Healthcare Enterprise – IT Architecture Building Blocks

Canonical Message Model for a HL7 Enterprise

Michael Czapski (michael@czapski.id.au), October 2010, Rel. 1.0.1

Table of Contents

Abstract

In any but the simplest of HL7 messaging environments there will be multiple sources and multiple destinations of HL7 messages. It is very unlikely that all, or even a majority of these, will use exactly the same HL7 message structures in terms of versions, optional/mandatory segments, extension Z segments, and so on. A sensible approach to dealing with these kinds of issues, and a key component of the HL7 Enterprise Architecture, is the so called Canonical (or Common) Message Model (CMM). The CMM works hand-in-glove with the enterprise architecture in which transformation to/from the CMM is performed at the edges of the integration domain. This article discusses major considerations and works through the mechanics of deriving a Canonical Message Model for a fictitious Healthcare Enterprise and implementing it using the Oracle SOA Suite 11g HL7 tooling as an example. The article will also discuss and illustrate a mechanism for injecting arbitrary metadata into the canonical message, generated by the B2B Document Editor, in such a way that it is ignored by the Edge-dwelling B2B infrastructure but is significant to the SOA infrastructure.

Introduction

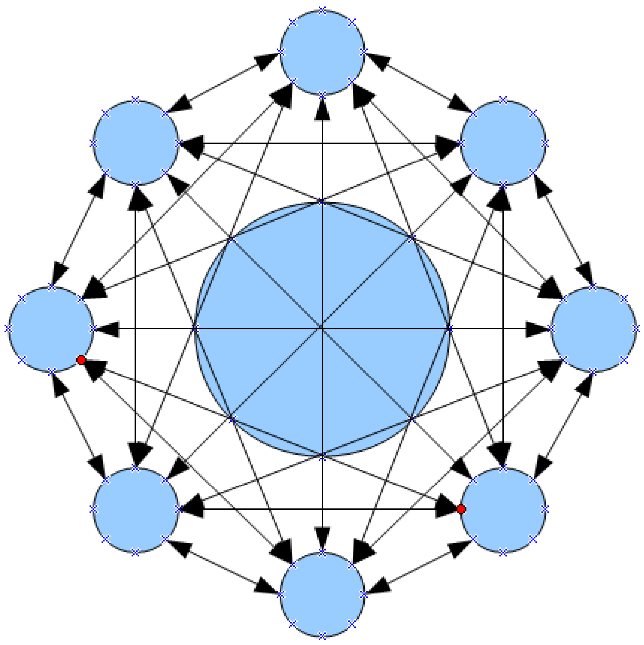


Illustration 1: Point-to-point message transformation

In any but the simplest of HL7 messaging environments there will be multiple sources and multiple destinations of HL7 messages. It is very unlikely that all, or even a majority of these, will use exactly the same HL7 message structures in terms of versions, optional/mandatory segments, extension Z segments, and so on. These differences necessitate transformation of messages from/to formats used by source/target systems, in extreme cases leading to the virtual point-to-point integration model, negating most benefits of having a modern integration infrastructure.
A sensible approach to dealing with these kinds of issues, and a key component of the Enterprise Architecture, is the so called Common (or Canonical) Message Model (CMM).

The Canonical Message Model works hand-in-glove with the enterprise architecture in which transformation to/from the CMM is performed at the edges of the integration domain. This ultimately leads to the reduction in the number of transformations to the number of external touch points. This also tightly couples endpoints and their transformations, localising change domains and insulating the rest of the enterprise integration infrastructure from the changes in endpoints.

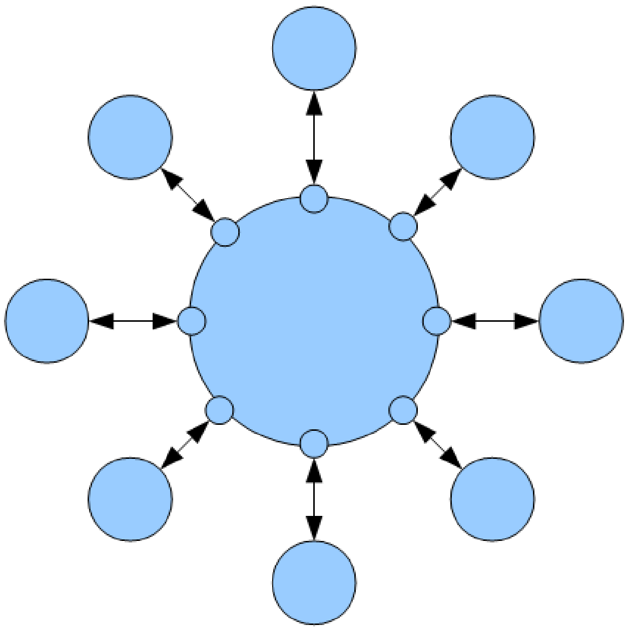


Illustration 2: Using Canonical Message Model

As a consequence of this design the integration solutions need only be concerned with message routing.

To accommodate all possible data of interest to the enterprise the CMM will have to be constructed in such a way that it is inclusive of all data.

It is likely that an enterprise will have multiple business domains, for example Patient Administration, Patient Billing, Catering, and so on, which while dealing with patient information in some manner are so different in nature that it is impractical to attempt to derive a canonical message model that would include data from all. This may lead to multiple canonical message models, one for each business domain. This is a valid approach and likely to be more robust then attempting to derive a single model for all business domains in the enterprise.

No enterprise remains unchanged over time. It is inevitable that any canonical model will need to evolve to accommodate changes. This evolution will need to be managed. The best approach to managing change will be to design the CMM in such a way that it can be extended and that extensions can be identified.

Managing changes to the model over time will require design of the model data structures so that they can only be extended “at the end of the structure”. This is the kind of policy the HL7 v2.x standard uses. Just what “at the end of the structure” means depends on the message structures in question. In the case of HL7 v2.x, for example, this would be at the end of message segments, at the end of components, etc..

A correctly designed transformation will not break if it has more data in the message then it has been designed to deal with. This statement must be qualified. Depending on the choice of technologies that are used to represent message structure, validate message structure and implement transformation support, there my arise an issue of tight coupling between technologies which eliminates flexibility.

Technologies that require implicit or explicit validation of the message structure, by for example parsing messages into a build-time fixed representation, will break when presented a non-compliant message, as a new variant of the canonical message will be. This kind of issue will arise for example when using Java CAPS Object Type Definitions or any XML-based technologies with Schema validation enabled. In the former case one could add “step-down” transformation proxies to transform new common message variants to old common message formats at the endpoints which are not upgraded to deal with new variants. In the later case XML schema validation might have to be disabled until transformations are upgraded to deal with, and validate, new message variants.

Implicit in support for evolution of canonical message model is the notion of versioning and, consequential identification of structure versions. A version system, recognised by the transformation infrastructure, would address the issue of identification of different versions if version identifiers were an integral part of the model and if each message carried meta data identifying structure version. This in turn leads to the consideration of the enveloped message pattern as the means of adding appropriate meta data or to adoption of messaging technologies which are transparently extensible, for example XML, or both.

For the remainder of this article let's assume that the following architectural decisions were made:

1. A canonical message model will be developed to support all HL7-capable clinical message sources and destinations in the enterprise
2. The canonical message model will be based on the HL7 version 2.3.1 standard
3. The canonical message model will include all HL7 v2.3.1 segments that occur in any of the messages provided by source systems or required by destination systems
4. The canonical message model will use the XML representation for HL7 v2.3.1 messages
5. The XML Schema for the canonical message model will use an attribute with the name “CMMVersion” on the root node of the structure to identify the version canonical message model to the message uses.
6. Transformation between endpoint-specific HL7 messages and canonical messages will take place at the edge of the integration domain (at the endpoint)

For the remainder of this article let's assume that the following major technical decisions were made:

1. Oracle SOA Suite 11g will be used to implement the canonical message model and the enterprise integration infrastructure
2. Oracle SOA Suite B2B HL7 functionality will be used to integrated source and destination systems and transform HL7 v2.x delimited messages into their XML equivalents
3. Mediator components will be used to transform HL7 v2.x XML messages into canonical message model messages, enriching each with the CMMVersion attribute and its value

Implied technical decisions, exploiting capabilities of the SOA Suite, are:

1. Oracle SOA Suite B2B Document Editor will be used to develop and maintain the common message model artefacts
2. Oracle SOA Suite B2B runtime will be configured to:
    i. Perform transparent transformation between HL7 v2.x delimited and their corresponding HL7 v2.x XML formats
    ii. Handle functional acknowledgements
    iii. Handle payload persistence and message tracking
    iv. Handle KPI collection and display for all HL7 endpoints
    v. Handle multiplexing of multiple senders to a single listener
    vi. Handle broadcast of a single message to multiple receivers

Modelling the Canonical Message

Let's assume that two systems in the enterprise use the following HL7 messages:

Type and Trigger	*HL7 Version*	*Description*
ADT A01	V2.3.1	Admin/Visit Notification
ADT A03	V2.3.1	Discharge/End Visit
ADT A08	V2.3.1	Update Patient Information
ADT A17	V2.3.1	Swap Patients

Table 1 HL7 v2.x Delimited messages in the Enterprise

There might be (and very likely will be) many more systems and many more messages. This example illustrates a method which will hold for any number of systems and messages.

With the objective of deriving a single message structure that will include all segments in the corresponding HL7 v2.3.1 messages let's prepare a table with Message and Event Types as columns and HL7 v 2.3.1 segments as rows, with assistive symbology where "?" Means 0 or 1 (optional), "*" means 0 or more (optional repeating) and "-" means no such segment in this event type. No special symbol means required.

	A01	*A03*	*A08*	*A17*
1	MSH	MSH	MSH	MSH
2	EVN	EVN	EVN	EVN
3	PID	PID	PID	PID
4	PD1?	PD1?	PD1?	PD1?
5	NK1*	-	NK1*	-
6	PV1	PV1	PV1	PV1
7	PV2?	PV2?	PV2?	PV2?
8	DB1*	DB1*	DB1*	DB1*
9	OBX*	-	OBX*	OBX*
10	AL1*	-	AL1*	-
11	DG1*	DG1*	DG1*	-
12	DRG?	DRG?	DRG?	-
Group PROCEDURE*				
13A	PR1	PR1	PR1	-
13B	ROL*	ROL*	ROL*	-
13C	GT1*	-	GT1*	-
Group INSURANCE*				
14A	IN1	-	IN1	-
14B	IN2?	-	IN2?	-
14C	IN3*	-	IN3*	-
15	ACC?	-	ACC?	-
16	UB1?	-	UB1?	-
17	UB2?	-	UB2?	-
18	-	OBX*	-	-
19	-	-	-	PID
20	-	-	-	PD1?
21	-	-	-	PV1
22	-	-	-	PV2?
23	-	-	-	DB1*
24	-	-	-	OBX*
25	-	Z01?	Z01?	-

Table 2: Segment-level comparison

Inspection of columns A01 and A08 reveals that the same segments are used in both and that they are a large superset of the segments used in other event types. To construct a structure that would cater for A01, A08 and A03 requires addition of an optional, repeating OBX segment (row 18). To construct a structure that would also cater for A17 would require addition of PID, PD1, PV1, PV2, DB1 and OBX segments. The PID ad PV1 segments, which in A17 are required, would have to be made optional to allow A01, A03 and A08 to be correctly parsed. This would mean that a malformed A17 with a missing PID or PV1 segment would be accepted as valid. We are prepared to live with that for the sake of the benefits of a canonical message model.

One of our systems sends messages with a custom Z01 segment. This segment's structure is shown in Table 3.

Sequence	*Length*	*Data Type*	*Element Name*
1	1	IS	Original Gender
2	1	IS	Current Gender
3	26	TS	Date of Change
4	1	IS	Legal Gender

Table 3: Z01 Segment composition

The segment is optional, since not very many people change their gender, but all components are required, since when they do all the required pieces of information are known.

Knowledge of our systems tells us that no system ever sends segments other then MSH, EVN, PID and PV1, and Z01. This is a grossly simplifying statement. It is unlikely that this will be the case in real systems but for the sake of brevity in this article it is a reasonable simplification. The method will hold no matter which and how many segments are included. It is likely, in fact, that an enterprise would include all optional segments in the canonical message model if for no other reason then to ensure that a system that one day gets upgraded, and starts including one of the optional segments, does not break the model.

With this simplification, however, our message will consist of HL7 segments listed in Table 3.

CMM
MSH
EVN
PID
PV1
PID?
PV1?
Z01?

Table 4: Canonical Message Model HL7 Segments

Analysis aimed at deriving a canonical message model would then proceed deeper to inspect components, subcomponents and fields, and determine whether and what might need to be done to restrict, relax or modify rules that might apply to them, and include or exclude any in/from the model. We will stop at the segment level for the purpose of this article.

Building the CMM Structure

Since the tooling we decided to use is the Oracle SOA Suite 11g we will use the Oracle B2B Document Editor to build the CMM structure and produce the external forms of it for use in our integration work.

In our example A01 and A08 both have the same set of segments, rows 1 through 17 in Table 2. We will need to add an optional repeating OBX segment to account for the requirements of A03 (row 18 in Table 2). This will address standard structures for A01, A03 and A08. To account for the requirement of the A17 we need to add PID, PD1, PV1, PV2, DB1 and OBX segments, shown in rows 19 through 24 in Table 2. Finally, we will add the custom Z01 segment.

Let's start the B2B Document Editor: Start → Programs → Oracle → Oracle B2B.

Click New Document. Expand HL7 → 2.3.1 → Event A01: ADT/ACK and select ADT: ADT message node.

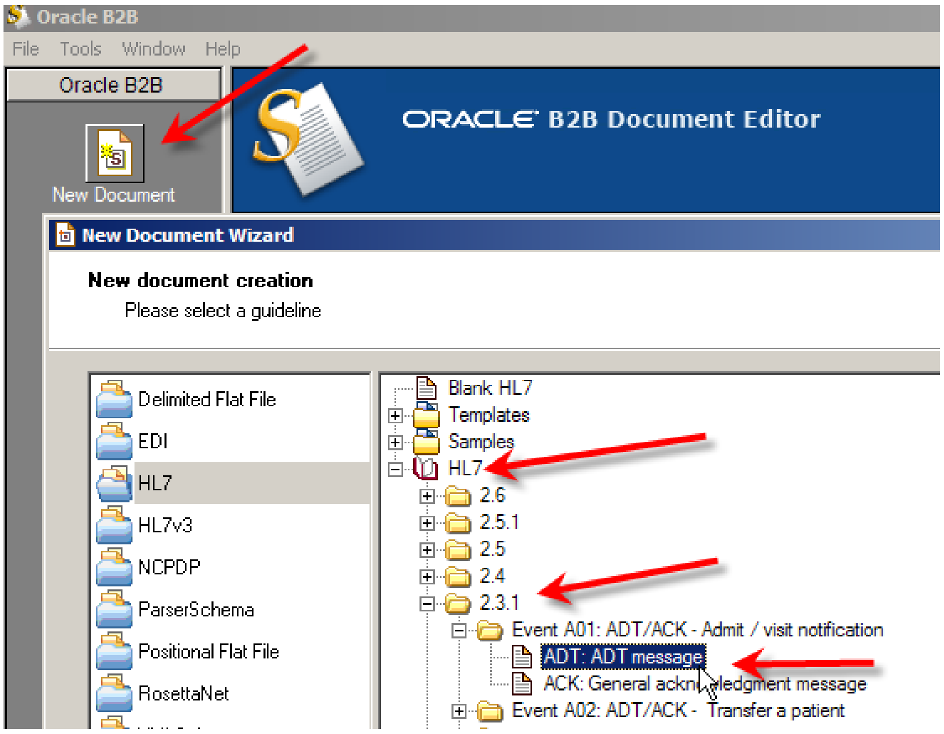
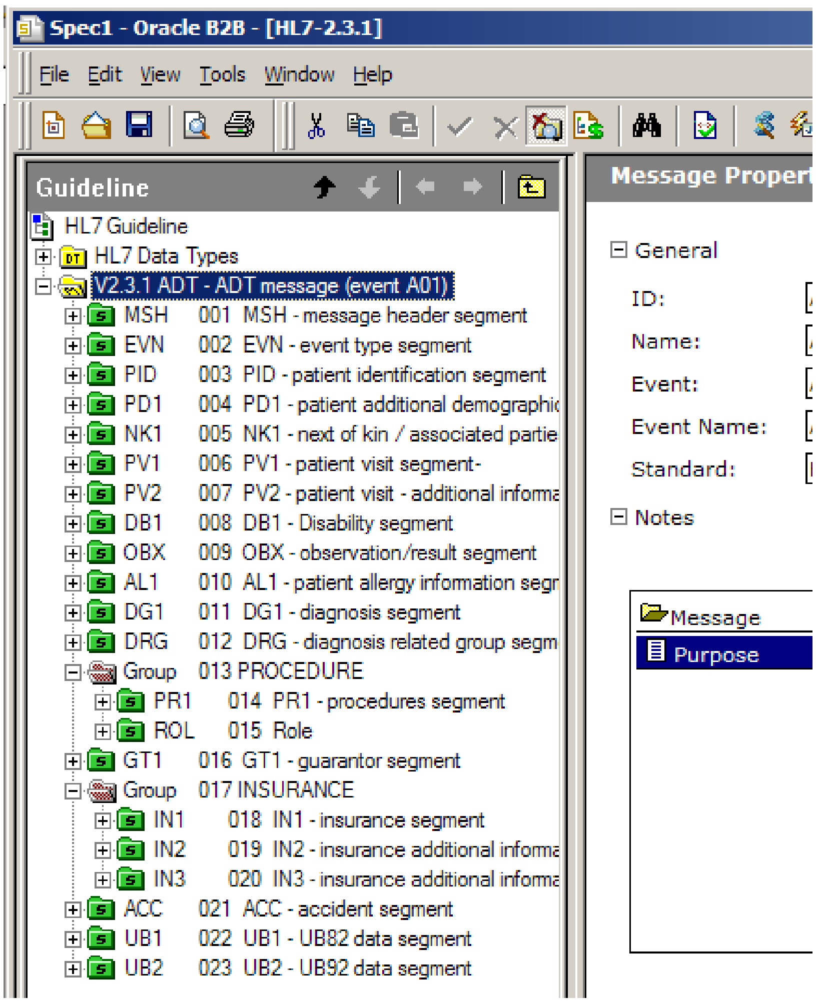


Illustration 3: Choose standard ADT A01 message to modify

The Spec / Guideline has all the segments needed for A01 (because this is A01) and A08 (because both have the same structure).

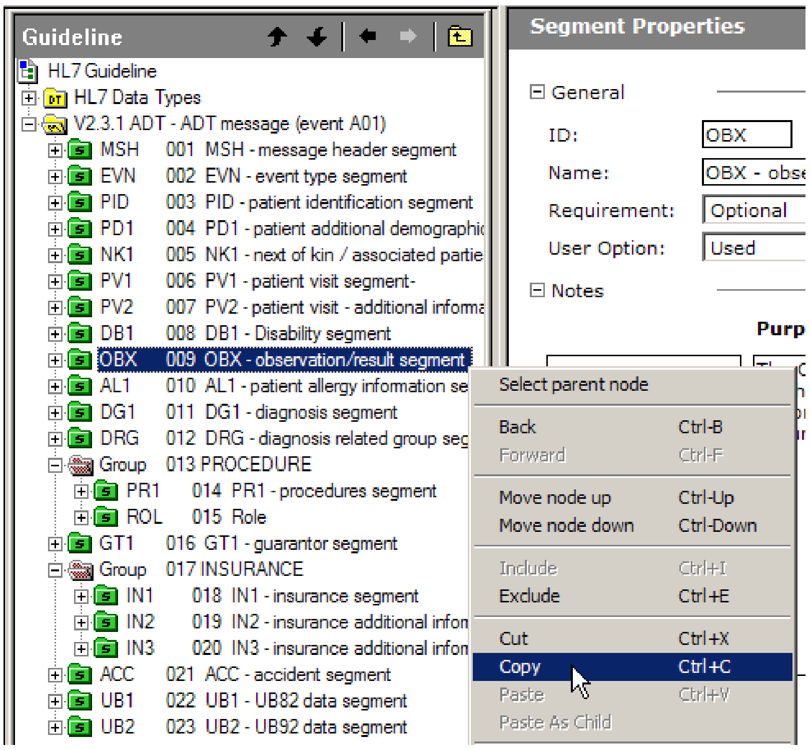


```
HL7 Guideline
  HL7 Data Types
  V2.3.1 ADT - ADT message (event A01)
    MSH    001  MSH - message header segment
    EVN    002  EVN - event type segment
    PID    003  PID - patient identification segment
    PD1    004  PD1 - patient additional demographic
    NK1    005  NK1 - next of kin / associated partie
    PV1    006  PV1 - patient visit segment-
    PV2    007  PV2 - patient visit - additional informa
    DB1    008  DB1 - Disability segment
    OBX    009  OBX - observation/result segment
    AL1    010  AL1 - patient allergy information segr
    DG1    011  DG1 - diagnosis segment
    DRG    012  DRG - diagnosis related group segm
    Group  013 PROCEDURE
      PR1    014  PR1 - procedures segment
      ROL    015  Role
    GT1    016  GT1 - guarantor segment
    Group  017 INSURANCE
      IN1    018  IN1 - insurance segment
      IN2    019  IN2 - insurance additional informa
      IN3    020  IN3 - insurance additional informa
    ACC    021  ACC - accident segment
    UB1    022  UB1 - UB82 data segment
    UB2    023  UB2 - UB92 data segment
```

Illustration 4: Standard A01 structure

Let's now add an optional, repeating OBX segment to the end of the structure (append) to accommodate the requirement of the A03 structure.

Right-click an OBX segment in the A01 message and choose Copy.

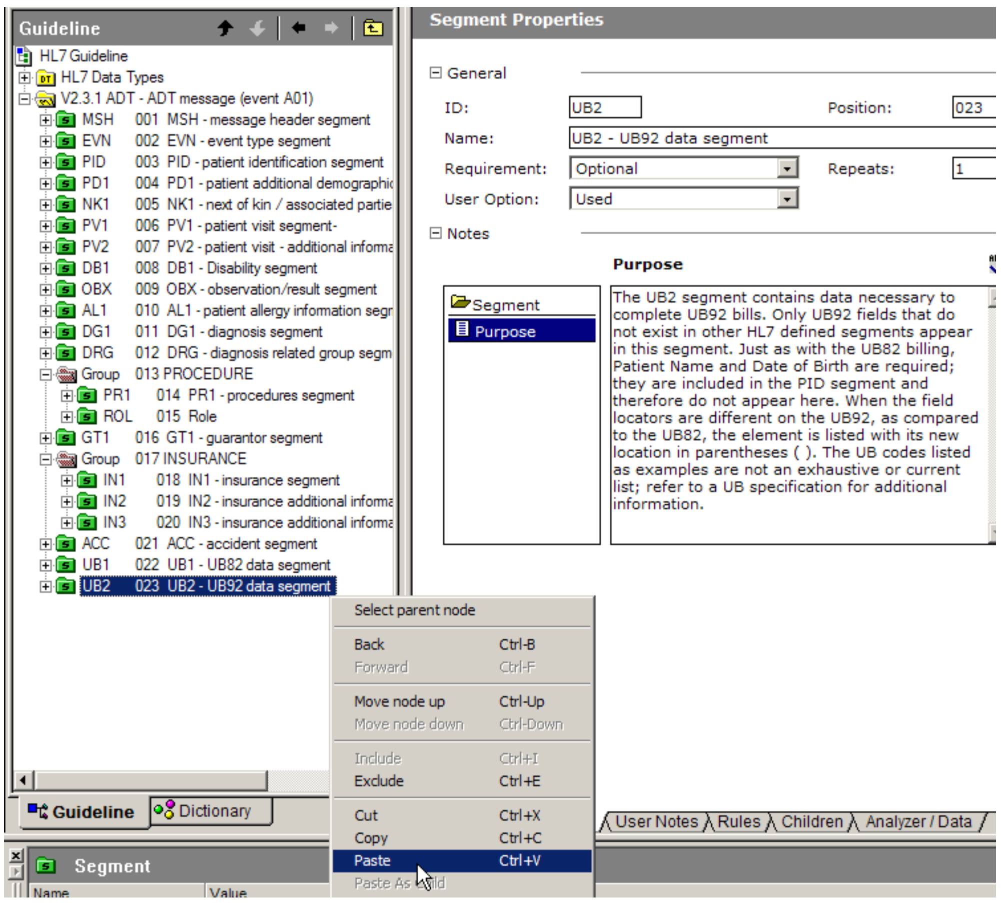


HL7 Guideline
HL7 Data Types
V2.3.1 ADT - ADT message (event A01)
MSH 001 MSH - message header segment
EVN 002 EVN - event type segment
PID 003 PID - patient identification segment
PD1 004 PD1 - patient additional demographic
NK1 005 NK1 - next of kin / associated partie
PV1 006 PV1 - patient visit segment-
PV2 007 PV2 - patient visit - additional informa
DB1 008 DB1 - Disability segment
OBX 009 OBX - observation/result segment
AL1 010 AL1 - patient allergy information se
DG1 011 DG1 - diagnosis segment
DRG 012 DRG - diagnosis related group seg
Group 013 PROCEDURE
PR1 014 PR1 - procedures segment
ROL 015 Role
GT1 016 GT1 - guarantor segment
Group 017 INSURANCE
IN1 018 IN1 - insurance segment
IN2 019 IN2 - insurance additional infor
IN3 020 IN3 - insurance additional infor
ACC 021 ACC - accident segment
UB1 022 UB1 - UB82 data segment
UB2 023 UB2 - UB92 data segment

General
ID: OBX
Name: OBX - obse
Requirement: Optional
User Option: Used
Notes

Illustration 5: Copy OBX segment

Right-click on the last node of the structure (UB2) and choose “Paste”.



HL7 Guideline
HL7 Data Types
V2.3.1 ADT - ADT message (event A01)
MSH 001 MSH - message header segment
EVN 002 EVN - event type segment
PID 003 PID - patient identification segment
PD1 004 PD1 - patient additional demographic
NK1 005 NK1 - next of kin / associated partie
PV1 006 PV1 - patient visit segment-
PV2 007 PV2 - patient visit - additional informa
DB1 008 DB1 - Disability segment
OBX 009 OBX - observation/result segment
AL1 010 AL1 - patient allergy information segr
DG1 011 DG1 - diagnosis segment
DRG 012 DRG - diagnosis related group segm
Group 013 PROCEDURE
PR1 014 PR1 - procedures segment
ROL 015 Role
GT1 016 GT1 - guarantor segment
Group 017 INSURANCE
IN1 018 IN1 - insurance segment
IN2 019 IN2 - insurance additional informa
IN3 020 IN3 - insurance additional informa
ACC 021 ACC - accident segment
UB1 022 UB1 - UB82 data segment
UB2 023 UB2 - UB92 data segment

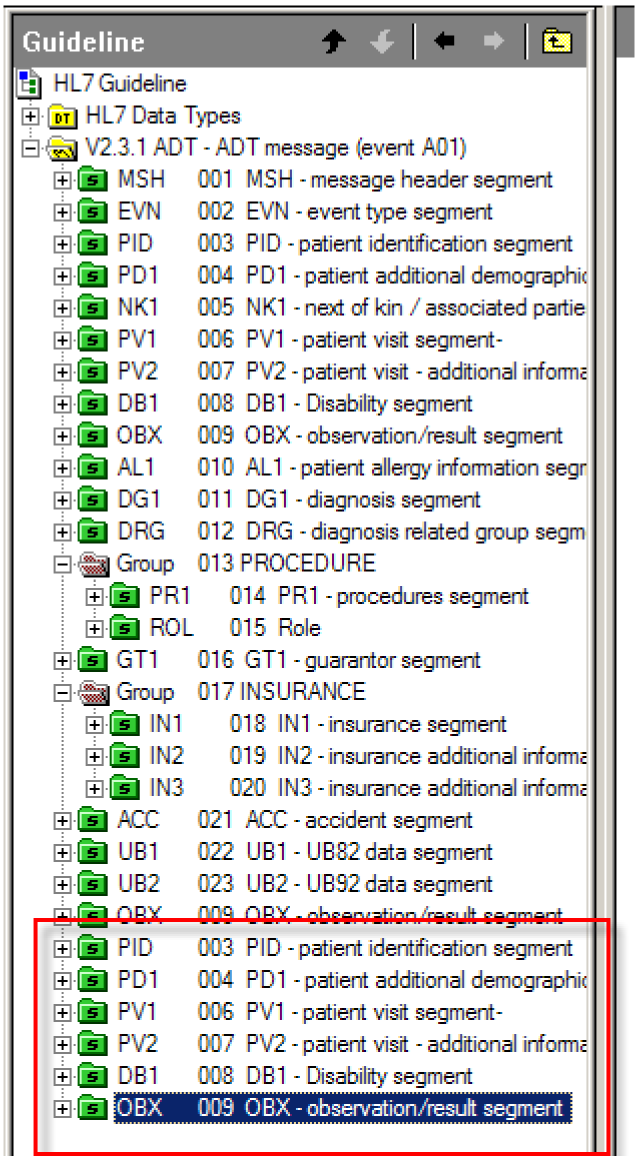
General
ID: UB2 Position: 023
Name: UB2 - UB92 data segment
Requirement: Optional Repeats: 1
User Option: Used
Notes
Segment
Purpose

Purpose

The UB2 segment contains data necessary to complete UB92 bills. Only UB92 fields that do not exist in other HL7 defined segments appear in this segment. Just as with the UB82 billing, Patient Name and Date of Birth are required; they are included in the PID segment and therefore do not appear here. When the field locators are different on the UB92, as compared to the UB82, the element is listed with its new location in parentheses ( ). The UB codes listed as examples are not an exhaustive or current list; refer to a UB specification for additional information.

Illustration 6: Append OBX to the end of the structure

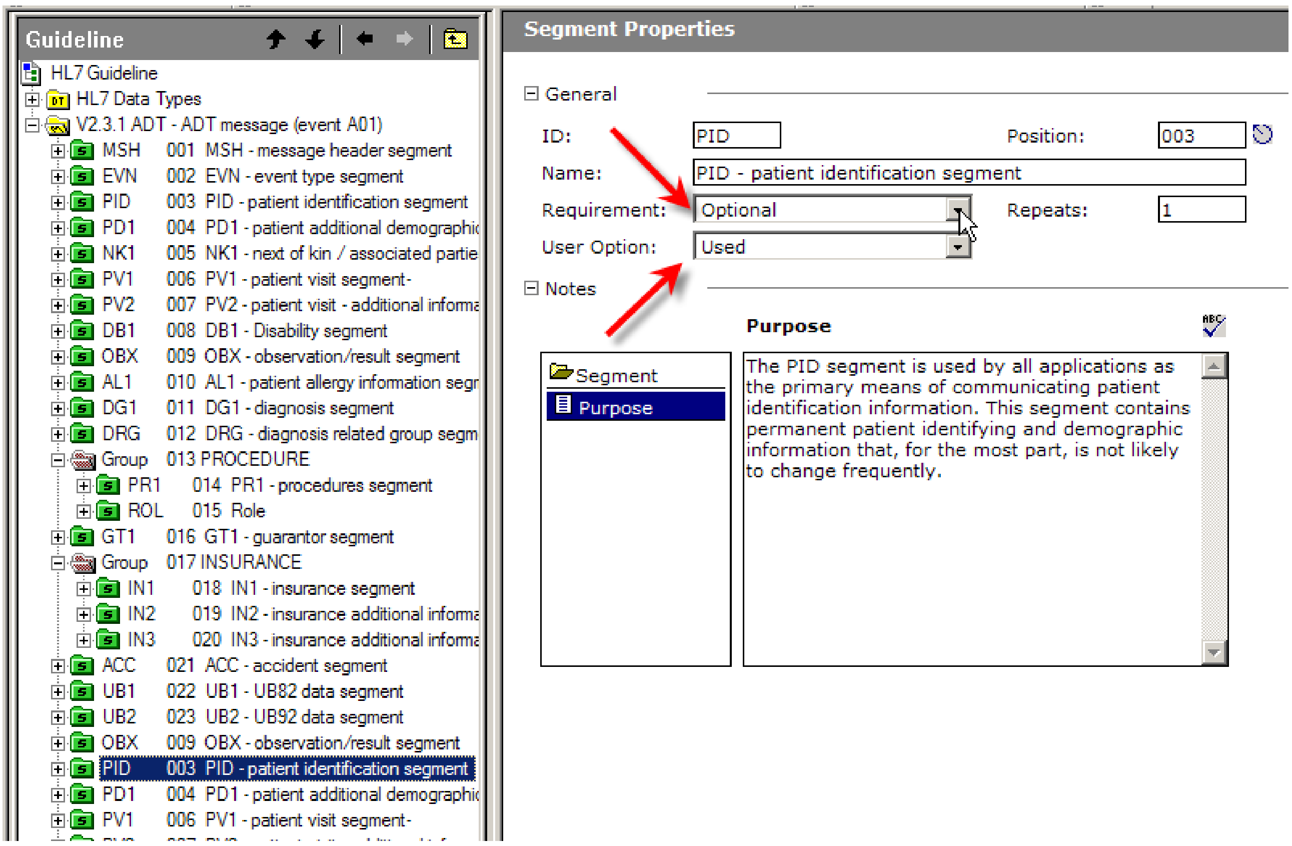
Now the message reflects the requirements of the standard A01, A03 and A08 messages. Using the same copy/paste process let's append PID, PD1, PV1, PV2, DB1 and OBX segments.



```
Guideline
HL7 Guideline
+ HL7 Data Types
- V2.3.1 ADT - ADT message (event A01)
    + MSH     001  MSH - message header segment
    + EVN     002  EVN - event type segment
    + PID     003  PID - patient identification segment
    + PD1     004  PD1 - patient additional demographi
    + NK1     005  NK1 - next of kin / associated partie
    + PV1     006  PV1 - patient visit segment-
    + PV2     007  PV2 - patient visit - additional informa
    + DB1     008  DB1 - Disability segment
    + OBX     009  OBX - observation/result segment
    + AL1     010  AL1 - patient allergy information segr
    + DG1     011  DG1 - diagnosis segment
    + DRG     012  DRG - diagnosis related group segm
    - Group   013 PROCEDURE
        + PR1     014  PR1 - procedures segment
        + ROL     015  Role
    + GT1     016  GT1 - guarantor segment
    - Group   017 INSURANCE
        + IN1     018  IN1 - insurance segment
        + IN2     019  IN2 - insurance additional informa
        + IN3     020  IN3 - insurance additional informa
    + ACC     021  ACC - accident segment
    + UB1     022  UB1 - UB82 data segment
    + UB2     023  UB2 - UB92 data segment
    + OBX     009  OBX - observation/result segment
    + PID     003  PID - patient identification segment
    + PD1     004  PD1 - patient additional demographi
    + PV1     006  PV1 - patient visit segment-
    + PV2     007  PV2 - patient visit - additional informa
    + DB1     008  DB1 - Disability segment
    + OBX     009  OBX - observation/result segment
```

Illustration 7: Additional segments for A17

The new segments only occur in A17s so let's make sure all are optional. The copied PID and PV1 are required. Select PID segment and change Required to Optional and Must Use to Used.



```
Guideline
HL7 Guideline
HL7 Data Types
V2.3.1 ADT - ADT message (event A01)
  MSH    001  MSH - message header segment
  EVN    002  EVN - event type segment
  PID    003  PID - patient identification segment
  PD1    004  PD1 - patient additional demographic
  NK1    005  NK1 - next of kin / associated partie
  PV1    006  PV1 - patient visit segment-
  PV2    007  PV2 - patient visit - additional informa
  DB1    008  DB1 - Disability segment
  OBX    009  OBX - observation/result segment
  AL1    010  AL1 - patient allergy information segr
  DG1    011  DG1 - diagnosis segment
  DRG    012  DRG - diagnosis related group segm
  Group  013 PROCEDURE
    PR1    014  PR1 - procedures segment
    ROL    015  Role
  GT1    016  GT1 - guarantor segment
  Group  017 INSURANCE
    IN1    018  IN1 - insurance segment
    IN2    019  IN2 - insurance additional informa
    IN3    020  IN3 - insurance additional informa
  ACC    021  ACC - accident segment
  UB1    022  UB1 - UB82 data segment
  UB2    023  UB2 - UB92 data segment
  OBX    009  OBX - observation/result segment
  PID    003  PID - patient identification segment
  PD1    004  PD1 - patient additional demographic
  PV1    006  PV1 - patient visit segment-
```

Segment Properties

General

ID: PID Position: 003

Name: PID - patient identification segment

Requirement: Optional Repeats: 1

User Option: Used

Notes

Segment
Purpose

Purpose

The PID segment is used by all applications as the primary means of communicating patient identification information. This segment contains permanent patient identifying and demographic information that, for the most part, is not likely to change frequently.

Illustration 8: Make the second PID Optional and Used

Repeat the process for the second PV1.

Let's now create the Z01 segment.

Right-click the last segment in the structure (second OBX). Choose Insert Node → Insert → Segment.

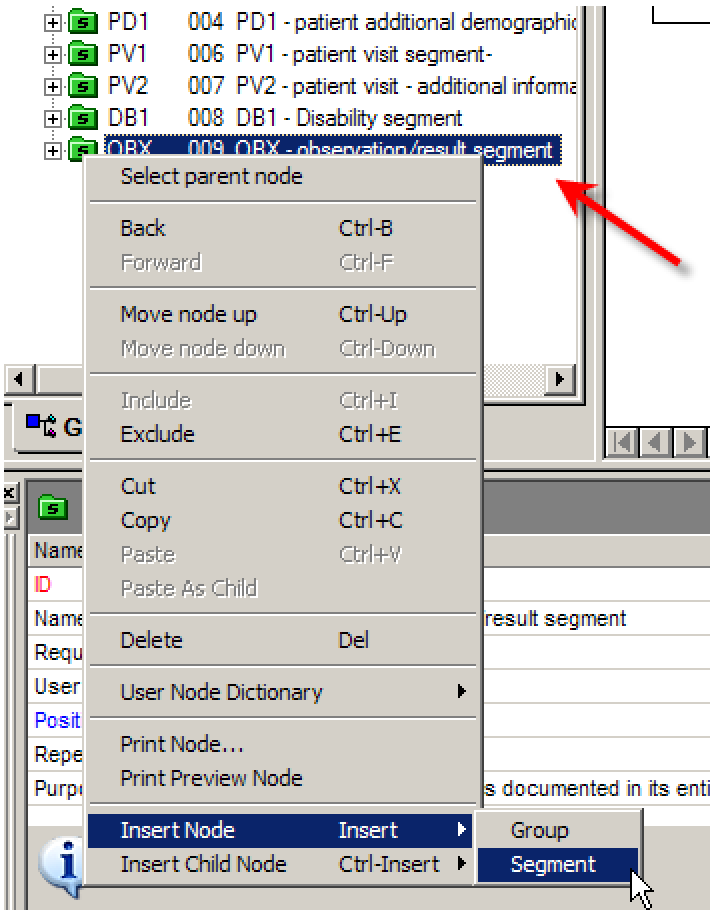


Illustration 9: Insert Segment

Choose “Create a new node” and click Next.

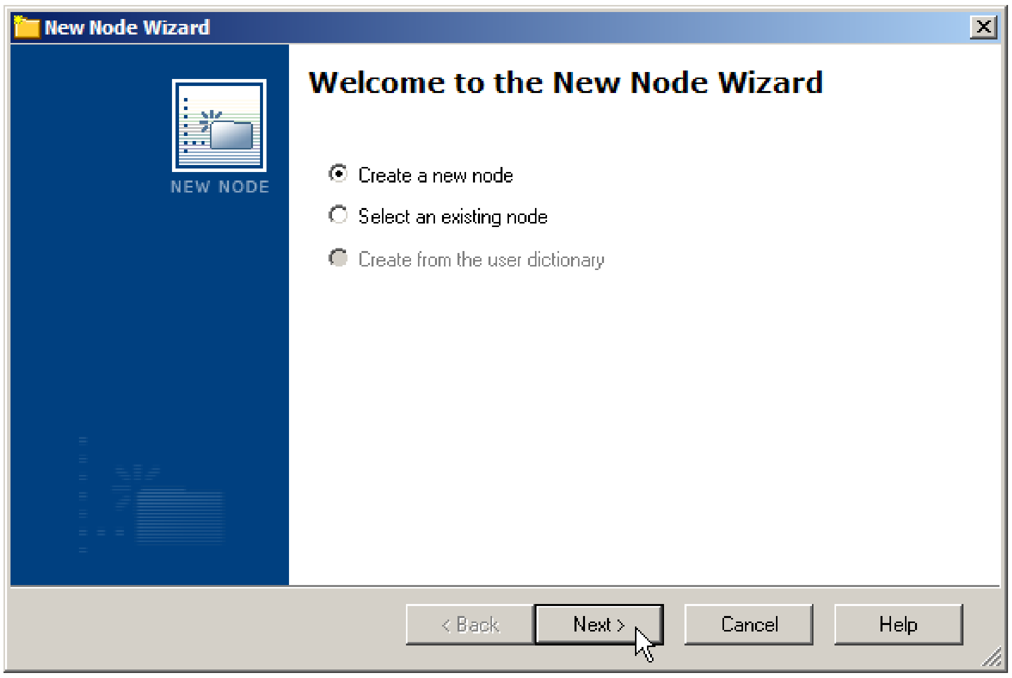
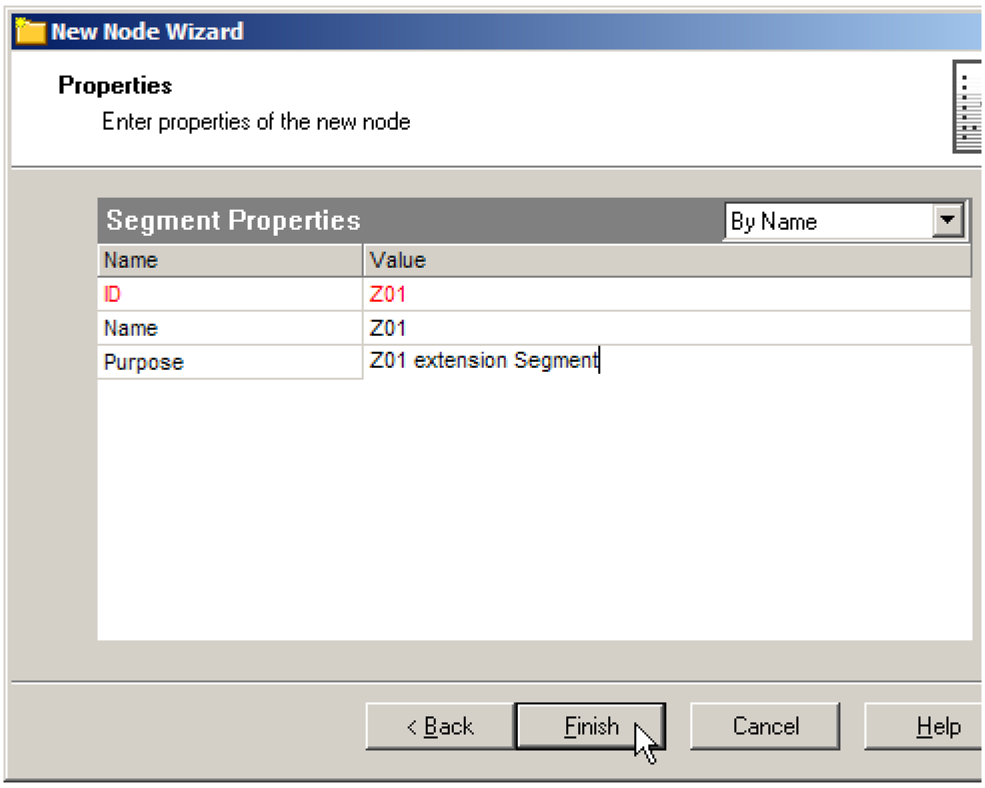


Illustration 10: Create a new node

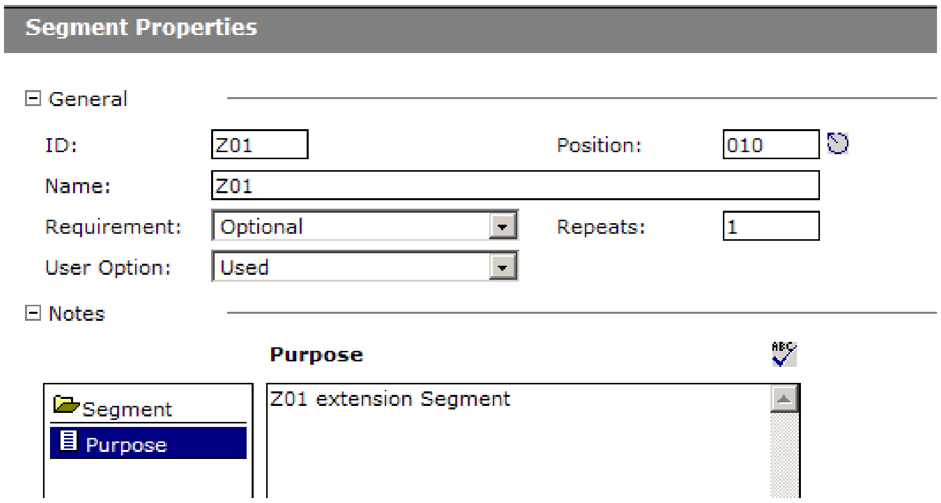
Enter ID: Z01, Name: Z01 and Purpose and click Finish.



Name	Value
ID	Z01
Name	Z01
Purpose	Z01 extension Segment

Illustration 11: Configure segment details

Change Requirement and User Option to Optional and Used. This makes the segment optional.



ID: Z01
Position: 010
Name: Z01
Requirement: Optional
Repeats: 1
User Option: Used

Purpose: Z01 extension Segment

Illustration 12: Make this segment optional

Recall the structure of the Z01 segment.

Sequence	*Length*	*Data Type*	*Element Name*
1	1	IS	Original Gender
2	1	IS	Current Gender
3	26	TS	Date of Change
4	1	IS	Legal Gender

We will add these components one at a time.

Right-click on the name of the segment and choose Insert Child Node → Field.

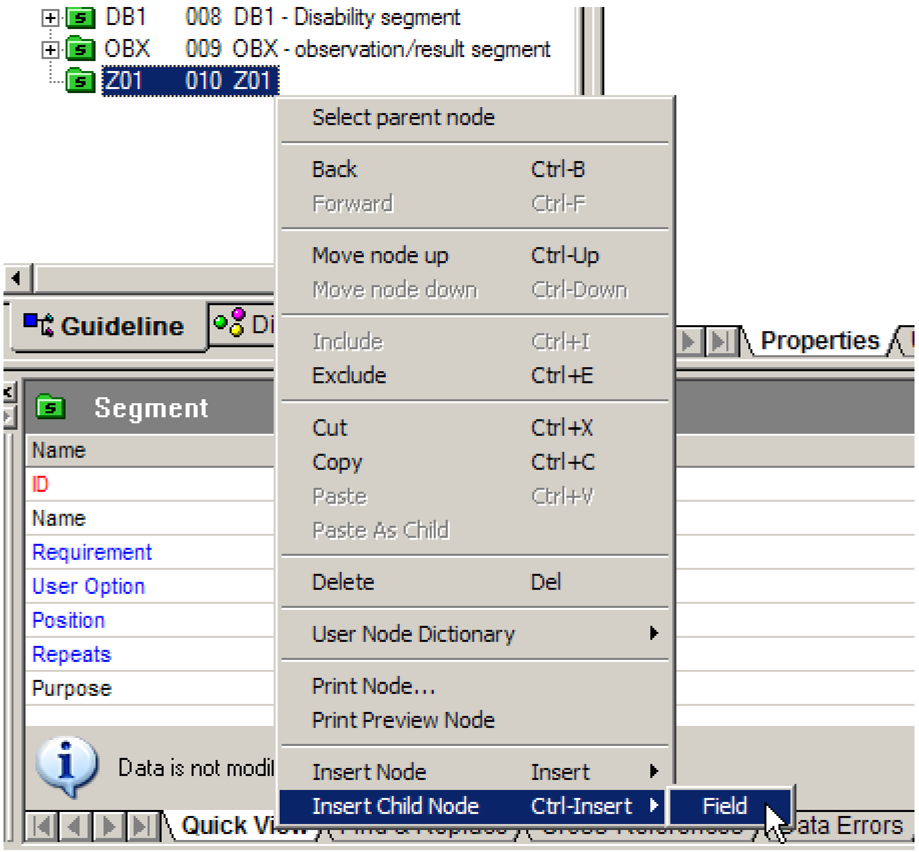
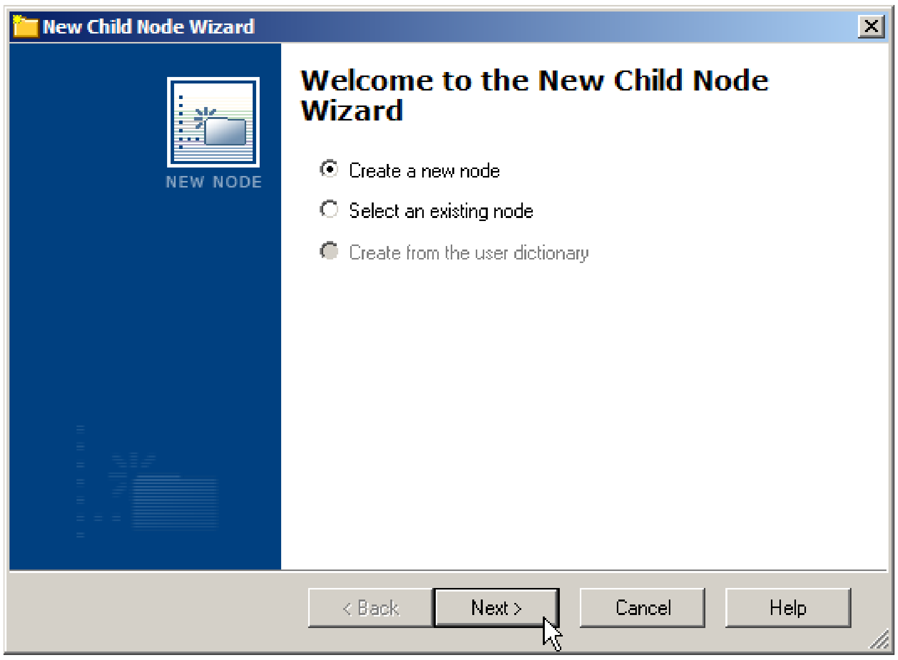


Illustration 13: Insert field as a child of the segment

Accept “create a new node” and click Next.

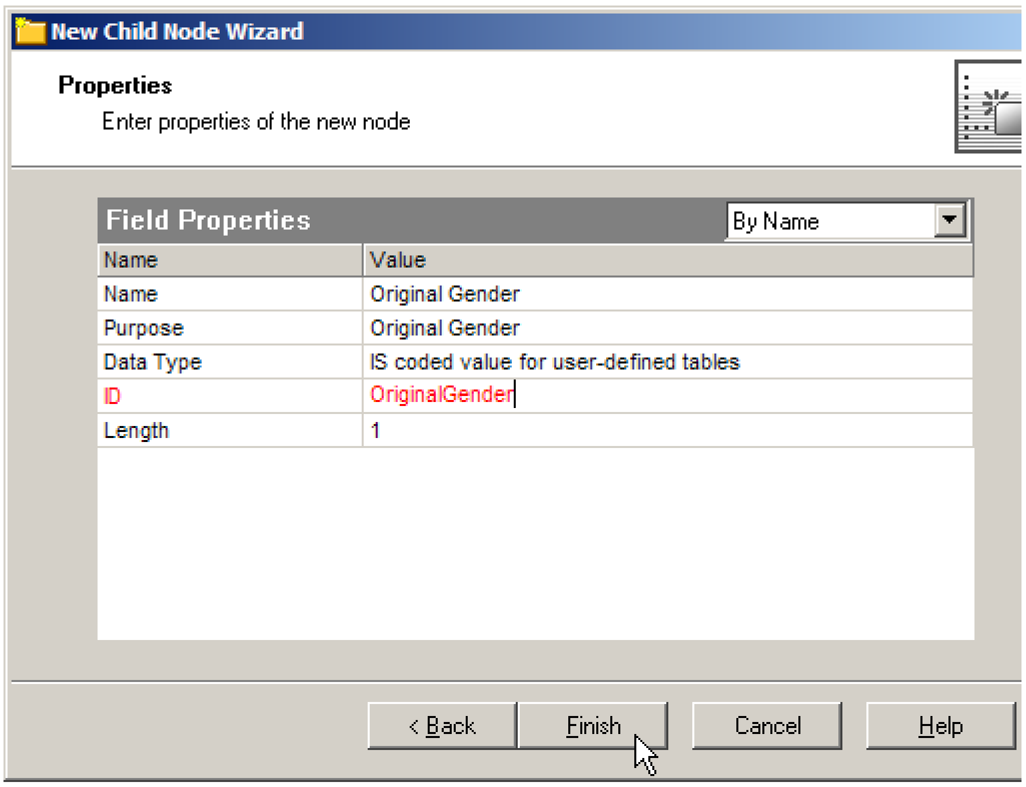

New Child Node Wizard

Welcome to the New Child Node Wizard

[✓ Create a new node]
Select an existing node
Create from the user dictionary

Illustration 14: Create a new node

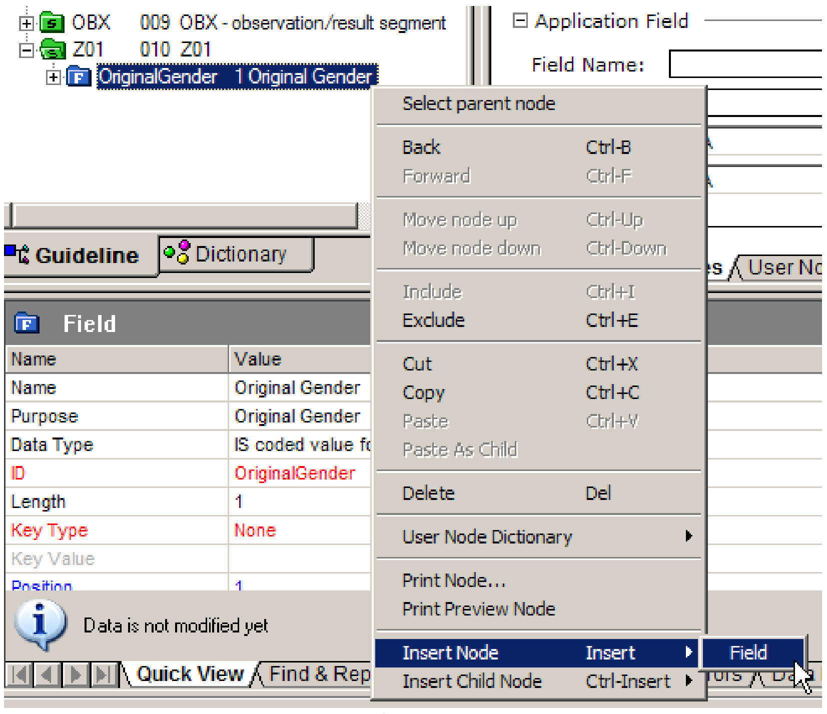
Provide values for Name, Purpose, Data Type (from a drop down menu), ID and Length and click Finish.



Name	Value
Name	Original Gender
Purpose	Original Gender
Data Type	IS coded value for user-defined tables
ID	OriginalGender
Length	1

Illustration 15: Defined Original Gender field

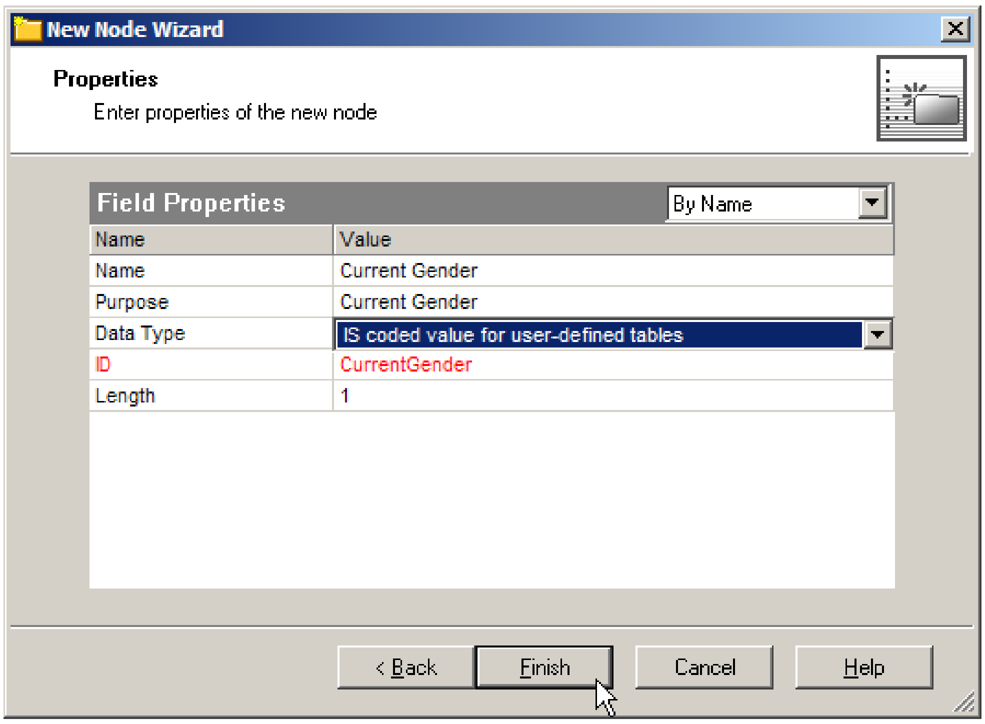
Right-click on the name of the new field and choose Insert Node → Field.



Name	Value
Name	Original Gender
Purpose	Original Gender
Data Type	IS coded value fo
ID	OriginalGender
Length	1
Key Type	None
Key Value	
Position	1

Illustration 16: Add another field

Provide values for Name: Current Gender, Purpose: Current Gender, Data Type: IS (from drop down menu), ID: CurrentGender and Length: 1. Click Finish.


New Node Wizard

Properties

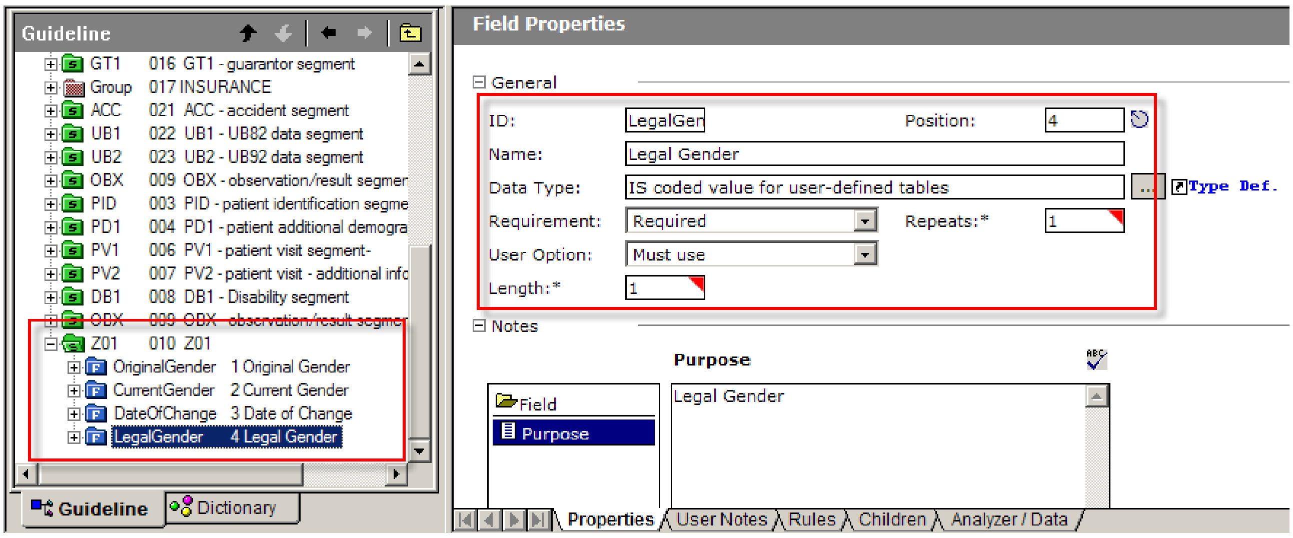
Enter properties of the new node

Field Properties — By Name

Name	Value
Name	Current Gender
Purpose	Current Gender
Data Type	IS coded value for user-defined tables
ID	CurrentGender
Length	1

Illustration 17: Configure Current Gender field

Repeat the process for the remaining two fields – Date of Change and Legal Gender.


Guideline

```
GT1          016 GT1 - guarantor segment
Group        017 INSURANCE
ACC          021 ACC - accident segment
UB1          022 UB1 - UB82 data segment
UB2          023 UB2 - UB92 data segment
OBX          009 OBX - observation/result segmer
PID          003 PID - patient identification segme
PD1          004 PD1 - patient additional demogra
PV1          006 PV1 - patient visit segment-
PV2          007 PV2 - patient visit - additional infc
DB1          008 DB1 - Disability segment
OBX          009 OBX - observation/result segme
Z01          010 Z01
  OriginalGender   1 Original Gender
  CurrentGender    2 Current Gender
  DateOfChange     3 Date of Change
  LegalGender      4 Legal Gender
```

Field Properties

General

ID:	LegalGen	Position:	4
Name:	Legal Gender		
Data Type:	IS coded value for user-defined tables		
Requirement:	Required	Repeats:*	1
User Option:	Must use		
Length:*	1		

Notes

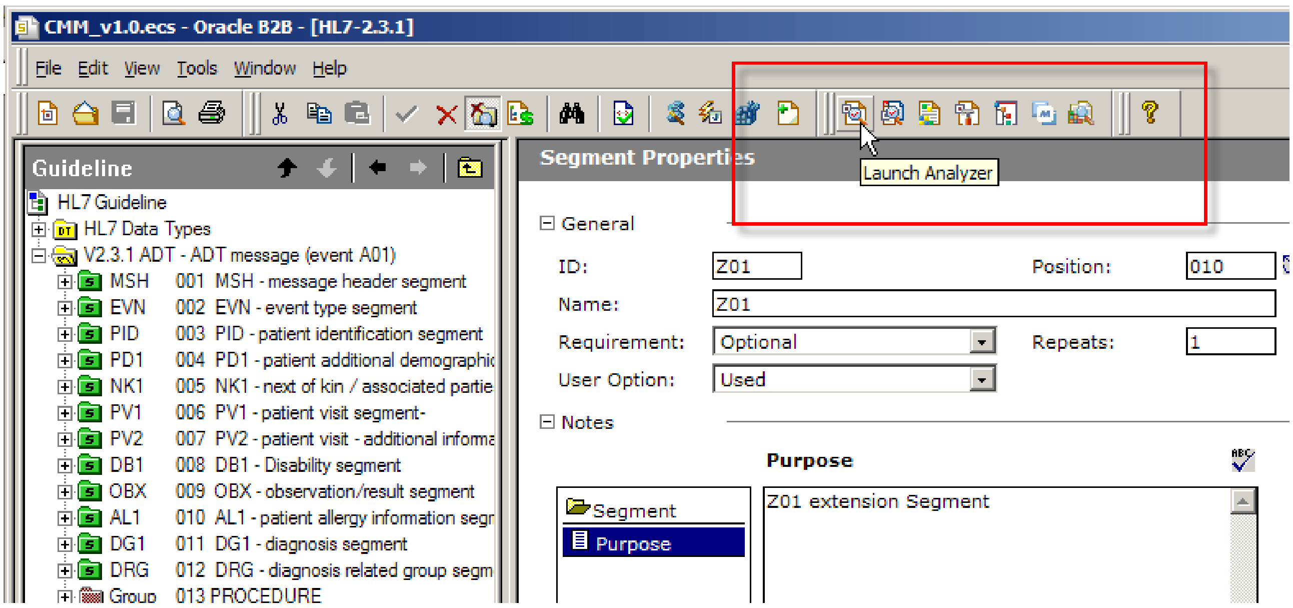
Purpose: Legal Gender

Illustration 18: Completed structure

Note that by default new fields are Required and Must Use. Modify as needed for your fields.

Save the structure.

This structure/guideline/spec is a customisation of the ADT A01 structure. The B2B Document Editor and associated tooling can be used to test how sample data fits this structure and validate data against it.



HL7 Guideline
HL7 Data Types
V2.3.1 ADT - ADT message (event A01)

MSH	001	MSH - message header segment
EVN	002	EVN - event type segment
PID	003	PID - patient identification segment
PD1	004	PD1 - patient additional demographic
NK1	005	NK1 - next of kin / associated partie
PV1	006	PV1 - patient visit segment-
PV2	007	PV2 - patient visit - additional informa
DB1	008	DB1 - Disability segment
OBX	009	OBX - observation/result segment
AL1	010	AL1 - patient allergy information segr
DG1	011	DG1 - diagnosis segment
DRG	012	DRG - diagnosis related group segm
Group	013	PROCEDURE

ID: Z01 Position: 010
Name: Z01
Requirement: Optional Repeats: 1
User Option: Used

Z01 extension Segment

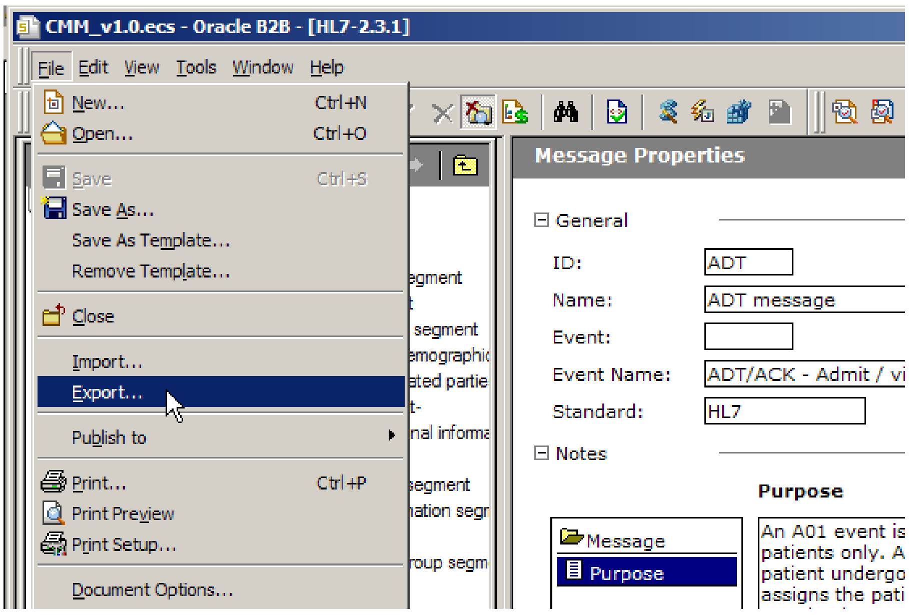
Illustration 19: Launch Analyzer

We can use a sample ADT A01 to see whether the data is valid according to the guideline. If data is not valid we can either modify data until it is valid or modify the guideline until the data is valid.

Since we create the custom structure starting with the ADT A01 other message types (trigger events) will fail trigger event validation even if all other spects of message validation succeed. Whether this is likely to be an issue depends on matters which are not related to the structure itself.

Once the structure is ready we can save the ECS file and export the XSD file. The former is used in Oracle SOA Suite B2B for message validation (optional) and conversion between HL7 v2 Delimited and XML (in either direction). The later is used for integration between the Oracle SOA Suite B2B and other parts of the SOA Suite.

Pull down the File menu and choose Export.

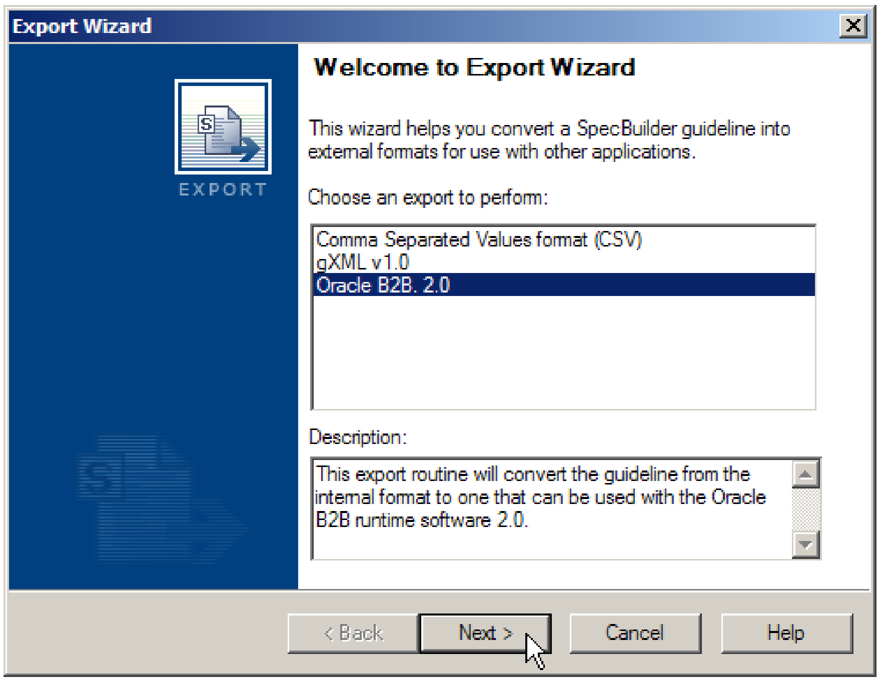


ID: ADT
Name: ADT message
Event:
Event Name: ADT/ACK - Admit / vi
Standard: HL7

An A01 event is patients only. A patient undergo assigns the pati

Illustration 20: Trigger Export Wizard

Choose Oracle B2B 2.0 and click Next.



Welcome to Export Wizard

This wizard helps you convert a SpecBuilder guideline into external formats for use with other applications.

Choose an export to perform:

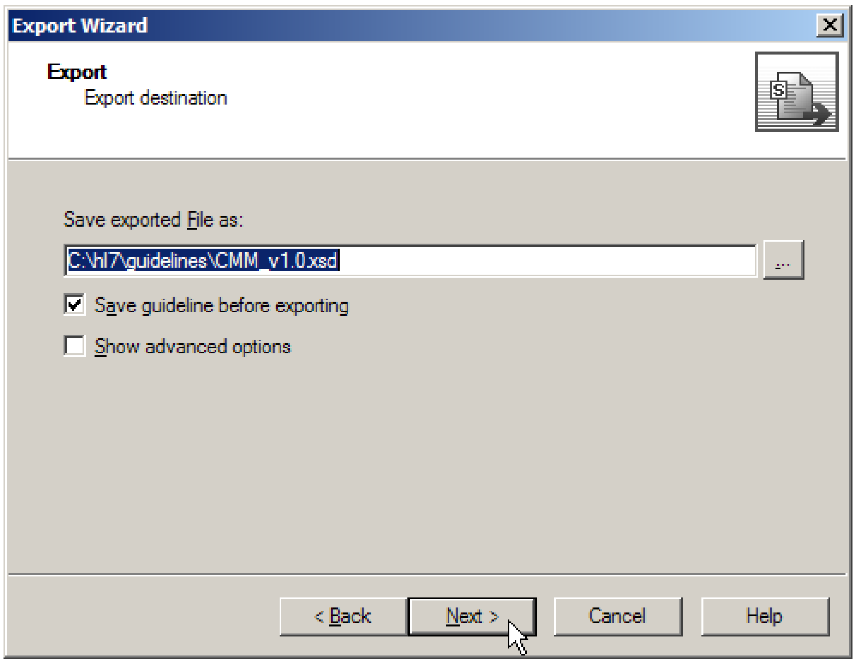
Comma Separated Values format (CSV)
gXML v1.0
Oracle B2B. 2.0

Description:

This export routine will convert the guideline from the internal format to one that can be used with the Oracle B2B runtime software 2.0.

Illustration 21: Coose to export to Oracle format

Check "Save guideline before exporting", change the name to an appropriate name and click Next.



Export
Export destination

Save exported File as:

C:\hl7\guidelines\CMM_v1.0xsd

[✓] Save guideline before exporting

[ ] Show advanced options

Illustration 22: Name the file and choose to save guideline

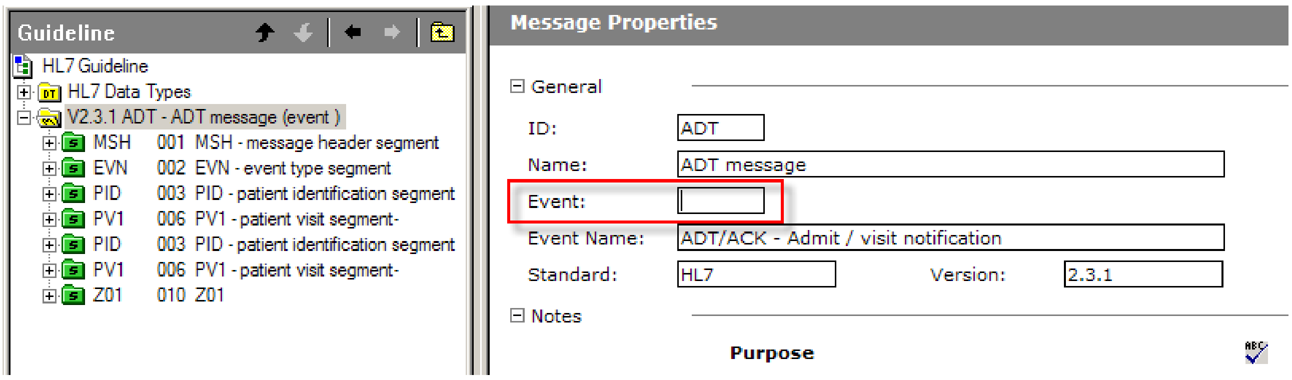
Two new files will appear in the file system – CMM_v1.0.ecs (EDIFECS Guideline) and CMM_v1.0xsd (XML Schema Document).

Customise the CMM

The canonical message model structure which we created includes all segments from the A01/A08 message and all extra segments from 17 and the Z segment. We stated earlier that our messages will only ever use MSH, EVN, PID, PV1 and Z01 segments. Let's delete all other segments and save/export the structure as CMM_v1.1.

Open the CMM_v1.0.ecs file and delete all segments except the ones mentioned, by right-clicking the segment and choosing Delete. Once done save and export the modified files as CMM_v1.1.

While at it, click the root node of the structure and clear the Event field to eliminate Event Type dependency.



HL7 Guideline
HL7 Data Types
V2.3.1 ADT - ADT message (event)
MSH 001 MSH - message header segment
EVN 002 EVN - event type segment
PID 003 PID - patient identification segment
PV1 006 PV1 - patient visit segment-
PID 003 PID - patient identification segment
PV1 006 PV1 - patient visit segment-
Z01 010 Z01

ID:	ADT		
Name:	ADT message		
Event:			
Event Name:	ADT/ACK - Admit / visit notification		
Standard:	HL7	Version:	2.3.1

Illustration 23: Minimalist CMM

Add CMM Metadata

We discussed the need to version the canonical message model and include metadata which will allow integration components to deal with changes over time.

An Enveloped Message EAI Pattern would call for a new message structure in which the original message would be a payload node. All other nodes would be used for metadata of various uses.

An Enveloping Message Pattern, which I just invented for this article :-), would inject metadata into the original message.

Of the two external forms of the canonical message structure, ecs and xsd, which we created, the ecs file is used by the edge infrastructure to convert from HL7 v2 Delimited to XML or the other way around, depending on whether the messages are inbound or outbound. This may make the ecs form of the structure tightly bound to the specific endpoint. The endpoint can only ever use one version of the canonical message at a time. The version, if embedded in the name of the ecs file, is reflected in the configuration of the endpoint.

The xml schema version of the ecs file is generated during the export from the B2B Document Editor.

On the inbound side, messages conforming to this schema are handed by the inbound B2B channel to the appropriate SOA Suite composite for further processing. This structure includes both the XML version of the HL7 v2 Delimited message, for example ADT A01, and B2B-generated metadata. The sample, abbreviated XML message for the canonical message model we were developing so far, is shown below.

```
<ADT_
      xmlns="NS_6E95A90E4BAF43AB9F9A3CBB863278FC20070423183755"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      XDataVersion="2.0"
      Standard="HL7"
      Version="2.3.1"
      CreatedDate="2010-10-08T14:11:53"
      CreatedBy="XEngine_2114"
      GUID="{DBFA5449-DC41-4969-8833-089882691395}">
   <Internal-Properties>
      <Data-Structure Name="Message">
         <Lookup Name="InternatCodeAlternateID"/>
...
         <Lookup Name="MessageCode">ADT</Lookup>
         <Lookup Name="MessageReceivingApp">PI</Lookup>
...
         <Lookup Name="MessageReceivingFacility">MDM</Lookup>
...
         <Lookup Name="MessageSendingApp">SystemA</Lookup>
...
         <Lookup Name="MessageSendingFacility">HosA</Lookup>
         <Lookup Name="MessageSendingFacilityUniversalID"/>
         <Lookup Name="MessageSendingFacilityUniversalIDType"/>
         <Lookup Name="MessageVersion">2.3.1</Lookup>
         <Lookup Name="Standard">HL7</Lookup>
         <Lookup Name="TriggerEvent">A01</Lookup>
         <Property Name="AcceptAckType">AL</Property>
         <Property Name="AlternateCharacterSetSchema"/>
         <Property Name="AppAckType">NE</Property>
...
         <Property Name="MessageCode">ADT</Property>
         <Property Name="MessageControlID">000000_CTLID_2008090801529</Property>
         <Property Name="MessageDate">2008090801529</Property>
         <Property Name="MessageDateTimePrecision"/>
         <Property Name="MessageEncodingCharacters">^~\&amp;</Property>
...
         <Property Name="MessageVersion">2.3.1</Property>
         <Property Name="ProcessingID">P</Property>
...
```

```
         <Property Name="ReleaseCharacter">0x5c</Property>
         <Property Name="RepeatingSeparator">0x7e</Property>
         <Property Name="SegmentDelimiter">0x0d</Property>
...
         <Property Name="Standard">HL7</Property>
         <Property Name="SubcomponentDelimiter">0x26</Property>
         <Property Name="SubelementDelimiter">0x5e</Property>
         <Property Name="TriggerEvent">A01</Property>
      </Data-Structure>
   </Internal-Properties>
   <MSH>
      <MSH.1>|</MSH.1>
      <MSH.2>^~\&amp;</MSH.2>
      <MSH.3>
         <HD.1>SystemA</HD.1>
      </MSH.3>
      <MSH.4>
...
</ADT_>
```

The Internal-Properties structure, populated by the B2B infrastructure, carries information gleaned from the inbound message, including selected fields from the MSH segment and from the MLLP configuration. Values of these properties can be used in message processing, if needed, though the MSH segment is available as part of the message anyway and the MLLP configuration seems unlikely to be of use in message processing. Be it as it may, these properties are there.

Please note the attribute “Standard”, highlighted in bold in the sample, with the value of “HL7”. This attribute is mandatory in an outbound message to B2B.

If the XML Schema document was modified by addition of optional elements outside the actual HL7 structure the B2B-generated XML instance document, which would not contain these element, would still be valid.

On the outbound side, messages conforming to this schema may be handed over by the SOA Suite composite to the B2B infrastructure for conversion to HL7 v2 Delimited format and sending on to the partner. It should be pointed out that the structure to be handed over to B2B does not have to have the Internal-Properties tree. The message may well be a standard XML version of a HL7 v2 message, conforming to the generally available HL7 v2 XML Schemas. Whether the Internal_Properties structure is present or absent the B2B processes the message the same way and requires certain B2B-related properties to be configured. This was discussed in earlier blog articles, for example the “Oracle SOA Suite 11g HL7 Outbound Example”, at http://blogs.czapski.id.au/2010/06/oracle-soa-suite-11g-hl7-outbound-example.

The one departure from this is the mandatory attribute “Standard”, which must be present, valued and containing the literal string “HL7”, as noted a couple of paragraphs ago. If one uses the Oracle B2B Document Editor-generated XSD this attribute is present and can be populated. If one uses standard HL7 v2 XML Schemas the specific schema document must be modifed through addition of this attribute and it must be populated at runtime before an MXL instance document is passed to the B2B infrastructure.

This “insensitivity” of the Oracle B2B HL7 infrastructure the the presence or absence of optional elements outside the HL7 structure makes it possible to construct an Enveloping Message, with metadata injected into the message without invalidating it.

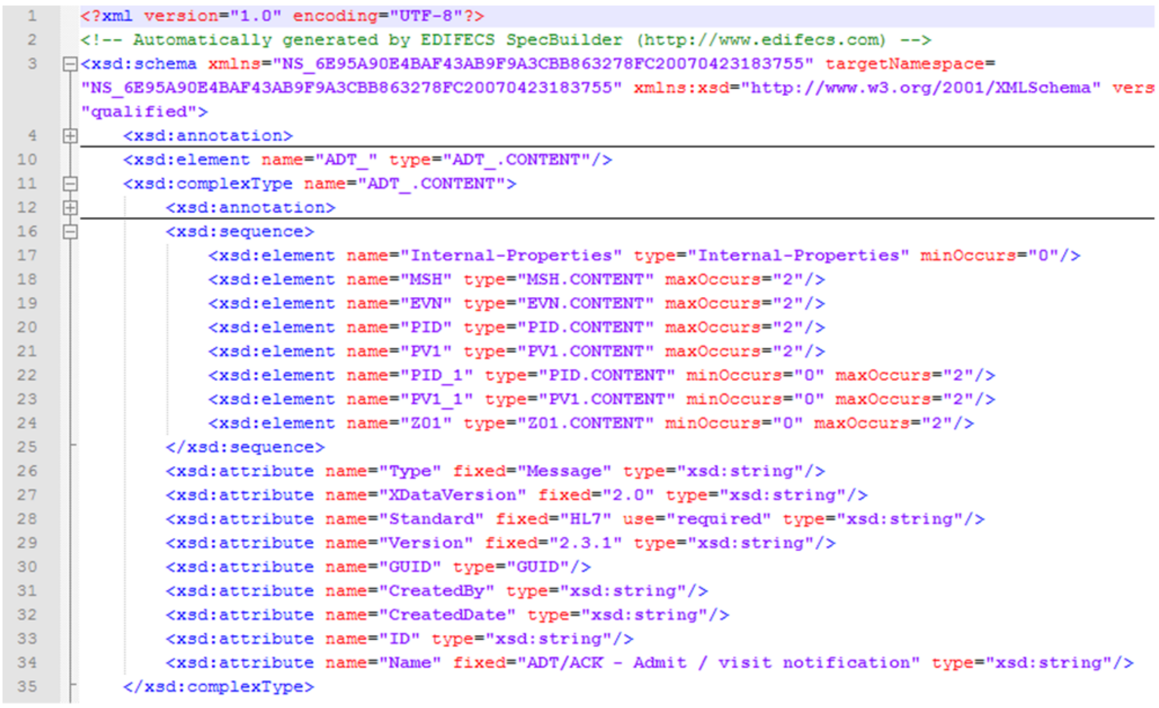
Let's consider a sample message shown below.

```
<?xml version="1.0" encoding="UTF-8" ?>
<ns1:ADT_
       xmlns:ns1="NS_6E95A90E4BAF43AB9F9A3CBB863278FC20070423183755"
       Type=""
       XDataVersion="2.0"
       Standard="HL7"
       Version="2.3.1"
```

```
        GUID="{DBFA5449-DC41-4969-8833-089882691395}"
        CreatedBy="XEngine_2114"
        CreatedDate="2010-10-08T14:11:53"
        ID=""
        Name=""
        xmlns="NS_6E95A90E4BAF43AB9F9A3CBB863278FC20070423183755">
   <ns1:CMMMetaData
          CMMVersion="1.2"
          CreatedDate="2010-10-08T14:11:53+11:00"
          ID="333530373739373937333136353336335">
      <ns1:RouteDelimiter>,</ns1:RouteDelimiter>
      <ns1:RouteHops>CMM_02_Mediator,</ns1:RouteHops>
      <ns1:RouteArrivals>2010-10-08T14:11:53+11:00,</ns1:RouteArrivals>
   </ns1:CMMMetaData>
   <Internal-Properties xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <Data-Structure Name="Message">
         <Lookup Name="InternatCodeAlternateID"/>
...
         <Lookup Name="TriggerEvent">A01</Lookup>
         <Property Name="AcceptAckType">AL</Property>
...
         <Property Name="TriggerEvent">A01</Property>
      </Data-Structure>
   </Internal-Properties>
   <MSH xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <MSH.1>|</MSH.1>
      <MSH.2>^~\&amp;</MSH.2>
      <MSH.3>
         <HD.1>SystemA</HD.1>
      </MSH.3>
      <MSH.4>
         <HD.1>HosA</HD.1>
      </MSH.4>
...
      </PV1.44>
   </PV1>
</ns1:ADT_>
```

This sample message carried a structure, CMMMetaData, which is not generated by the B2B infrastructure for inbound messages and which is ignored by the B2B infrastructure for the outbound messages. The name and the structure of CMMMetaData is completely arbitrary.

Fragment of the original XSD, generated by the B2B Document Editor is shown below.

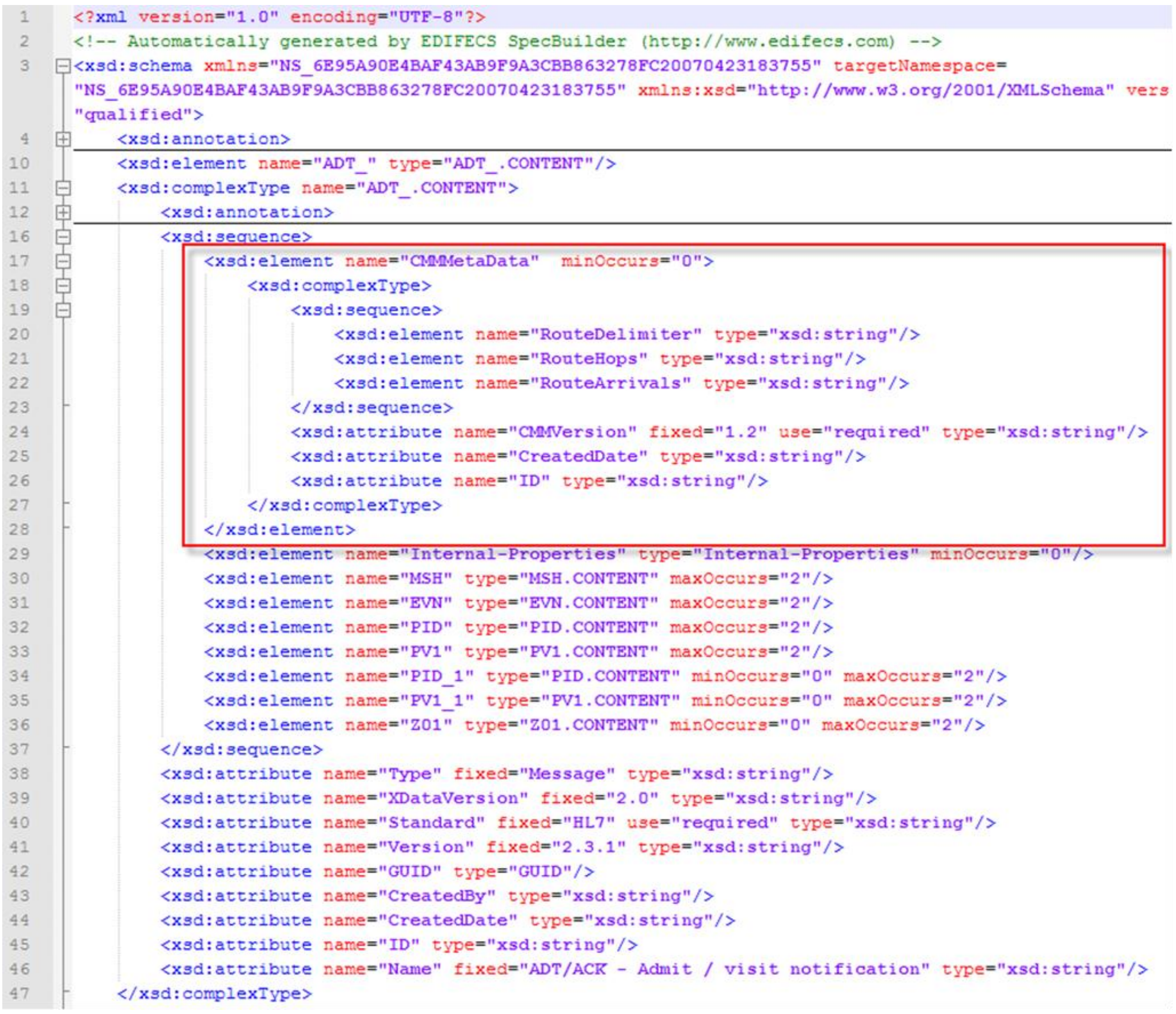


```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- Automatically generated by EDIFECS SpecBuilder (http://www.edifecs.com) -->
<xsd:schema xmlns="NS_6E95A90E4BAF43AB9F9A3CBB863278FC20070423183755" targetNamespace=
"NS_6E95A90E4BAF43AB9F9A3CBB863278FC20070423183755" xmlns:xsd="http://www.w3.org/2001/XMLSchema" vers
"qualified">
    <xsd:annotation>
    <xsd:element name="ADT_" type="ADT_.CONTENT"/>
    <xsd:complexType name="ADT_.CONTENT">
        <xsd:annotation>
        <xsd:sequence>
            <xsd:element name="Internal-Properties" type="Internal-Properties" minOccurs="0"/>
            <xsd:element name="MSH" type="MSH.CONTENT" maxOccurs="2"/>
            <xsd:element name="EVN" type="EVN.CONTENT" maxOccurs="2"/>
            <xsd:element name="PID" type="PID.CONTENT" maxOccurs="2"/>
            <xsd:element name="PV1" type="PV1.CONTENT" maxOccurs="2"/>
            <xsd:element name="PID_1" type="PID.CONTENT" minOccurs="0" maxOccurs="2"/>
            <xsd:element name="PV1_1" type="PV1.CONTENT" minOccurs="0" maxOccurs="2"/>
            <xsd:element name="Z01" type="Z01.CONTENT" minOccurs="0" maxOccurs="2"/>
        </xsd:sequence>
        <xsd:attribute name="Type" fixed="Message" type="xsd:string"/>
        <xsd:attribute name="XDataVersion" fixed="2.0" type="xsd:string"/>
        <xsd:attribute name="Standard" fixed="HL7" use="required" type="xsd:string"/>
        <xsd:attribute name="Version" fixed="2.3.1" type="xsd:string"/>
        <xsd:attribute name="GUID" type="GUID"/>
        <xsd:attribute name="CreatedBy" type="xsd:string"/>
        <xsd:attribute name="CreatedDate" type="xsd:string"/>
        <xsd:attribute name="ID" type="xsd:string"/>
        <xsd:attribute name="Name" fixed="ADT/ACK - Admit / visit notification" type="xsd:string"/>
    </xsd:complexType>
```

Illustration 24: B2B DOcument Editor-generated XSD - fragment

One can easily add an arbitrarily structured element to carry the Canonical Message Model-related metadata. One can also exploit this to add other arbitrary metadata structures, for example capturing the route which the message follows through the SOA infrastructure, capturing timings and performance collection-related data, and whatever else the architecture calls for, and what is not a part of the message.

An XSD fragment, with CMMMetaData structure corresponding to the data shown earlier, might look like that in the figure below.



```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Automatically generated by EDIFECS SpecBuilder (http://www.edifecs.com) -->
<xsd:schema xmlns="NS_6E95A90E4BAF43AB9F9A3CBB863278FC20070423183755" targetNamespace=
"NS_6E95A90E4BAF43AB9F9A3CBB863278FC20070423183755" xmlns:xsd="http://www.w3.org/2001/XMLSchema" vers
"qualified">
    <xsd:annotation>
    <xsd:element name="ADT_" type="ADT_.CONTENT"/>
    <xsd:complexType name="ADT_.CONTENT">
        <xsd:annotation>
        <xsd:sequence>
            <xsd:element name="CMMMetaData"  minOccurs="0">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="RouteDelimiter" type="xsd:string"/>
                        <xsd:element name="RouteHops" type="xsd:string"/>
                        <xsd:element name="RouteArrivals" type="xsd:string"/>
                    </xsd:sequence>
                    <xsd:attribute name="CMMVersion" fixed="1.2" use="required" type="xsd:string"/>
                    <xsd:attribute name="CreatedDate" type="xsd:string"/>
                    <xsd:attribute name="ID" type="xsd:string"/>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="Internal-Properties" type="Internal-Properties" minOccurs="0"/>
            <xsd:element name="MSH" type="MSH.CONTENT" maxOccurs="2"/>
            <xsd:element name="EVN" type="EVN.CONTENT" maxOccurs="2"/>
            <xsd:element name="PID" type="PID.CONTENT" maxOccurs="2"/>
            <xsd:element name="PV1" type="PV1.CONTENT" maxOccurs="2"/>
            <xsd:element name="PID_1" type="PID.CONTENT" minOccurs="0" maxOccurs="2"/>
            <xsd:element name="PV1_1" type="PV1.CONTENT" minOccurs="0" maxOccurs="2"/>
            <xsd:element name="Z01" type="Z01.CONTENT" minOccurs="0" maxOccurs="2"/>
        </xsd:sequence>
        <xsd:attribute name="Type" fixed="Message" type="xsd:string"/>
        <xsd:attribute name="XDataVersion" fixed="2.0" type="xsd:string"/>
        <xsd:attribute name="Standard" fixed="HL7" use="required" type="xsd:string"/>
        <xsd:attribute name="Version" fixed="2.3.1" type="xsd:string"/>
        <xsd:attribute name="GUID" type="GUID"/>
        <xsd:attribute name="CreatedBy" type="xsd:string"/>
        <xsd:attribute name="CreatedDate" type="xsd:string"/>
        <xsd:attribute name="ID" type="xsd:string"/>
        <xsd:attribute name="Name" fixed="ADT/ACK - Admit / visit notification" type="xsd:string"/>
    </xsd:complexType>
```

Illustration 25: CMMMetaData structure added to the XSD

As mentioned, the B2B inbound and outbound don't care whether this structure is present or absent. Because of this we can use it to carry arbitrary metadata as part of the Enveloping Message and use it for CMM tracking and any other purposes required by the enterprise architecture.

Summary

In any but the simplest of HL7 messaging environments there will be multiple sources and multiple destinations of HL7 messages. It is very unlikely that all, or even a majority of these, will use exactly the same HL7 message structures in terms of versions, optional/mandatory segments, extension Z segments, and so on. A sensible approach to dealing with these kinds of issues, and a key component of the HL7 Enterprise Architecture, is the so called Canonical (or Common) Message Model (CMM). The CMM works hand-in-glove with the enterprise architecture in which transformation to/from the CMM is performed at the edges of the integration domain.

This article discussed major considerations and worked through the mechanics of deriving a Canonical Message Model for a fictitious Healthcare Enterprise and implementing it using the Oracle SOA Suite 11g HL7 tooling as an example.

The article also discussed and illustrated a mechanism for injecting arbitrary metadata into the canonical message, generated by the B2B Document Editor, in such a way that it is ignored by the Edge-dwelling B2B infrastructure but is significant to the SOA infrastructure.

The externalised Canonical Message Model forms, the ECS and the XSD flies are now available for use in the Oracle SOA Suite B2B HL7-based solutions.