

Quickstart Guide for NHapi

By Chad Chenoweth
Whitedog12@users.sourceforge.net
<http://nhapi.sourceforge.net>

Table of Contents

Table of Contents	2
Overview	3
Assembly Structure	4
.Net 1.1	4
.Net 2.0.....	4
Key Namespaces	5
.Net 1.1	5
.Net 2.0.....	5
Create a new HL7 message from within .NET	6
.Net 1.1	6
.Net 2.0.....	6
Parse an existing message for use within .NET	8
.Net 1.1	8
.Net 2.0.....	8
How to extend NHapi for use with your own institution messages?	9

Overview

NHapi is a port of a Java version named Hapi. It can be used to transform HL7 2.x into an object model for use in Microsoft .Net 1.1 and 2.0. It can also be used to transform data to/from pipe delimited formats and XML.

Pay careful attention to the version of nHapi that you are using, for there are namespace differences between the 1.1 version and the 2.0 versions that are for use with .NET. Notably, we rewrite the entire library to take advantage of Generics. We also renamed all interfaces to be IInterface to follow the .NET naming conventions.

Assembly Structure

.Net 1.1

There are 8 major assemblies:

- ca.uhn: assembly contains the core components for parsing/encoding messages. Contains the base classes and interfaces for datatypes, segments, and messages.
- ca.uhn.v21: assembly for HL7 version 2.1. Contains data types, segments, and messages that follow the HL7 2.1 schema.
- ca.uhn.v22: assembly for HL7 version 2.2. Contains data types, segments, and messages that follow the HL7 2.2 schema.
- ca.uhn.v23: assembly for HL7 version 2.3. Contains data types, segments, and messages that follow the HL7 2.3 schema.
- ca.uhn.v231: assembly for HL7 version 2.3.1. Contains data types, segments, and messages that follow the HL7 2.3.1 schema.
- ca.uhn.v24: assembly for HL7 version 2.4. Contains data types, segments, and messages that follow the HL7 2.4 schema.
- ca.uhn.v25: assembly for HL7 version 2.5. Contains data types, segments, and messages that follow the HL7 2.5 schema.
- unittest: assembly for all of the unit tests to validate the HL7 versions and base library

.Net 2.0

There are 9 major assemblies:

- NHapi.Base.: assembly contains the core components for parsing/encoding messages. Contains the base classes and interfaces for datatypes, segments, and messages.
- NHapi.Model.V21: assembly for HL7 version 2.1. Contains data types, segments, and messages that follow the HL7 2.1 schema.
- NHapi.Model.V22: assembly for HL7 version 2.2. Contains data types, segments, and messages that follow the HL7 2.2 schema.
- NHapi.Model.V23: assembly for HL7 version 2.3. Contains data types, segments, and messages that follow the HL7 2.3 schema.
- NHapi.Model.V231: assembly for HL7 version 2.3.1. Contains data types, segments, and messages that follow the HL7 2.3.1 schema.
- NHapi.Model.V24: assembly for HL7 version 2.4. Contains data types, segments, and messages that follow the HL7 2.4 schema.
- NHapi.Model.V25: assembly for HL7 version 2.5. Contains data types, segments, and messages that follow the HL7 2.5 schema.
- NHapi.NUnit: assembly for the unit tests for the base library and v2.x libraries.
- NHapi.NUnit.Additional: additional unit test library.

Key Namespaces

.Net 1.1

ca.uhn.dll:

- ca.uhn.hl7v2.model: provides for all of the base classes for data types, messages, segments, etc
- ca.uhn.hl7v2.parser: provides for all parsing of HL7 messages in pipe delimited and XML.
- ca.uhn.hl7v2.sourcegen: provides for all regeneration of the HL7 2.x schemas.

ca.uhn.v2X.dll:

- ca.uhn.hl7v2.model.v2X.datatype: contains the datatype (i.e. CE, TX, ST) for the HL7 2.X schema
- ca.uhn.hl7v2.model.v2X.group: contains the groups for the HL7 2.X schema. Groups are used to represent groupings of message fields (i.e. ADT_A01_Insurance)
- ca.uhn.hl7v2.model.v2X.message: contains the messages (i.e. ADT_A01, ACK, etc) for the HL7 2.X schema.
- ca.uhn.hl7v2.model.v2X.segment: contains the segments (i.e. PID, MSH, etc) for the HL7 2.X schema.

.Net 2.0

The namespaces within the 2.0 version of nHapi have been shortened and modified.

NHapi.Base.dll:

- NHapi.Base.Model: provides for all of the base classes for data types, messages, segments, etc
- NHapi.Base.Parser: provides for all parsing of HL7 messages in pipe delimited and XML.
- NHapi.Base.SourceGeneration: provides for all regeneration of the HL7 2.x schemas.

NHapi.Model.V2X.dll:

- NHapi.Model.V2X.Datatype: contains the datatype (i.e. CE, TX, ST) for the HL7 2.X schema
- NHapi.Model.V2X.Group: contains the groups for the HL7 2.X schema. Groups are used to represent groupings of message fields (i.e. ADT_A01_Insurance)
- NHapi.Model.V2X.Message: contains the messages (i.e. ADT_A01, ACK, etc) for the HL7 2.X schema.
- NHapi.Model.V2X.Segment: contains the segments (i.e. PID, MSH, etc) for the HL7 2.X schema.

Create a new HL7 message from within .NET

.Net 1.1

```
using ca.uhn.hl7v2.model.v23.message;
using ca.uhn.hl7v2.parser;
using ca.uhn.hl7v2;

PipeParser parser = new PipeParser();
ca.uhn.hl7v2.model.v231.message.QRY_R02 qry = new
ca.uhn.hl7v2.model.v231.message.QRY_R02();

qry.MSH.MessageType.MessageType.Value = "QRY";
qry.MSH.MessageType.TriggerEvent.Value = "R02";
qry.MSH.MessageType.MessageStructure.Value = "QRY_R02";
qry.MSH.FieldSeparator.Value = "|";
qry.MSH.SendingApplication.NamespaceID.Value="CohieCentral";
qry.MSH.SendingFacility.NamespaceID.Value = "COHIE";
qry.MSH.ReceivingApplication.NamespaceID.Value="Clinical Data Provider";
qry.MSH.ReceivingFacility.NamespaceID.Value=facility;
qry.MSH.EncodingCharacters.Value = @"^~\&";
qry.MSH.VersionID.VersionID.Value = "2.3.1";
qry.MSH.DateTimeOfMessage.TimeOfAnEvent.SetLongDate(DateTime.Now);
qry.MSH.MessageControlID.Value = messageControlId;
qry.MSH.ProcessingID.ProcessingID.Value="P";

XCN st = qry.QRD.getWhoSubjectFilter(0);
st.AssigningAuthority.UniversalID.Value = facility;
st.IDNumber.Value = mrn;

qry.QRD.QueryDateTime.TimeOfAnEvent.SetLongDate(DateTime.Now);
qry.QRD.QueryFormatCode.Value = "R";
qry.QRD.QueryPriority.Value = "I";
CE what = qry.QRD.getWhatSubjectFilter(0);
what.Identifier.Value = "RES";

return parser.encode(qry);
```

.Net 2.0

```
using NHapi.Base.Parser;
using NHapi.Base;

PipeParser parser = new PipeParser();
NHapi.Model.V231.Message.QRY_R02 qry = new
NHapi.Model.V231.Message.QRY_R02();

qry.MSH.MessageType.MessageType.Value = "QRY";
qry.MSH.MessageType.TriggerEvent.Value = "R02";
qry.MSH.MessageType.MessageStructure.Value = "QRY_R02";
qry.MSH.FieldSeparator.Value = "|";
qry.MSH.SendingApplication.NamespaceID.Value="CohieCentral";
qry.MSH.SendingFacility.NamespaceID.Value = "COHIE";
qry.MSH.ReceivingApplication.NamespaceID.Value="Clinical Data Provider";
qry.MSH.ReceivingFacility.NamespaceID.Value=facility;
qry.MSH.EncodingCharacters.Value = @"^~\&";
qry.MSH.VersionID.VersionID.Value = "2.3.1";
qry.MSH.DateTimeOfMessage.TimeOfAnEvent.SetLongDate(DateTime.Now);
qry.MSH.MessageControlID.Value = messageControlId;
qry.MSH.ProcessingID.ProcessingID.Value="P";

XCN st = qry.QRD.getWhoSubjectFilter(0);
st.AssigningAuthority.UniversalID.Value = facility;
st.IDNumber.Value = mrn;
```

```
qry.QRD.QueryDateTime.TimeOfAnEvent.SetLongDate(DateTime.Now);
qry.QRD.QueryFormatCode.Value = "R";
qry.QRD.QueryPriority.Value = "I";
CE what = qry.QRD.getWhatSubjectFilter(0);
what.Identifier.Value = "RES";

return parser.encode(qry);
```

Parse an existing message for use within .NET

.Net 1.1

```
string message =
@"MSH|^~\&|SENDING|SENDER|RECV|INST|20060228155525||QRY^R02^QRY_R02|1|P|2.3|
QRD|20060228155525|R|I|||10^RD&Records&0126|38923^^^^^^&INST|||";

ca.uhn.hl7v2.parser.PipeParser parser = new ca.uhn.hl7v2.parser.PipeParser();

ca.uhn.hl7v2.model.Message m = parser.parse(message);

ca.uhn.hl7v2.model.v23.message.QRY_R02 qryR02 = m as
ca.uhn.hl7v2.model.v23.message.QRY_R02;

you can now access properties of the qryR02 message as such:

Console.WriteLine(qryR02.QRD.getWhoSubjectFilter(0).IDNumber.Value); //Should be 38923
```

.Net 2.0

```
using NHapi.Base.Model;
using NHapi.Base.Parser;
using NHapi.Base;
using NHapi.Model.V23;
using NHapi.Model.V23.Message;
using NHapi.Model.V23.Segment;

string message =
@"MSH|^~\&|SENDING|SENDER|RECV|INST|20060228155525||QRY^R02^QRY_R02|1|P|2.3|
QRD|20060228155525|R|I|||10^RD&Records&0126|38923^^^^^^&INST|||";

PipeParser parser = new PipeParser();

IMessage m = parser.parse(message);

QRY_R02 qryR02 = m as QRY_R02;
Console.WriteLine(qryR02.QRD.GetWhoSubjectFilter(0).IDNumber.Value);
```

How to extend NHapi for use with your own institution messages?

There are two ways to accomplish this:

1. Purchase the HL7 database and tweak this for your institution
2. Extend from the existing classes

For the 1st option, I've done this at my own institution. The hard part is going to be buying the HL7 database (which is about \$1200) from <https://www.hl7.org/library/bookstore/>. Once you have the database, you'll have to meddle through the tables and tweak the segments to match your institution. You can then use the SourceGeneration code to generate that HL7 schema set.

For the 2nd option, thanks to dabudddhaman for suggesting this. Here is a link to some help:

http://sourceforge.net/tracker/index.php?func=detail&aid=1545143&group_id=166900&atid=840800. Also, check the UnitTests NHapi.NUnit.Additional for examples.