML source file is in the specified directory.

Frequently Asked Questions

Message Profile

Q. What is a message profile and why is it important?

A. A message profile is a detailed specification for a message structure. For more information please visit the Conformance Special Interest Group page on www.hl7.org

Q. What is the message structure?

A. HL7 messages have a hierarchical format. At the top of the hierarchy is a message. A message can have one or more segments or segment groups. Each segment can have one or more fields. Each field can have one or more components. Each component may or may not have sub-components.

All of these items can carry various standardized codes. The message could, for example, carry the code ADT, which signifies the data of the admission of a patient. The segments could include a Message Header Segment (MSH), description of trigger events (EVN), patient identification (PID) and patient visit information (PV1). Fields could include such items as date (DT), person name (XPN) and address (XAD). And the same pattern works for components and sub-components.

Conformance

Q. What is conformance?

A. Conformance means that an HL7 message adheres to a certain message specification in terms of the number of fields, length of data, constant values and number of repetitions of segments in messages. The procedures in this documentation deal primarily with ensuring conformance to specifications that are expressed as XML “profiles”.

Q. What is being constrained to meet conformance?

A. The following items are being constrained:

- Minimum and maximum repetitions of segments
- Maximum length of data
- Constant values
- Structure of messages

Q Where can I get more information on conformance?

A. Visit the Web site http://hl7.org. Under the Committees section click on Special Interest Groups. On the left hand side under Special Interest Groups, click on Conformance.

Glossary

Compiled profile Java archive

The compiled profile Java archive (JAR) is software that allows access to ready-made Java code, which can be used in a program. Many types of programs can be written with the code from the compiled profile Java archive. For example, a program could create a user interface for data entry or a utility helper function that will build the HAPI library even further.

Conformance

The conformance framework is a way of writing detailed message specifications as XML documents, or “profiles”. These profiles describe allowable number of fields, length of data, constant values and number of repetitions of segments in messages.

Conformance classes

Conformance classes are special classes that facilitate the creation of HAPI message objects that fully conform to a given profile specification.

Component

A component is the fourth or fifth level in a message profile structure. Its parent is the field. Its child is the sub-component.

Field

A field is the third or fourth level in a message profile structure. Its parent is the segment and its child is the component.

Generator system

The generator system is Java software that generates message profiles in Java from XML code.

Health Level 7 (HL7)

HL7 is an organization that develops standards in the health care field, particularly electronic messaging standards. HL7 is accredited by the American National Standards Institute.

Message

A message is an electronic hierarchical structure that contains data about an event.

Message profile

A message profile is a specification for a message structure, which defines further restrictions beyond the corresponding message definition in the HL7 standard. Profiles are written in XML.

Messaging Workbench

Messaging Workbench is freeware that creates, maintains and displays message profiles. This software allows rapid prototyping of new profiles, and quick and easy alteration of existing profiles to meet new requirements.

Segment

A segment is the second or third level in a message profile structure. Its parent is the message or the segment group and its child is the field.

Segment group

A segment group is the second level in a message profile structure. Its parent is the message and its child is the segment.

Sub-Component

A sub-component is the fifth or sixth level in a message profile structure. Its parent is the field. It has no children.

Contact Information

For more information about HAPI Conformance Classes System, please contact:

- Bryan Tripp of the University Health Network in Toronto, Canada at bryan_tripp@users.sourceforge.net
- The software developers who created the system:
  - James Agnew at jait214@yahoo.ca
  - Paul Brohman at pbrohman@rogers.com
  - Mitch Delachevrotiere at hapimgd@hotmail.com
  - Shawn Dyck at sdyck613@yahoo.ca
  - Cory Metcalf at cory_Metcalf@hotmail.com
- The technical writer for this manual, Wendy Hall, at whall_ca@yahoo.ca

Related Documentation and Links

- Apache Ant: http://ant.apache.org
- Conformance tutorial: hl7.org/library/committees/Cont/ConformanceTutorial.ppt
- Health Level 7: http://hl7.org
- HAPI (Health Level 7 Application Programming Interface): http://hl7api.sourceforge.net
- HAPI developers forum: hl7api-devel@lists.sourceforge.net
- *HL7 Conformance User Guide*: http://hl7.org.
- *HL7 v2 Global Message Profile Library User's Guide*: http://hl7.org
- Kai U. Heitmann, Bernd Blobel and Joachim Dudeck, *HL7: Communication Standard in Medicine*. Cologne, Germany: Verlag Alexander Monch,1999
- Messaging Workbench, http://hl7.org.
- Profile tutorial: hl7.org/library/committees/Cont/ProfileTutorial
- Jennifer Puyenbroek, *V2 Global Message Profile Library Helps Build Conformance*. http://hl7.org/press/newsletter/aug2002/page01.asp
- Xerces, http://xml.apache.org

Index