HL7 International

ANSI/HL7 V2.9-2019

12/9/2019

2B. Control: Conformance

Chapter Chair	Frank Oemig Deutsche Telekom Healthcare Security Solutions GmbH
Chapter Chair	Robert Snelick National Institute of Standards and Technology (NIST)
Chapter Chair	Ioana Singureanu
Chapter Chair	Nathan Bunker American Immunization Registry Assoication (AIRA)
Sponsoring Work Group:	Conformance
List Server:	cgit@lists.hl7.org

2B.1 CHAPTER 2B CONTENTS

2B.2 PURPOSE

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 of a single interaction 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.

A conformance statement is a claim that the behavior of an application or application module agrees with the constraints stated in one or more message profiles. This section defines the message profile; however, the conformance statement will not be discussed further in this document.

An implementation guide is often created to organize a collection of message profiles for specifying a set of related HL7 V2.x interactions described in a use case. Implementation guides typically describe broader conformance requirements such as application behavior. Such requirements may include how a set of messages are to be used to enact certain application functionality. Implementation guides, which have broad scope have been introduced in this section to provide context of message profiles and will not be discussed further in this document. A message profile provides a mechanism for specifying a single message definition.

Definition: An HL7 message profile is an unambiguous specification of one standard HL7 message that has been analyzed for a particular interaction. It prescribes a set of precise constraints upon this HL7 message.

An HL7 message profile is compliant, in all aspects, with the HL7 defined message(s) used in the profile. It may specify constraints on the standard HL7 message definition.

A message profile fully describes a conversation between two or more systems through the combination of the following:

a) one interaction analysis,

b) one or more dynamic definitions,

c) one static definition, and

d) one table (vocabulary) definition.

The *interaction analysis* may be documented as a sequence diagram (supported with text) or just a textual description (See section *2.B.2*, "*Interaction Definition*".)

The dynamic definition is an interaction specification for a conversation between 2 or more systems (See Section *2.B.3*, "*Dynamic definition*".)

The static definition is a specification that defines the constraints for a single message structure (see Section *2.B.4*, "*Static definition*").

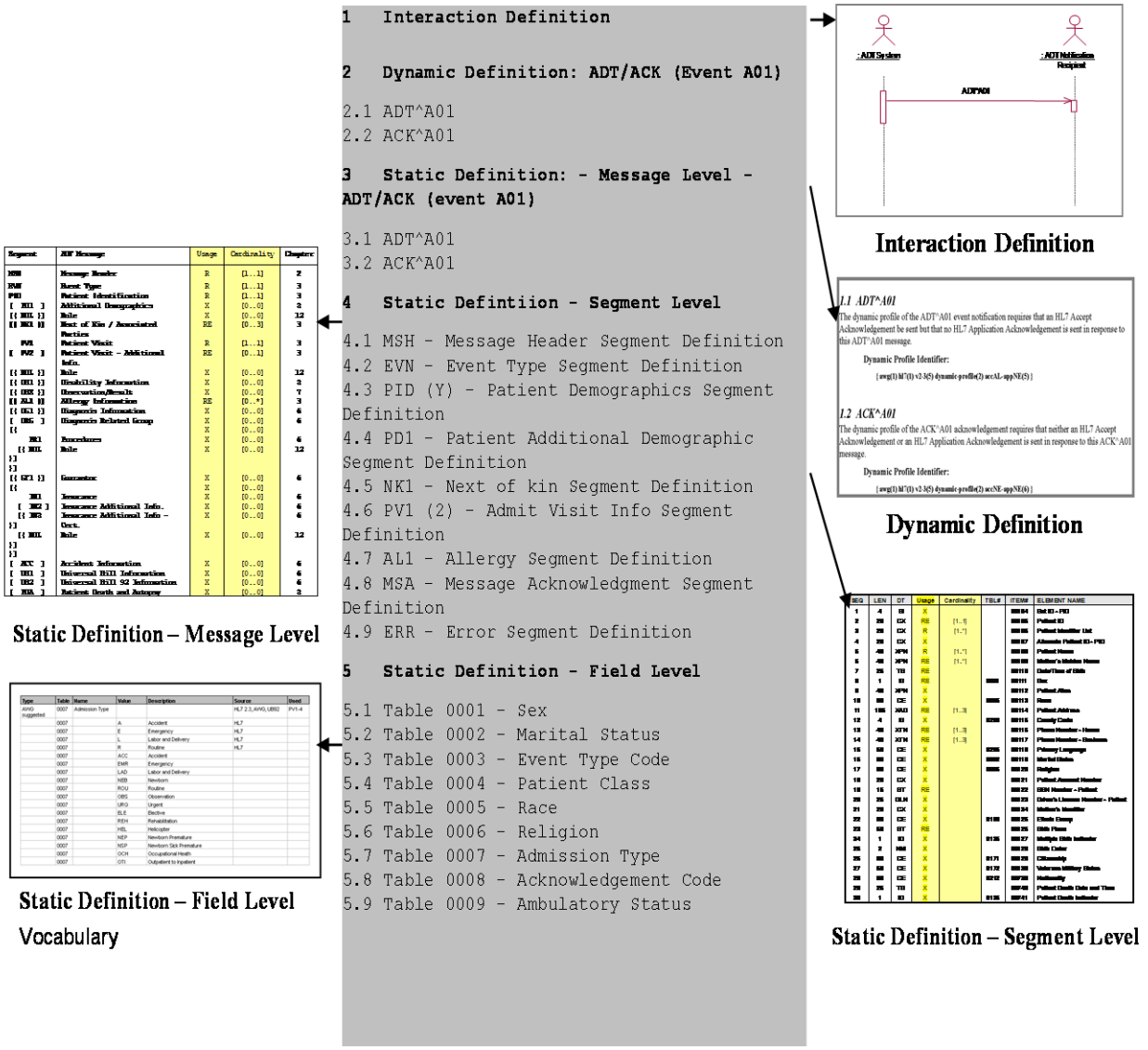
The table (vocabulary) definition is a specification of the tables referenced in the static definition. (see Section *2.B.5*, "*Table definition*").

The message profile is normatively expressed as an XML document validated against the normative message profile Schema, which may be registered on the HL7 web site (see Section *2.B.10*, "*Message profile document*"). The normative table definition can partly or wholly be contained in the message profile XML document. The table definition can also be partly or wholly defined in a table library. The table library is normatively expressed as an XML document validated against the normative table library Schema, which may be registered on the HL7 web site (see Section *2.B.10*, "*Message profile document*").

For detailed background information regarding message profiles, the reader is referred to the Implementation/Conformance Work Group balloted informative document, "Message Profiling Specification, Version 2.2", published November 30, 2000, upon which this section is based. This document is available from the HL7 Resources Web site (*http://www.hl7.org*).

A sample message profile is shown on the next page to assist in illustrating the constituents of a message profile and how they work together.

Message Profile Example



2B.1.1 Message profile

Definition: An HL7 message profile is an unambiguous specification of one standard HL7 message that has been analyzed for a particular interaction. Each message profile may have a unique identifier as well as publish/subscribe topics.

2B.2.1 Message profile identifier

Each message profile may have a unique identifier to facilitate reference.

2B.2.2 Message profile publish/subscribe topics

The message profile publish/subscribe topics is not required to be unique but might be used by publish/subscribe systems to convey aspects of the message profile (see *MSH-21 Message Profile Identifier* in the opening section of chapter 2).

The topics are not a normative constituent of the message profile but, if provided as part of the metadata, should be in the format described below. The topic elements will be separated by the dash (-). Any element that does not have a value should use null. As this information may be used in a message instance; it should not contain any HL7 message delimiters.

Message Profile Publish/Subscribe Topics Elements

Seq	Topic Element Name	Value
1	Implementation/Conformance Work Group ID	Confsig
2	An organization identifier	Abbreviated version of the organization name
3	The HL7 version	Refer to *HL7 Table 0104 - Version ID* for valid values
4	Topic Type	Profile
5	Accept Acknowledgement	The accept acknowledgement responsibilities.(refer to *HL7 Table 0155 – Accept/application Acknowledgment Conditions* for valid values)
6	Application Acknowledgement	The application acknowledgement responsibilities (refer to *HL7 Table 0155 – Accept/application Acknowledgment Conditions* for valid values)
7	Acknowledgement Mode	Deferred or Immediate

An example of message profile publish/subscribe topics:

confSig-MyOrganization-2.4-profile-AL-NE-Immediate

2B.3 INTERACTION DEFINITION

Definition: An interaction definition documents the scope and requirements for an HL7 message profile.

The interaction definition must:

a) Provide a name that clearly and concisely defines the exchange

b) Document the purpose for the message exchange

c) Define the actors, including the sending and receiving applications

d) Define the flow of events between these actors including, where appropriate, derived events

e) Document the situations in which the exchange of a particular HL7 message profile is required

2B.4 DYNAMIC DEFINITION

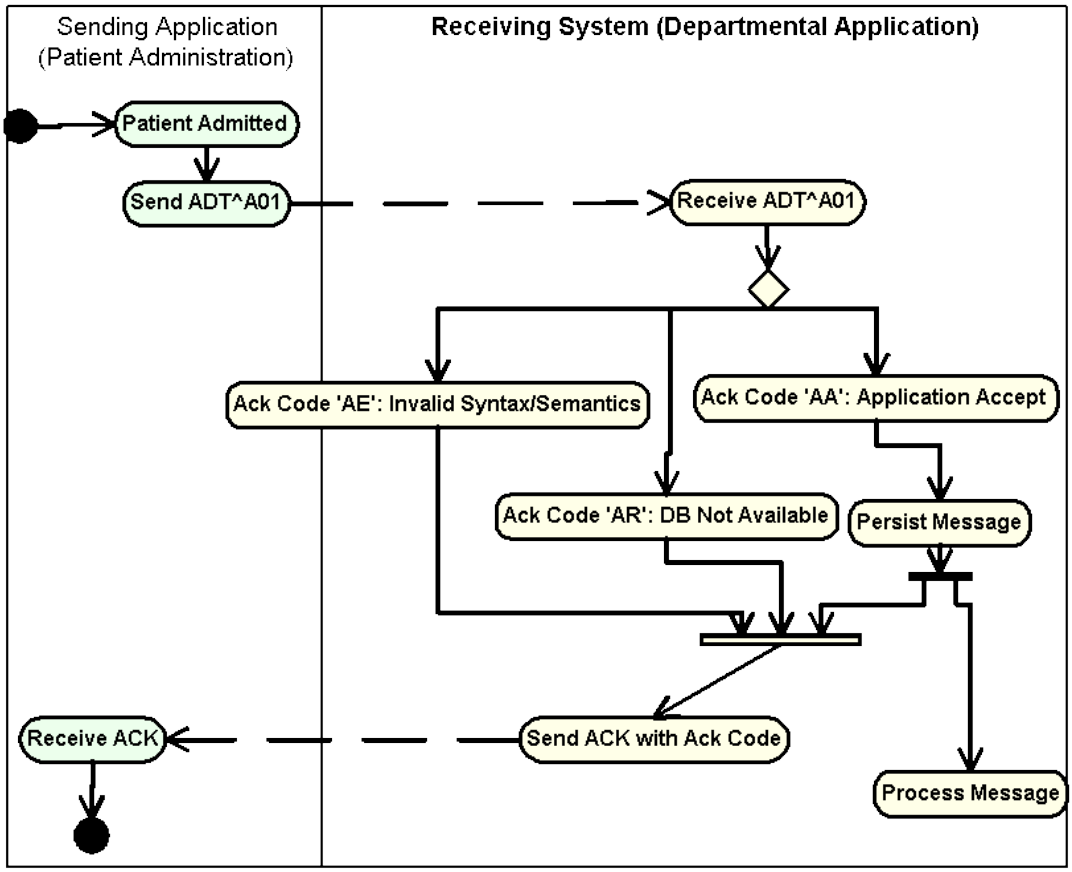
Definition: The dynamic definition is an interaction specification for a conversation between 2 or more systems. It may reference one static definition. The dynamic definition may include an interaction model in addition to the acknowledgement responsibilities.

2B.4.1 Interaction model

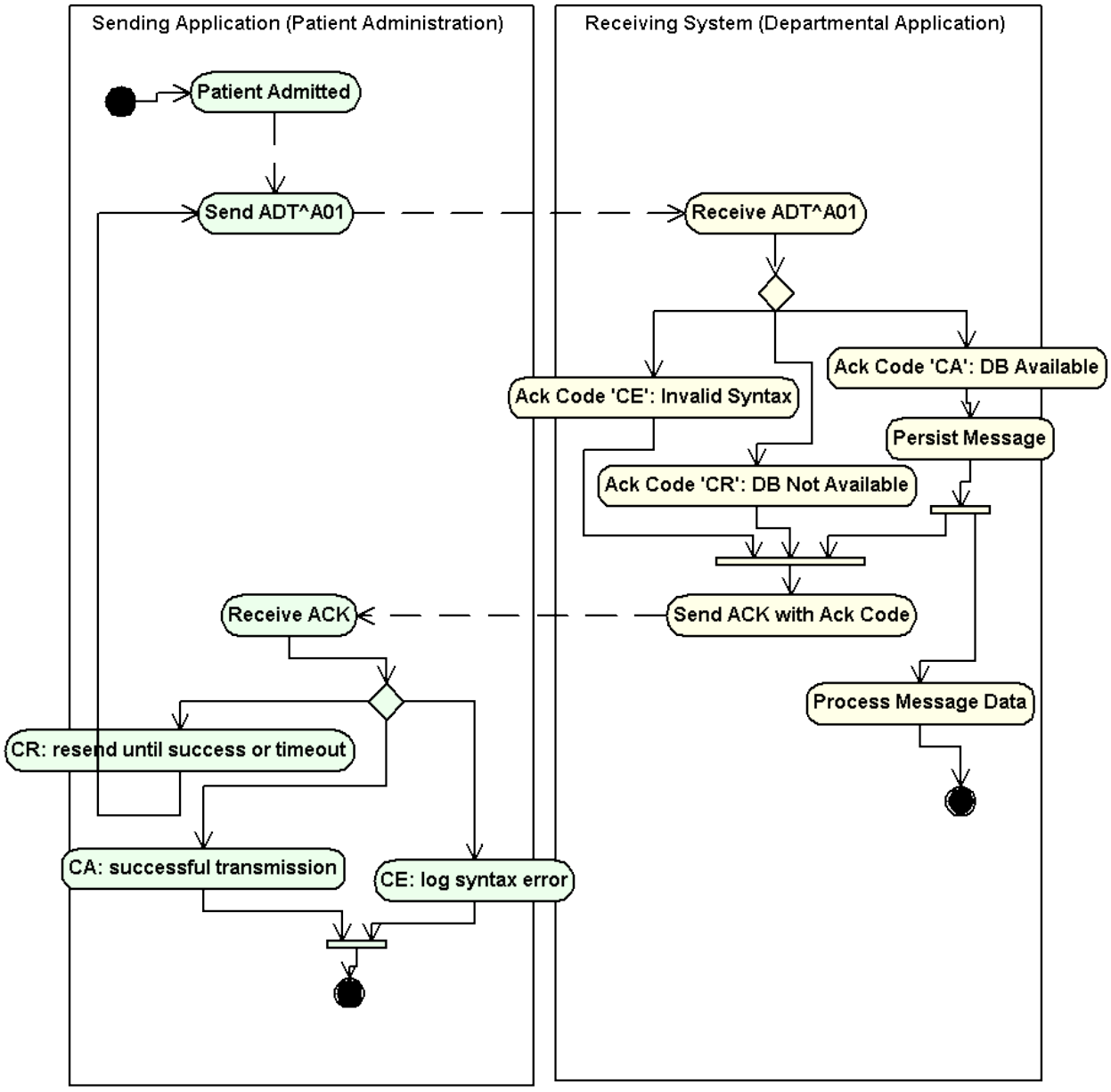
Definition: The Interaction Model illustrates the sequence of trigger events and resulting message flows between 2 or more systems. It may be in literal or graphical form. Graphical form should be a UML activity

diagram. Example activity diagrams are shown here for the original and enhanced acknowledgement modes.

Interaction Model Example – ADT^A01/ACK^A01 (Original Acknowledgement Mode)



Interaction Model Example – ADT^A01/ACK^A01 (Enhanced Acknowledgement Mode)



2B.4.2 Acknowledgements

The specific HL7 acknowledgements required and/or allowed for use with the specified static definition of the HL7 message profile shall be defined. Specifically, the dynamic definition shall identify whether an **accept** and/or **application** level acknowledgement is allowed or required.

For any one static definition there may be one or more dynamic definition.

The dynamic definition shall define the conditions under which an accept and/or application level acknowledgement is expected.

Allowed conditions include:

a) Always

b) Never

c) Only on success

d) Only on error.

2B.5 STATIC DEFINITION

Definition: The static definition is an exhaustive specification for a single message. Normatively expressed in XML, it may be registered on the HL7 web site (See Section *2.B.10*, "*Message profile document*"). The static definition is based on a message structure defined in the HL7 Standard. The message code, trigger event, event description, role (Sender or Receiver) and, if applicable, the order control code will be provided. A complete static definition shall be defined at the **message**, **segment**, and **field** levels. A static definition is compliant in **all aspects** with the HL7-defined message it profiles. However, the static definition may define additional constraints on the standard HL7 message.

A static definition identifies only those specific elements of a standard HL7 message that are used in the exchange.

A static definition explicitly defines:

a) Segments, segment groups, fields and components usage rules

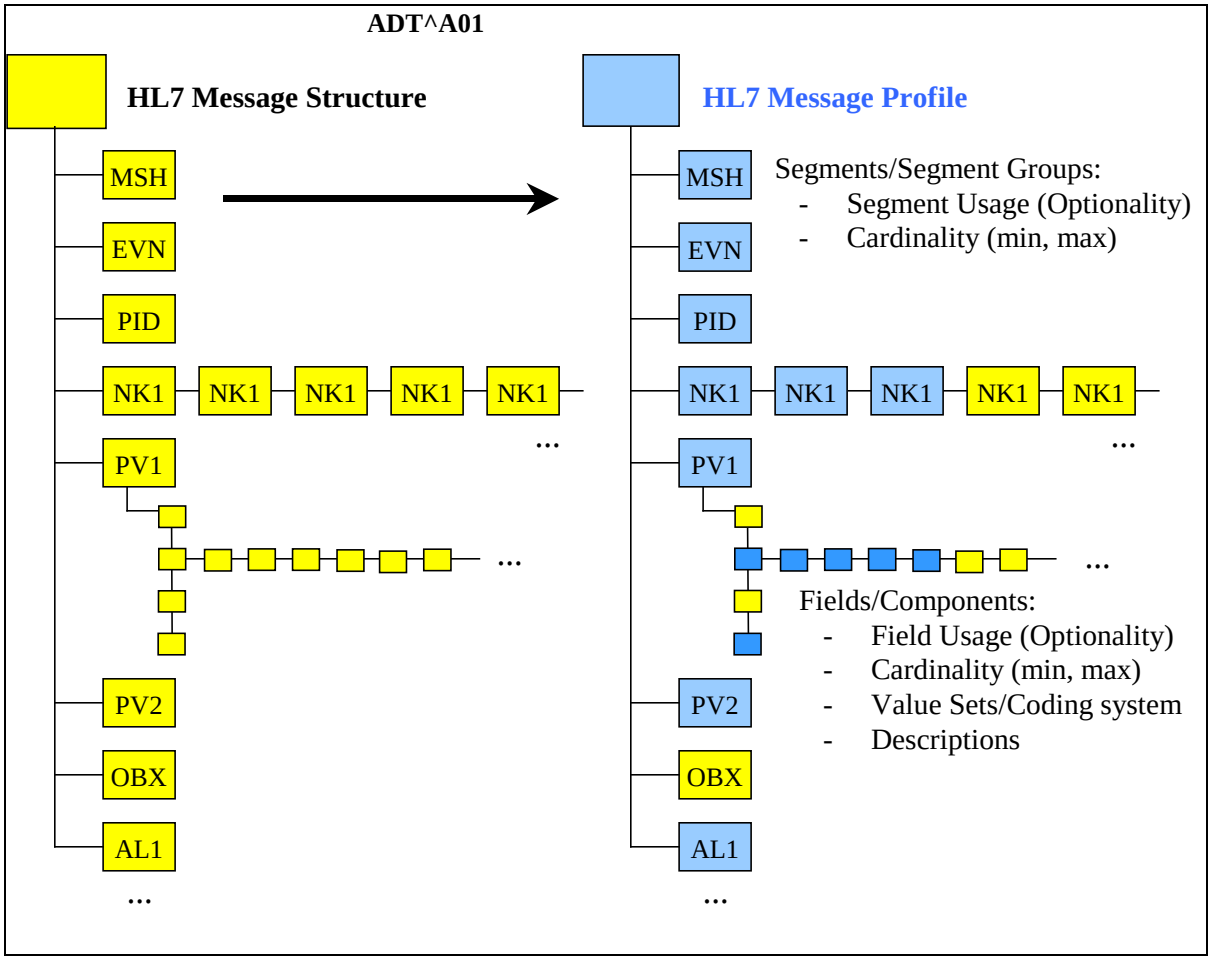
b) Cardinalities

c) Length information

d) Value sets and coding systems.

The following figure depicts, in a graphical way, the concept that the static definition is an overlay of the HL7 message structure further constraining it. For example, where the HL7 message structure shows unlimited number of NK1 Segments, the static definition allows for only three repetitions. Additionally, fields that are optional in the HL7 message structure may be required within the HL7 static definitions.

Static Definition Illustration



2B.5.1 Static definition identifier

Each static definition must have a unique identifier when registered (See section *2.B.10, "Message profile document").* An authority other than the registry may define this identifier. If, at the time of registration, the static profile does not have an identifier assigned by the submitter's authority, the registry authority will assign one. The static definition identifier would be the identifier used if a system asserts a strict conformance claim (see *MSH-21 Message Profile Identifier* in the first section of chapter 2).

2B.5.2 Static definition publish/subscribe topics

Static definition publish/subscribe topics convey the static definition aspects of the message profile. These topics may be used by publish/subscribe systems (see *MSH-21 Message Profile Identifier* in the first section of chapter 2).

The topics are not a normative constituent of the message profile but, if provided as part of the metadata (see section *2.B.10, "Message profile document"*), should be in the format described below. The topic elements will be separated by the dash (-). Any element that does not have a value should be null (nothing between the dashes). As this information may be used in a message instance, it should not contain any HL7 message delimiters.

Static Definition Publish/Subscribe Topics Components

Seq	Topic element name	Value(s)
1	Implementation/Conformance Work Group ID	confsig
2	An organization identifier	Abbreviated version of the organization name
3	The HL7 version	Refer to *HL7 Table 0104 – Version ID* for valid values
4	Topic Type	static
5	Message Type Code	Refer to *HL7 Table 0076 - Message type* for valid values
6	Event Type	Refer to *HL7 Table 0003 - Event Type* for valid values (this table may be extended by locally defined Z trigger events)
7	Order Control Code	Refer to HL7 Table 0119 - Order Control Codes for valid values
8	Structure Type	Refer to *HL7 Table 0354 - Message Structure* for valid values (this table may extended by locally defined message structures)
9	Specification Version	Version number of the application, interface, or specification
10	Specification Status	Status of the application, interface, or specification
11	Role	Sender or Receiver

An example of static definition publish/subscribe topics:

confsig-MyOrganization-2.4-static-ADT-A04--ADT_A01-v2-draft-Sender

2B.6 TABLE DEFINITION

The table definition (or vocabulary) is a specification of a collection of tables used to constrain coded message element content. The table definition is an independent specification that can be embedded in a message profile or referenced by one or more message profiles. Each table definition consists of meta data and code/description pairs. A coded message element is associated with a table in the static definition with the "table" attribute for fields, components, and sub-components. Elements that use coded data types (e.g., CNE, CWE, etc.) are also associated with a table. For the latter, the message includes the code, drawn from *HL7 Table 0396 – Coding System* in Chapter 2C, Code Tables, that uniquely defines the table respective coding system, as well as the coded value itself.

2B.6.1 Table Library

The table definition defines a vocabulary that can be bound to a message profile. The table library specifies a standardized format to organize the vocabulary and provides support to reference it (See section 2.B.2.14). In short, the table library is a container for a collection of tables. Note that the table definition may also be embedded in the XML message profile. A table library is organized as follows:

1) Table library meta data

2) Collection of tables and their values

The table library supports references to tables and instances of tables.

2B.6.1.0 Table Library Meta Data

The following table describes the table library meta data. Definitions for each attribute are given.

Table Library Meta Data

Attributes	Definition	Example/Enumerations
Name	The name of the table library.	AdminMessagesVocabulary

Attributes	Definition	Example/Enumerations
OrganizationName	The organization that created the library.	ABC Medical Group
TableLibraryVersion	The version of the table library.	1.2
Status	The status of the table library.	D (Development), A (Active), I (Inactive)
TableLibraryIdentifier	A unique identifier for the table library.	2.16.840.1.113883.3.72.4.2.134
Description	A text description of the table library.	The Admin Table library provides the vocabulary for our patient registration application (supporting the following message profile events: ADT^A01, ADT^A04, ADT^A08, and ADT^A40).

2B.6.1.1 Table Definition

A ***table definition*** consists of metadata describing the table and specifies a list of code/value pairs. The table definition metadata consists of a table identifier, OID, name, type, version, and code system. ***Table elements*** express code/description pairs. The Table below gives a summary of each table definition attribute along with an example. The table type is a recognized HL7 table as described in section *2.C.1 – Code Tables: Definitions*. Valid identifiers for a table type are HL7, User, Local, External, and Imported.

Table Definition for HL7 Tables

Table Attribute	Definition	Example
Identifier	The table identifier	0001
OID	An OID that identifies the table, not the code system	2.16.840.1.113883.12.1
Name	A descriptive name of the table	Administrative Sex
Type	The type of the table as described in section 2.6.3.6	HL7
Version	The version of the table	1.0
Code System	A code system as specified in the HL7 table 0396	2.5.1

A table element consists of a code, display name (value), and source. The table below gives a summary of each table element attribute along with an example. Valid identifiers for the table element source are HL7, Local, Redefined, or Standard Development Organization (SDO). Redefined means the code/display name has been changed from its original value.

Table Definition for Other Tables

Element Attribute	Definition	Example
Code	The code for the data value	M
Display Name	The long description of the code	Male
Source	The source of the code/value pair	HL7

2B.6.2 Conformance Requirements

The content of a table definition is used to express conformance requirements of coded message elements defined in a message profile. If the content of a coded message element matches that of a table element defined in the table definition identified with the *Table* attribute, the content of the message element is said to be conformant with respect to the message profile. If the content of a coded message element does not match a table element defined in the table definition identified with the *Table* attribute, the content of the message element is said to be non-conformant with respect to the message profile.

2B.7 PROFILE TYPE

There are three basic profile types used in documenting standard conformance:

a) HL7 Standard profile (represents a specific HL7 published standard, creation and publication limited to HL7 use),

b) constrainable profile (with "Optional" elements which must be further constrained in order to create implementation profiles), and

c) implementation profile (no "Optional" parts, fully implementable).

This model allows vendors, SDOs, PEOs or providers to publish generic profiles from which fully constrained implementation profiles can be created.

In comparison with the HL7 standard, separate constrainable and implementation profiles may exist for the receiving and the sending role.

2B.7.1 Vendor constrainable profiles

A vendor might develop a message profile to which all their software products must comply but, in itself, is not an implementation profile. The different products serve potentially different domains and might be implemented with products from other vendors. The vendor profile constrains the HL7 Standard by defining agreed-to vocabularies, conditionality rules, supported items, and local extensions that are shared across all products. The profile is not necessarily fully constrained. For example, the vendor profile might allow the usage code of optional as, across different products, an element may be required in some use cases, be optional or conditional in others, and not be supported at all in still others. Furthermore, at this level, providing exact length definition is optional as well.

The vendor's individual software products might themselves have profiles that would build on, and further constrain, their vendor profile. The product profile would specifically define the information model and the elements contained within. The product profile might still be a constrainable profile as elements might result in different HL7 messages based on configuration settings and customizations. Only once all configuration settings and customizations have been taken into account can you have a fully-constrained 'Implementation' profile.

Constrainable profiles can be useful for interface engine applications which must be flexible enough to allow for receipt of messages based on a variety of message profiles. The desire of the application would be to validate message instances against one constrainable profile.

2B.7.2 Realm constrainable profiles

Realm, national and regional, profiles represent localization and restrictions placed on the appropriate standard, while providing enough optionality for basing the more specific implementation profiles. Some examples of realm constrainable profiles are:

a) AS4700.1-2001 Implementation of HL7 v2.3.1 Part 1:Patient Administration (constrainable profile for Australian Standards, constrains HL7 2.3.1, Chapter 3).

b) AS/NZS 4700.3-1999 Implementation of HL7 v2.3 Part 3: Electronic messages for exchange of information on Drug Prescription (constrainable profile for Australian Standards, constrains HL7 2.3, various Chapters).

2B.7.3 Implementation profiles

Implementation profiles represent the lowest level of specification required for unambiguous implementation. Examples of some implementation profiles are:

a) Adverse Drug Reaction Implementers Specification, 2001, TGA (implementation profile, constrains Australian Standards and HL7 v2.3.1 constrainable profiles for Therapeutic Goods Administration ADRAC Messaging Implementation Project),

b) Diabetes Reporting Implementers Specification, 2001, UNSW (implementation profile, constrains Australian Standards and HL7 v2.3.1 constrainable profiles for University of NSW Diabetes

Messaging Implementation Project),

c) Specific version of a product, as implemented, at a specific provider.

Within an implementation profile the exact length definition must be provided.

2B.8 STATIC DEFINITION CONCEPTS

This section discusses concepts common to each level of the static definition (message, segment and field). It uses the generic term 'element' to refer to segment groups, segments, fields, components and sub-components.

2B.8.1 Length

An unambiguous definition of length requires a clear understanding of what it applies to and how it is measured. Length is defined to be a constraint on the number of characters that may be present in one occurrence of a message element. This definition satisfies both requirements; it applies (strictly) to an element's data value—i.e., the set of characters present in the message representing a value of the element's predefined data type—and it is measured in characters as defined by the rules in chapter 2. The definition is system independent. For example, a system that encodes characters using one byte and a system that uses two-byte encoding would use the same value for length to impose the same length constraint. Length does not count the HL7 characters used to represent the value, only the number of characters in the value itself is counted. If the null character is represented by transmitting "", length conforms to any minimum and maximum length specification.

Length shall be interpreted as a restriction on an element's data value, not on the presence or absence of the element. A length value of zero is appropriate when the null character is transmitted, but this does not imply the element is not present; clearly two characters will be present if null is transmitted. Restricting the applicability of length to data values present in the message is necessary. Not only does it keep the concept simple and eliminate the need to address special cases, it also allows for the transmission of null values for required elements. A required element can have a null value, since this still clearly means there is a data value encoded for the element in the message. If an element is empty or not present in the message, i.e., there is no data value encoded for the element in the message, then length restrictions do not apply since there is nothing to restrict and no length constraint that can be violated.

Length shall be specified using the following syntax: "m..n", where m and n are non-negative integers designating the minimum and maximum number of characters the element may have, respectively, where n ≥ m. When an upper bound for length cannot be determined in advance, the use of the asterisk character, "*", may be used as a place holder for the maximum value, so that, in addition, to the above syntax, where m and n are integer values, a constraint of the form "m..*" may be used to indicate the maximum length constraint is unknown.

Example length constraints are shown in the *Minimum and Maximum Length Examples* table.

Minimum and Maximum Length Examples

Value	Description	Comment
	For constrainable profile: no length defined, i.e. no requirements on the length are given. Leaving this information empty is not allowed for implementable profiles.	
0..0	For withdrawn elements: minimum and maximum set to 0.	
1..1	Element must have exactly one character	
1..n	Element may have up to n characters	
n..n	Element must have exactly "n" characters	
1..*	Element may have any length.	
n..*	Element may have any length which is greater than or equal to "n", where "n" is greater than or equal to 1.	

Value	Description	Comment
m..n	Element must have a minimum length of "m" and a maximum length of "n" where "m" is less than or equal to "n" and "m" is greater than or equal to 1.	

Note: Whether or not an element is populated is controlled by cardinality. But if the element is populated with a non-null value, the minimum and maximum length definition must hold. The null information representation (two double quotes) is not considered to be a value with applicable length information.

Length should not be specified for composite elements. In these cases, the actual minimum and maximum lengths can be very difficult to determine due to the interdependencies on the component content, and the specification of actual lengths is not useful either. For example, if an overall length of 4..20 is assigned to a data element with a type CWE, what does this mean in practice? However if a length is specified, the vertical bar representation of the data must conform to the stated length, allowing for an additional character for each HL7 separator character.

2B.8.2 Conformance Length

Constrainable profile specifications may also specify a conformance length. This is the number of characters that any conformant application must be able to correctly handle. For example, a constrainable profile may declare that the minimum and maximum lengths of a specific field are 3 and 2500. An implementation profile may further constrain this length to specify what is actually supported by an application. However an application could declare a length of 3..4, which may not be useful within the context of the constrainable profile. A constrainable profile may specify conformance lengths to establish a minimum expectation. In the example case, if the constrainable profile specifies a conformance length of 200, no other profile may assert conformance to the constrainable profile unless its maximum length is 200 or greater.

Conformance length is a redundant concept in implementation profiles that will not be further constrained, and should not be specified.

2B.8.3 Truncation Flag

As of 2.7, a truncation pattern is defined which can be used to assist applications to manage the existence of data that exceeds the maximum number of characters that can be properly handled. The truncation pattern is described in Chapter 2, section 2.5.5.2, "Truncation Pattern". Note that the actual truncation behavior of the pattern is data type dependent. Applications shall not use truncation if the profile prohibits it. Applications may support truncation if the profile permits it. Message Profiles may specify the truncation behaviour.

The truncation flag is a simple Boolean. In a constrainable profile, the value may be true or false. False signifies that the element may not be truncated, while true means that the value may be truncated. If a profile fixes truncation to false, no other further constraining profile may mark this value as true. If the value is fixed to true, other further constraining profiles may mark it as true or false.

In an implementation profile, a value of true for the truncation flag signifies that the application supports the defined truncation behavior for the appropriate type. A value of false indicates that the application does not support data truncation for this element.

Although the truncation pattern was only defined in v2.7, the behavior may be adopted for previous versions of HL7, and the truncation flag may be used with previous version. Note that in these cases, the truncation character cannot be specified in the message, and some other arrangement must exist.

2B.8.4 Cardinality

In order to separate message content requirements from application behavior requirements, cardinality is used to control message content, and usage is used to define application requirements. Cardinality controls the number of occurrences of an element appearing in a message. Some elements shall always be present (e.g., cardinality [1..1], [1..n]). Others shall never be present (i.e., cardinality [0..0]). Others may be optional with zero or more occurrences (e.g., cardinality [0..n]). Cardinality identifies the minimum and maximum number of occurrences that a message element must have in a conformant message. Cardinalities

are expressed as a minimum-maximum pair of non-negative integers. A conformant message must always contain at least the minimum number of occurrences, and shall not contain more than the maximum number of occurrences.

An explicit cardinality range is required for segment group, segment, and field elements. Component and sub-component elements do not explicitly include a cardinality range, but a cardinality range is implicitly associated with each component and sub-component element. The associated cardinality depends on the element's *usage* code. For components and sub-components with a usage code of *R*, the associated cardinality range is [1..1]; for all elements with a usage code of *RE* or *O*, the associated cardinality is [0..1]; for all elements with a usage code of C(a/b), the associated cardinality is determined by the resultant usage based on the evaluation of the condition predicate, and for all elements with an *X* usage code, the associated cardinality is [0..0].

There are two special values for cardinality. If the minimum number of occurrences is 0, the element may be omitted from a message. In certain circumstances, the maximum number of occurrences may have no specified limit. In this case, it is identified with "*" (e.g., [n..*]).

Valid cardinality values are shown in the *Cardinality* table; combinations not designated in the table are invalid. In particular, usage code RE is not allowed with cardinalities [1..1], [1..n], and [1..*]. Cardinality [m..n], where *m* is greater than 1, is allowed with usage codes R and RE. If an element with this cardinality range has a usage code RE, the element may be omitted from the message but if present, it must have at least *m* occurrences and may not have more than *n* occurrences.

Cardinality

Value	Description	Valid Usage Codes	Comment
[0..0]	Element never present	X	
[0..1]	Element may be omitted and it can have at most one occurrence	RE, O, C(a/b)	
[1..1]	Element must have exactly one occurrence	R	
[0..n]	Element may be omitted or may have up to n occurrences	RE, O, C(a/b)	
[1..n]	Element must appear at least once, and may have up to n occurrences	R	
[0..*]	Element may be omitted or may have an unlimited number of occurrences	RE, O, C(a/b)	
[1..*]	Element must appear at least once, and may have an unlimited number of occurrences	R	
[m..n] [1]	Element must have at least "m" occurrences and may have at most "n" occurrences. Except that in the case where the usage code is RE, the element may also be omitted or have zero occurrences	R and RE	
[m..*][3]	Element must have at least "m" occurrences and may have an unlimited number of occurrences. Except that in the case where the usage code is RE, the element may also be omitted or have zero occurrences.	R and RE	

2B.8.5 Usage

Message content is governed by the cardinality specification associated (explicitly or implicitly) with each element of an HL7 message. Usage rules govern the expected behavior of the sending application and receiving application with respect to the element. The usage codes expand/clarify the optionality codes defined in the HL7 standard. Usage codes are employed in a message profile to constrain the use of elements defined in the standard. The usage code definitions are given from a sender and receiver perspective and specify implementation and operational requirements.

The standard allows broad flexibility for the message structures that HL7 applications must be able to receive without failing. But while the standard allows that messages may be missing data elements or may contain extra data elements, it should not be inferred from this requirement that such messages are

[1] *m* must be greater than 1 and *n* must be greater than or equal to m; the case where *m* equals 1 is addressed separately.

conformant. In fact, the usage codes specified in a message profile place strict conformance requirements on the behavior of the application.

2B.8.5.0 Definition of Conditional Usage

The conditional usage is defined as follows:

C(a/b) - "a" and "b" in the expression are placeholders for usage codes representing the true ("a") predicate outcome and the false ("b") predicate outcome of the condition. The condition is expressed by a conditional predicate associated with the element ("See section *2.B.7.9*, "Condition predicate"). "a" and "b" shall be one of "R", "RE", "O" and/or "X". The values of "a" and "b" can be the same.

The example C(R/RE) is interpreted as follows. If the condition predicate associated with the element is true then the usage for the element is R-Required. If the condition predicate associated with the element is false then the usage for the element is RE-Required but may be empty.

There are cases where it is appropriate to value "a" and "b" the same. For example, the base standard defines the usage of an element as "C" and the condition predicate is dependent on the presence or non-presence of another element. The profile may constrain the element that the condition is dependent on to X; in such a case the condition should always evaluate to false. Therefore, the condition is profiled to C(X/X) since the desired effect is for the element to be not supported. Note it is not appropriate (in this case) to profile the element to X since this breaks the rules of allowable usage profiling (see table HL7 Optionality and Conformance Usage).

Usage Rules for a Sending Application

Optionality/ Usage Indicator	Description	Implementation Requirement	Operational Requirement
R	Required	The application shall implement "R" elements.	The application shall populate "R" elements with a non-empty value.
RE	Required but may be empty	The application shall implement "RE" elements.	The application shall populate "RE" elements with a non-empty value if there is relevant data. The term "relevant" has a confounding interpretation in this definition[2].
C(a/b)	Conditional	An element with a conditional usage code has an associated condition predicate (See section *2.B.7.9*, "*Condition predicate*") that determines the requirements (usage code) of the element. If the condition predicate associated with the element is true, follow the rules for ***a*** which shall be one of "R", "RE", "O" or X": If the condition predicate associated with the element is false, follow the rules for ***b*** which shall be one of "R", "RE", "O" or X". ***a*** and ***b*** can be valued the same.	
X	Not supported	The application (or as configured) shall not implement "X" elements.	The application shall not populate "X" elements.
O	Optional	None. The usage indicator for this element has not yet been defined. For an implementation profile all optional elements must be profiled to R, RE, C(a/b) or X.	Not Applicable.

[2] There are multiple interpretations of "RE" when a value is known. One is "the capability must always be supported and a value is sent if known", the other is "the capability must always be supported and a value may or may not be sent even when known based on a condition external to the profile specification. The condition may be noted in the profile but cannot be processed automatically". This is what can be interpreted from the "relevant" part of the definition. Regardless of the interpretation the "RE" usage code, a set of test circumstances can be developed to sufficiently test the "RE" element. See the "Conformity Assessment of Conformance Constructs" section for more details.

Usage Rules for a Receiving Application

Optionality/ Usage Indicator	Description	Implementation Requirement	Operational Requirement
R	Required	The application shall implement "R" elements.	The receiving application shall process (save/print/archive/etc.) the information conveyed by a required element. A receiving application shall raise an exception due to the absence of a required element. A receiving application shall not raise an error due to the presence of a required element,
RE	Required but may be empty	The application shall implement "RE" elements.	The receiving application shall process (save/print/archive/etc.) the information conveyed by a required but may be empty element. The receiving application shall process the message if the element is omitted (that is, an exception shall not be raised because the element is missing).
C(a/b)	Conditional	An element with a conditional usage code has an associated condition predicate (See section *2.B.7.9*, "*Condition predicate*") that determines the requirements (usage code) of the element. If the condition predicate associated with the element is true, follow the rules for a which shall be one of "R", "RE", "O" or X": If the condition predicate associated with the element is false, follow the rules for b which shall be one of "R", "RE", "O" or X". ***a*** and ***b*** can be valued the same.	
X	Not supported	The application (or as configured) shall not implement "X" elements.	None, if the element is not sent. If the element is sent the receiving application may process the message, shall ignore the element, and may raise an exception. The receiving application shall not process (save/print/archive/etc.) the information conveyed by a not-supported element.
O	Optional	None. The usage indicator for this element has not yet been defined. For an implementation profile all optional elements must be profiled to R, RE, C(a/b), or X.	None.

2B.8.6 Relationship between HL7 optionality and conformance usage

Conformance usage codes are more specific than HL7 Optionality codes. Because of the requirement that conformance statements must be compliant with the HL7 message definition it is derived from, there are restrictions on what usage codes may be assigned to an element based on the HL7 Optionality. For example, any element designated as required in a standard HL7 message definition shall also be required in all HL7 message profiles of that standard message.

HL7 Optionality and Conformance Usage

Base Optionality/Usage	Derived (Constrained) Optionality/ Usage	Comment
R - Required	R	
RE - Required but may be empty[3]	RE, R	
O - Optional	R, RE, O, C(a/b), X	O is only permitted for constrainable profiles
C - Conditional	C(a/b),R	"a" and "b" shall be one of "R", "RE", "O" or "X". "a" and "b" can be valued the same.

[3] This is an optionality added with v 2.7. The meaning is slightly different from the message profile usage code of RE.

Base Optionality/Usage	Derived (Constrained) Optionality/ Usage	Comment
X – Not Supported	X	
B – Backward Compatibility	R, RE, O, C(a/b), X	B is only permitted for constrainable definitions
W - Withdrawn	X	

Conformance usage codes can also be further constrained in subsequence constrainable or implementation profiles. The *allowed conformance usage profiling* table indicates the how the conformance usage codes can be further constrained.

Allowed Conformance Usage Profiling

Conformance Usage	Constrained Possibilities	Comment
R - Required	R	
RE - Required but may be empty	RE, R	
C(a/b)	Follow the rules for R, RE, and X in this table as they apply to a and b.	For example, C(RE/X) can be further constrained to C(R/X).
X – Not Supported	X	

Note:

- The conditional usage construct shall not be nested.
- The condition in the derived profile shall not modify the condition in the *base* profile. The exception is a removal of the condition in a derived profile if it can be constantly evaluated to either true or false and then be replaced by the appropriate usage code.

2B.8.7 Relationship between usage and cardinality

Cardinality governs the appearance of a field, and usage governs the expected behavior of applications. Nevertheless, a relationship exists between them that must be maintained. The valid combinations of the two are defined in the *Cardinality* table. For purposes of message profiles, selected cardinality and usage combinations are examined here. The constraints on allowed combinations are:

a) If the usage of an element is Required (R), the minimum cardinality for the element shall always be greater than or equal to 1.

b) If the usage of an element is not Required (R) (i.e., any code other than 'R'), the minimum cardinality shall be 0 except in the following condition:

   If the profile author wishes to express a circumstance where an element will not always be present, but when present must have a minimum number of repetitions greater than one, this may be indicated by specifying the non-required Usage code with the minimum cardinality representing the minimum number of repetitions when the element is present. This is accomplished, as in UML, with an expression of the form as (m..n), indicating that permitted occurrences are either zero, or the range of m through n.

Example combinations:

Example Usage-Cardinality Combinations

Cardinality	Usage	Interpretation
[1..1]	R	There will always be exactly 1 occurrence present.
[1..5]	R	There will be between 1 and 5 occurrences present.
[0..1]	RE	The element must be supported, but may not always be present.

Cardinality	Usage	Interpretation
[0..5]	C(R/X)	If the condition predicate is true, there will be between 1 and 5 occurrences. If the condition predicate is false, there will be 0 occurrences.
[3..5]	RE	If any values for the element are sent, there must be at least 3 and no more than 5 occurrences. However, the element may be absent (0 occurrences).

2B.8.8 Usage within hierarchical elements

As part of the conformance framework, there is an additional rule for determining whether a particular 'element' is present. The rule is as follows: For an element to be considered present, it must have content. This means that simple elements (fields, components or sub-components with simple data types such as NM, ST, ID) must have at least one character. Complex elements (those composed of other elements, e.g., Messages, Segment Groups, Segments, Fields with complex data types such as CNE, XPN, etc.), must contain at least one element that is present. Elements that do not meet these conditions are not considered to be present.

For example, if a field is made up of 4 required but may be empty components, at least one of the components must be present in order for the field to be considered present. Thus, if the field is profiled as Required, an instance message would only be conformant if the field contained at least one populated component. The reason for this rule is to ensure that the intent of the profile is met. The rule is necessary because the traditional 'vertical bar' encoding allows, for example, for a bare segment identifier with no fields (e.g., a line containing just "NTE|" would be considered valid under the standard rules, but would be considered not present as far as testing against a conformance specification). The XML encoding also allows this, as well as fields without their components, components without their sub-components, etc. (e.g., <PID.3/>).

2B.8.9 Condition predicate

If the usage code of an element is C (i.e., C(a/b), then a conditionality predicate must be associated with this element that identifies the conditions under which the element must be or is allowed to be present. The predicate must be testable and based on other values within the message. This predicate may be expressed as a mathematical expression or in text and may utilize operators such as equivalence, logical AND, logical OR and NOT. The conforming sending and receiving applications shall both evaluate the predicate. When the Usage is not 'C', the conditionality predicate will not be valued.

2B.8.10 Annotation

Annotations provide further explanations to educate prospective users and/or implementers. These are usually used to enhance the descriptions of the elements of the base specification in order to relate them to a particular context.

Types of annotations supported:

- Definition: An explanation of the meaning of the element.
- Description: An explanation of the associated element. This may contain formatting markup.
- Design Comment: Internal development note about why particular design decisions were made, outstanding issues and remaining work. They may contain formatting markup. Not intended for external publication.
- Implementation Note: Implementation Notes provide a general description about how the element is intended to be used, as well as hints on using or interpreting the it.
- Other Annotation: Additional content related to the element.
- Example: An example instance
- Added ability to communicate pattern matching and element relationships. These, as well as condition predicate, will allow for text and formal testable constraints.

2B.9 STATIC DEFINITION - MESSAGE LEVEL

The message level static definition shall be documented using the HL7 **abstract message** syntax, with the addition of specifying cardinality and usage for each of the segments contained within the message structure.

The **usage** column (OPT) shall be updated to reflect the usage of the segment or group within this particular static definition.

The **cardinality** column shall accurately reflect the minimum and maximum number of occurences of the field allowed for the segment or group within this particular static definition.

Sample Static Definition - Message Level

ADT^A01^ADT_A01	ADT Message	Status	Usage	Cardinality	Chapter
MSH	Message Header		R	[1..1]	2
[{ SFT }]	Software Segment		X	[0..0]	2
[UAC]	User Authentication Credential		X	[0..0]	2
EVN	Event Type		R	[1..1]	3
PID	Patient Identification		R	[1..1]	3
[PD1]	Additional Demographics		X	[0..0]	3
[ARV]	Access Restrictions		X	[0..0]	3
[{ ROL }]	Role		X	[0..0]	15
[{ NK1 }]	Next of Kin / Associated Parties		RE	[0..3]	3
PV1	Patient Visit		C	[0..1]	3
[ARV]	Access Restrictions		X	[0..0]	3
[PV2]	Patient Visit - Additional Info.		RE	[0..1]	3
[{ ROL }]	Role		X	[0..0]	15
[{ DB1 }]	Disability Information		X	[0..0]	3
[{ OBX }]	Observation/Result		X	[0..0]	7
[{ AL1 }]	Allergy Information		RE	[0..10]	3
[{ DG1 }]	Diagnosis Information		X	[0..0]	6
[DRG]	Diagnosis Related Group		X	[0..0]	6
[{	--- PROCEDURE begin		X	[0..0]	
PR1	Procedures		X	[0..0]	6
[{ ROL }]	Role		X	[0..0]	15
}]	--- PROCEDURE end				
[{ GT1 }]	Guarantor		X	[0..0]	6
[{	--- INSURANCE begin		X	[0..0]	
IN1	Insurance		X	[0..0]	6
[IN2]	Insurance Additional Info.		X	[0..0]	6
[{ IN3 }]	Insurance Additional Info - Cert.		X	[0..0]	6
[{ ROL }]	Role		X	[0..0]	15
}]	--- INSURANCE end				
[ACC]	Accident Information		X	[0..0]	6
[UB1]	Universal Bill Information		X	[0..0]	6
[UB2]	Universal Bill 92 Information		X	[0..0]	6

ADT^A01^ADT_A01	ADT Message	Status	Usage	Cardinality	Chapter
[PDA]	Patient Death and Autopsy		X	[0..0]	3

2B.9.1 Segment definitions

The set of segments and segment groups included within the message shall be defined. Any segments or segment groups that are required by HL7 shall be included.

2B.9.2 Segment usage

The usage of the segment or group within a message shall be defined using one of the codes in the previously defined usage table.

2B.9.3 Segment cardinality

Some segments and segment groups within the HL7 message are allowed to repeat. The cardinality of all the segments and groups within the message shall be defined.

Static definition - segment level

The segment level static definition shall be documented using the HL7 **segment attribute table** format with the addition of specifying length, usage and cardinality for each of the fields contained within the segment.

- The **length column** shall be updated to accurately reflect the maximum allowed length for the field within this segment definition.
- The **usage column** shall accurately reflect the usage of the field within this segment definition.
- The **cardinality column** shall accurately reflect the minimum and maximum number of repetitions of the field allowed for this segment definition.

Sample Segment Level Definition – PID (Patient Identification) Segment

SEQ	LEN	C.LEN	DT	OPT	RP/#	TBL#	ITEM#	ELEMENT NAME
1	1..4		SI	O			00104	Set ID - PID
2				W			00105	Patient ID
3			CX	R	Y		00106	Patient Identifier List
4				W			00107	Alternate Patient ID - PID
5			XPN	R	Y	0200	00108	Patient Name
6			XPN	O	Y		00109	Mother's Maiden Name
7			DTM	O			00110	Date/Time of Birth
8			CWE	O		0001	00111	Administrative Sex
9				W			00112	Patient Alias
10			CWE	O	Y	0005	00113	Race
11			XAD	O	Y		00114	Patient Address
12				W			00115	County Code
13			XTN	B	Y		00116	Phone Number - Home
14			XTN	B	Y		00117	Phone Number - Business
15			CWE	O		0296	00118	Primary Language
16			CWE	O		0002	00119	Marital Status

SEQ	LEN	C.LEN	DT	OPT	RP/#	TBL#	ITEM#	ELEMENT NAME
17			CWE	O		0006	00120	Religion
18			CX	O			00121	Patient Account Number
19				W			00122	SSN Number - Patient
20				W			00123	Driver's License Number - Patient
21			CX	O	Y		00124	Mother's Identifier
22			CWE	O	Y	0189	00125	Ethnic Group
23		250#	ST	O			00126	Birth Place
24	1..1		ID	O		0136	00127	Multiple Birth Indicator
25		2=	NM	O			00128	Birth Order
26			CWE	O	Y	0171	00129	Citizenship
27			CWE	O		0172	00130	Veterans Military Status
28				W			00739	Nationality
29			DTM	O			00740	Patient Death Date and Time
30	1..1		ID	O		0136	00741	Patient Death Indicator
31	1..1		ID	O		0136	01535	Identity Unknown Indicator
32			CWE	O	Y	0445	01536	Identity Reliability Code
33			DTM	O			01537	Last Update Date/Time
34			HD	O			01538	Last Update Facility
35			CWE	C		0446	01539	Taxonomic Classification Code
36			CWE	C		0447	01540	Breed Code
37		80=	ST	O			01541	Strain
38			CWE	O	2	0429	01542	Production Class Code
39			CWE	O	Y	0171	01840	Tribal Citizenship
40			XTN	O	Y		02289	Patient Telecommunication Information

2B.9.4 Field definitions

The set of fields of each segment within the message level definition shall be specified.

If a segment occurs multiple times within a message profile, it may be represented by different segment profiles. This shall be explicitly defined within the message level definition.

2B.9.5 Field cardinality

Some fields within a segment are allowed to repeat. The cardinality of all the fields within the segment shall be defined.

2B.9.6 Field usage

The **usage** of the field within a segment shall be defined consistent with the profile type, and using one of codes identified in the previously defined Usage tables.

2B.9.7 Data type

The data type of the field within a segment shall be updated to accurately reflect the data type for the field within this segment definition.

2B.9.8 Length

The length of the field within a segment shall be updated to accurately reflect the correct length for the field within this segment definition. Here two information items shall be provided representing the required minimum length a data constituent must have and the maximum length this constituent must not exceed.

2B.9.9 Conformance Length

This length represents the number of characters that a conformant application must be able to handle. For further information please see Chapter 2, section 2.5.5.3, "Conformance Length".

2B.9.10 Table reference

The name of the table of the field within a segment shall be updated to accurately reflect the table used for the field within this segment definition.

2B.10 STATIC DEFINITION - FIELD LEVEL

2B.10.1 Field Definitions

Each individual field within a segment shall be completely defined to eliminate any possible ambiguity.

In cases where HL7 2.x field descriptions are not sufficient, a precise semantic definition shall be specified.

2B.10.2 User-defined and suggested field values

The allowed code sets (table) for many fields within the HL7 Standard are specified as user-defined (data type IS) or HL7-defined (data type ID) values.

In these cases, the **exact allowed code set** shall be specified. These values shall be defined according to the specified scope of use for the message profile by vendors, provider, or within a realm.

Coded Entry (CE, CF, CWE, and CNE) type fields are specified as being populated based on coding systems. For each of these fields, the specific coding system used shall be identified. Compliant applications are required to use the specified coding system, but may also use an alternate coding system as supported by the data type (See the example within each data type definition).

2B.10.3 Constant values

If an element will always have a constant value, this shall be specified. Constant values may only be specified for elements that represent primitive data types, i.e., they have no components or sub-components.

2B.10.4 Data values

A list of example data values for the element may be specified. Data values may only be specified for elements that represent primitive data types i.e. they no components or sub-components.

2B.10.5 Pattern Matching

Constraints for matching patterns within fields may be specified. In addition to textual description of the constraint, formal expressions may be specified. These formal expressions can be Object Constraint Language (OCL), regular expressions (RegEx)[4], and XML Path Language (XPath)[5]

[4] Refer to Part 2 of W3C schema (http://www.w3.org/TR/xmlschema-2/) for details on regular expressions.

[5] Refer to XML Path Language (http://www.w3.org/TR/xpath)

2B.10.6 Element Relationships

Element relationships may be may be specified. In addition to textual description of these constraints, formal expressions may be specified. These formal expressions can be Object Constraint Language (OCL), regular expressions (RegEx)[6], and XML Path Language (XPath)[7]

2B.10.7 Profiling Multiple Occurrences

Individual occurrences of an element can be profiled differently. This mechanism allows for greater flexibility for constraining elements. For example, each occurrence of a patient identifier list can be profiled differently. Prior to HL7 V2.8 individual occurrences of an element had to be profiled the same constraints and often represented a superset of the constraints.

To illustrate the utility of profiling individual elements, consider the following example:

Suppose a field has a cardinality of 3. Suppose the field is made up of 4 components and that the first occurrence of the field should always be constrained as indicated:

<comp name="comp 1" usage="R"/>

<comp name="comp 2" usage="R"/>

<comp name="comp 3" usage="X"/>

<comp name="comp 4" usage="X"/>

Suppose the second occurrence of the field, when present, is constrained in the following way:

<comp name="comp 1" usage="R"/>

<comp name="comp 2" usage="RE"/>

<comp name="comp 3" usage="RE"/>

<comp name="comp 4" usage="X"/>

And that the third occurrence of the field is as follows:

<comp name="comp 1" usage="X"/>

<comp name="comp 2" usage="X"/>

<comp name="comp 3" usage="R"/>

<comp name="comp 4" usage="R"/>

Using the profiling mechanism prior v 2.8, it is not possible to constrain the field in the manner indicated. To allow for the 3 occurrences described above, the profile would have to include an element such as the following for constraining the field:

<field name="field X" min="1" max ="3" ...>

<comp name="comp 1" usage="RE"/>

<comp name="comp 2" usage="RE"/>

<comp name="comp 3" usage="RE"/>

<comp name="comp 4" usage="RE"/>

</field>

[6] Refer to Part 2 of W3C schema (http://www.w3.org/TR/xmlschema-2/) for details on regular expressions.

[7] Refer to XML Path Language (http://www.w3.org/TR/xpath)

To allow the profile writer to constrain each occurrence of the field, an occurrence element can be used. Using the occurrence element, the profile would replace the field definition above with the following field definition:

```
<field name="field X" min="1" max ="3" ...>
    <occurrence >
        <comp name="comp 1" usage="R"/>
        <comp name="comp 2" usage="R"/>
        <comp name="comp 3" usage="X"/>
        <comp name="comp 4" usage="X"/>
    </occurrence>
    <occurrence>
        <comp name="comp 1" usage="R"/>
        <comp name="comp 2" usage="RE"/>
        <comp name="comp 3" usage="RE"/>
        <comp name="comp 4" usage="X"/>
    </occurrence>
    <occurrence>
        <comp name="comp 1" usage="X"/>
        <comp name="comp 2" usage="X"/>
        <comp name="comp 3" usage="R"/>
        <comp name="comp 4" usage="R"/>
    </occurrence>
</field>
```

To be clear, it should be noted that the new profile element does not imply any changes to the message encoding; it simply allows for greater control over the individual occurrences of the element than the standard presently allows.

There are a number of options that can be used to specify individual occurrences. The *order* attribute can be specified to indicate if the order of the occurrences is significant. By default, as in the example above if the *order* is not specified or specified as "false" then the instance elements can adhere to any of the occurrences specified. If the *order* attribute is "true" then order is significant and the instance elements must appear in the order specified.

To enable specific occurrences to be constrained without the need to specify an order constraint for every occurrence of the element, the *number* attribute is optional. The *order* and *number* attributes are mutually exclusive. As an example, if it was necessary to tightly constrain the second occurrence but more flexibility was allowed the other occurrences, the profile could specify:

```
<field name="field X" min="1" max ="3" ...>
    <occurrence>
        <comp name="comp 1" usage="RE"/>
        <comp name="comp 2" usage="RE"/>
        <comp name="comp 3" usage="RE"/>
        <comp name="comp 4" usage="RE"/>
```

```
        </occurrence>
        <occurrence number="2">
                <comp name="comp 1" usage="R"/>
                <comp name="comp 2" usage="R"/>
                <comp name="comp 3" usage="X"/>
                <comp name="comp 4" usage="X"/>
        </occurrence>
</field>
```

An occurrence element without a number attribute would be interpreted as applying to all occurrences of the element not otherwise specified. In other words, in the above case, all components in each occurrence of the field are designated as RE except the second, which must have exactly the first two components present. Occurrence specifications can have any number of *number* or non-number attributes.

The occurrence element can also be specified based on the value of a designated component in the field. The "*position*" attribute specifies the component that is evaluated to determine the profiled occurrence. The component that the *position* attribute references shall always be profiled as required. The "*value*" attribute provides the value for determining the profiled occurrence; it is evaluated with the component referenced by the *position* attribute. For example, the occurrences of field with an XPN data type can be profiled individually based on the name type code (component 7 in this case).

```
<field name="field X" min="1" max ="3" position= "7"...>
        <!—legal name-->
        <occurrence  value="L">
                <Component name="family name" usage="R"/>
                <Component name="given name" usage="R"/>
                <Component name="Initials" usage="X"/>
                <Component name="Suffix" usage="X"/>
                <Component name="Prefix" usage="R"/>
                <Component name="Degree" usage="X"/>
                <Component name="Name Type Code" usage="R"/>
                Etc.
        </occurrence>
        <!—display name-->
        <occurrence value="D">
                <Component name="family name" usage="R"/>
                <Component name="given name" usage="R"/>
                <Component name="Initials" usage="RE"/>
                <Component name="Suffix" usage="RE"/>
                <Component name="Prefix" usage="RE"/>
                <Component name="Degree" usage=RE"/>
                <Component name="Name Type Code" usage="R"/>
                Etc.
```

```
        </occurrence>
        <!—birthname name-->
        <occurrence value="B">
                <Component name="family name" usage="R"/>
                <Component name="given name" usage="R"/>
                <Component name="Initials" usage="RE"/>
                <Component name="Suffix" usage="X"/>
                <Component name="Prefix" usage="X"/>
                <Component name="Degree" usage="X"/>
                <Component name="Name Type Code" usage="R"/>
                Etc.
        </occurrence>
</field>
```

If the name type code in the element occurrence is "L" then the occurrence specification is applied. The same logic is also applied to display and birthname occurrences.

Not all occurrences need to have a *value* attribute when using the *position/value* occurrence profiling option. An occurrence element without a value attribute would be interpreted as applying to all occurrences of the element not otherwise specified.

The *position/value* occurrence specifications cannot be specified with either *order* or *number* occurrence specifications. That is, the concepts cannot be combined.

2B.10.8 Components and subcomponents

Many fields and components in versions of HL7 prior to v 2.5 were defined to be **Composite Data (CM)** data types. As of 2.5, all field instances will reference a valid data type other than CM. Addenda for versions 2.3.1 and 2.4 are available that define more precise names for the CM data types. These names allow a more precise data type name for each of the fields using the former CM data type to be more easily used for XML encoding of message instances. Although message profiling is not limited to a specific version of HL7, it is **strongly** encouraged that these new data types be used to increase interoperability between versions.

Each component within composite fields shall be profiled. This requires defining the usage, length, data type, and data values of each of the components. Where there are sub-components of a component, each of the sub-components shall also be profiled using the same method. With the exception of cardinality, the rules for these definitions follow those for fields (See section *2.B.9*, "*Static definition - field level*").

2B.11 MESSAGE PROFILE DOCUMENT

HL7 Headquarters will provide a utility, hereafter called registry, on the Members' Only Web site *(http://www.hl7.org)* where the message profile can be registered.

Messages profiles in the registry are all catalogued with a set of metadata. Those entities submitting message profiles into the registry will need to fill out a form that captures the required metadata information. The registry and the metadata will be documented in an informative document and will not be discussed further in this document.[8]

[8] The message profile metadata has changed between v 2.5 and v 2.6. An attribute to reflect the version of the metadata will be included in the metadata. The registries and tools will need to allow for compatibility between message profile versions. At a minimum, a transform will be available.

2B.11.1 Message profile document format

The Conformance and Guidance for Implementation and Testing (CGIT) Work Group researched the best approach to standardize the format of a message profile to facilitate comparison and measurement. XML (eXtensible Markup Language XML W3C XML 1.0 2nd Ed[9]) documents appeared to be the best tool for this.

This use of XML is not, in any way, related to the HL7 2.xml encoding specification that describes the XML encoding of message instances. The message profile document format provides structure to the documentation of the message profile and does not limit the encoding of an actual message instance.

2B.11.2 Message profile document definition

A message profile document will be a valid HL7 message profiles if it conforms to the constraints expressed in the message profile document definition (See section *2.B.13*. "*Conformity Assessment of Usage Codes*"), and the additional rules described in this document.

2B.12 DOCUMENTATION

2B.12.1 Documentation Hierarchy

The standard provides the foundation for implementation development. The standard includes many options which allows for multiple interpretations and implementations that exhibit differing behavior dependent on the options supported by the implementation. Given the possibilities for variations in implementation behavior, it is essential that vendors claim conformance to the standard with a clear identification of the optional behavior supported.

To emphasize the importance of the above claim and its relationship to the standard the concept of a "documentation hierarchy" is introduced.

2B.12.2 Introduction

According to IEEE (glossary 1990) the term "semantic interoperability" is "the ability of two or more systems or components to **exchange** information and to **use** the information that has been exchanged". The first condition requires that the system is capability of importing information from another system or to export information to another system. Generally testing that systems satisifes this condition is difficult to ascertain and can only be achieved using the actual system that provides or consumes the data. Good documentation though, may serve to ameliorate the problem. One has to be aware, that good documentation is an essential part of all interfaces and that (semantic) interoperability cannot be achieved without sufficient documentation.

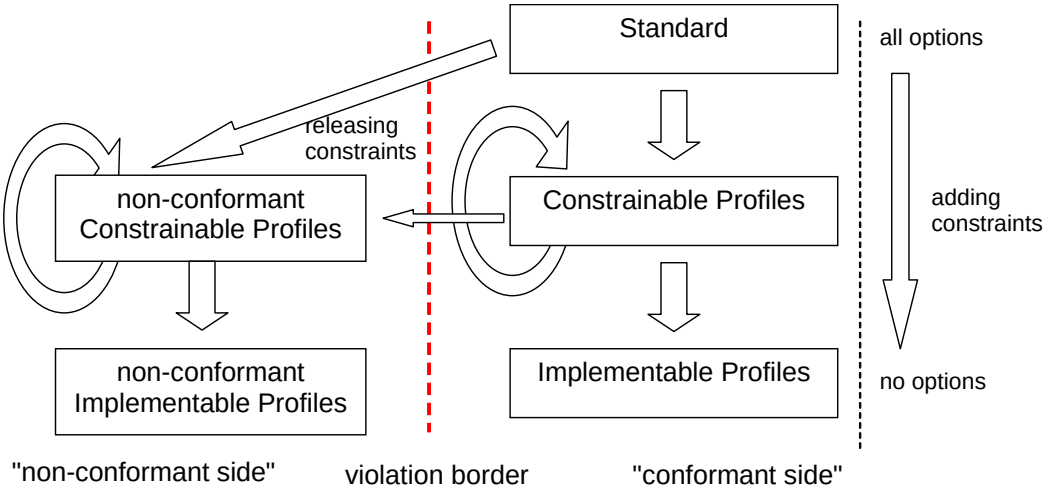
2B.12.3 Problem Space

As mentioned previously, testing outside an environment in which applications are deployed is not practical; otherwise tests are based on assumptions which may not be valid. For example, the configuration, the master files, or the software components in use may differ. Users, (e.g., hospitals) may prefer to make preliminary assessments of an application's interoperability capabilities before committing to purchase. To achieve this goal without investing too many resources, vendors and user need to have different choices available to them.

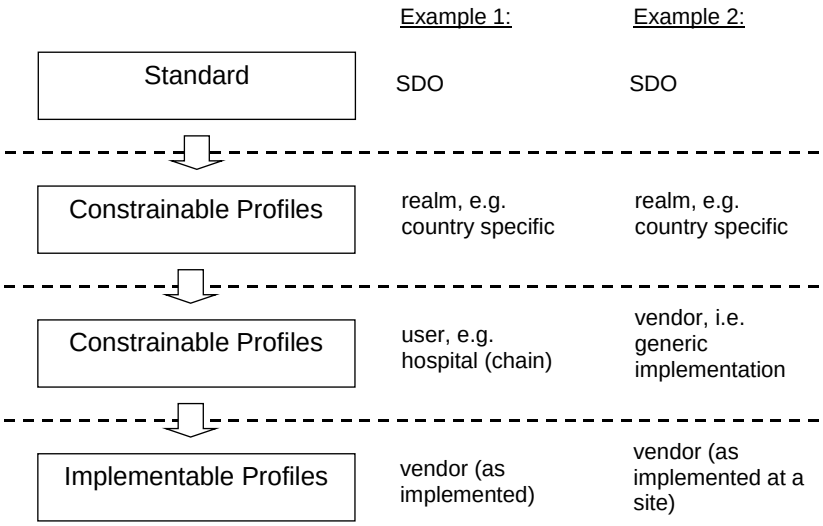
9 Refer to the World Wide Web Consortium web site for this recommendation (http://www.w3.org/TR/REC-xml).

2B.12.4 Hierarchy of Profiles

First, it is useful to recognize that the standard is a profile and profiles that are derived from it introduce a hierarchy of profiles:



Following this hierarchy, profiles can be used by different authorities as explained in the follow illustration:



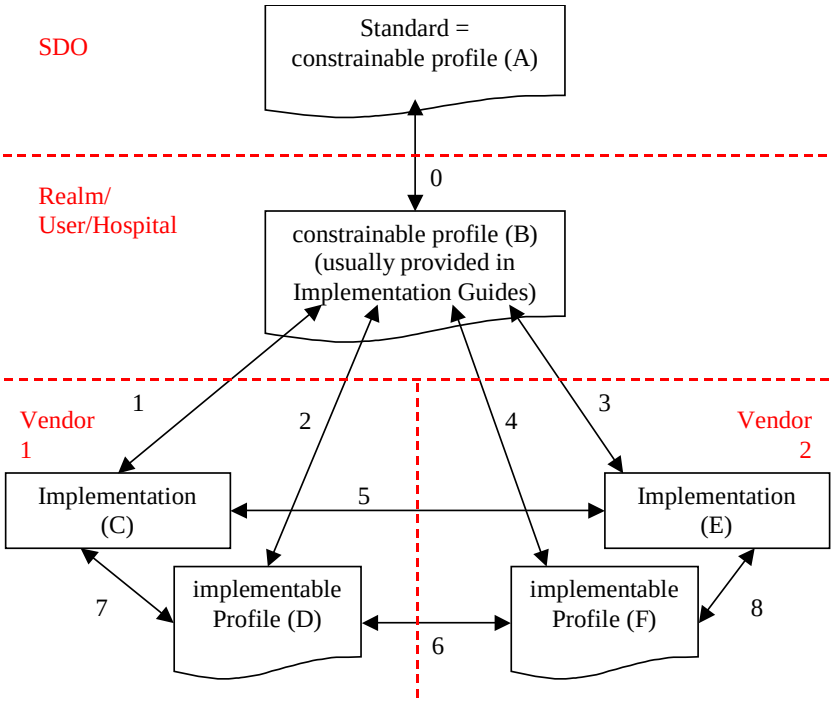
As shown, there are different options for how profiles can be used.

Note 1: The hierarchy of constrainable profiles may contain more than two levels/recursions.

Note 2: IHE offers constrainable profiles which are domain specific. They are also based on national addenda like realm-specific Z-segments. From that perspective they form a two level hierarchy alone.

2B.12.5 Architecture

Therefore, in almost all situations the following component architecture applies. Unfortunately, some of the architectural components are not yet in existence and can therefore be regarded as "virtual". Nevertheless, creating them is straightforward.



Note: Although the relationship shown are for implementation profiles (arrow 6), it equally applies to constrainable profiles.

2B.12.6 Components

The boxes in the diagram above represent the following components:

	Documentation		Implementation
Components	**Constrainable Profile**	**Implementable Profile**	
A	X		
B	X		
C			X
D		X	
E			X
F		X	

The standard and an implementation guide based on this standard both represent a constrainable profile, i.e., they provide a set of requirements and constraints, and both still contain optionality. An implementation guide should introduce additional constraints.

An implementation guide can either be universal (e.g., published by PEOs like IHE), realm specific (e.g., C32 from HIPSP), or site-specific (hospital chain like Kaiser Permanente in the USA or Rhön in Germany).

A vendor—either 1 or 2 in the diagram above—is providing an implementation (C or E) based on that guidance. Usually, these products are accompanied by appropriate documentation (D or F). Based on this notion, these documents should fulfil the requirements for implementation profiles defined in this chapter.

2B.12.7 Relationships

Hence, a relationship among those components exists. For the purpose of testing semantic interoperability, the following are of importance:

Arrow	Linkage	Conformance	Compliance	Compatibility	
0	A – B		X		
1	B – C	X			<==+
2	B – D		X		<==|==+
3	B – E	X			<==+ |
4	B – F		X		<=====+
5	C – E			X	
6	D – F			X	
7	C – D	X			
8	E – F	X			

If two documents or profiles are evaluated for consistency we describe them as compliant (0, 2, and 4). If an implementation is tested against some guidance it is called conformant (1, 3, 7, and 8). These two evaluations are conducted on different levels. If a test is done on the same level against the same type (either documents/profiles or implementation) it is called compatibility (5 and 6).

2B.12.8 Testing Types

The profile architecture diagram above dictates the necessity for different types of testing. The table below provides details of the testing types:

Testing Type	Direction	Test Artifact	Description
Profile Compliance	Vertical	Profiles	Profiles are tested against each other to determine whether one is a constraint of (i.e., consistent with) the other. Profile compliance testing is appropriate when additional constraints are specified to successive profiles in the hierarchy (e.g., standard to a constrainable profile to an implementation profile).
Application Conformance	Vertical	Application	Provides an assessment of how well the application fulfils the requirements specified in a profile. This is conformance testing.
Profile Compatibility	Horizontal	Profiles	Profiles are tested against each other to determine whether the pair can be used by applications to successfully exchange information (interoperate). If a profile pair that constrains the same underlying profile conflict with each other chances of interoperability of applications that implement these profiles are diminished.
Application Interoperability	Horizontal	Applications	Applications are tested with each other to determine whether they can successfully exchange information (interoperate). Applications that implement the same profile or compatible profiles and have successfully passed conformance tests will increase the likelihood of interoperating. IHE connectathons are an example of interoperability testing.

2B.12.9 Documentation Quality

The table given below indicates the level of documentation quality for an implemented interface.

Documentation Quality	Description	Consequences
0 – Undocumented Unsubstantiated Claim	A vendor (or more generically, a document provider) claims conformance to HL7; however the claim is unsubstantiated. Note: This level can be treated as the "standard conformance" as all vendors claim conformance to HL7 per se.	None
1 – Documented Unsubstantiated Claim	A vendor provides evidence of a claim with documentation of the interface. The documentation can be in any format (e.g., a text document) and the contents of the claim are not substantiated. Note: For example, a vendor may copy paragraphs from the original standard; this is an acceptable practice at this level of documentation quality.	Vendor must provide a document.
2 – Documented Standard Unsubstantiated Claim	The documentation provided fulfils the requirements of a conformance profile as listed in HL7 V2 chapter 2B.	Vendor must provide a conformance statement.
3 – Documented Standard Machine Processable Unsubstantiated Claim	The documentation is machine processable. Note: One option is to use a tool (e.g., the MWB) to create this file.	The user or vendor has to provide a computable (constrainable) conformance profile file.
4 - Documented Standard (Implementation Profile Level) Machine Processable Unsubstantiated Claim	The documentation is a conformance profile fulfilling the criteria for implementable profiles, i.e. no options are allowed any more. Note: the criteria listed for level 3 allows a test to verify that the profile fulfils the requirements for implementable profiles.	Vendor must provide a file which fulfils the requirements for implementable conformance profile.
5 - Documented Standard Substantiated Claim	The profile is (successfully) tested against another profile. Note: This testing is done against a higher level (=less constraint) profile, probably a constrainable profile issued either by a vendor, affiliate or HQ itself. Therefore, this testing is done vertically and checks whether the provided profile only add additional constraints.	Requires references to the profiles tested against. Furthermore, the results of the tests must be stated. Furthermore, the means and details of this testing must be specified. As a consequence, it will become obvious, whether the relationship to the underlying profile is "conformant" or "conflicting".
6 - Documented Standard Compatibility Substantiated Claim	Another option of testing is horizontally. This way, two (constrainable or implementable) profiles are tested with a sender/receiver perspective. One profile takes the role of the sender, the other of the receiver. It should be tested, whether this will result in interoperability problems. During the testing, the (machine processable) representations of the profiles are compared with each other. Note: On this level we talk about "compatible" and "incompatible" profiles.	Requires references to the profiles tested against. Furthermore, the role must be mentioned.

2B.12.10 Impacts

A significant advantage of this concept is that it has no impact on implementations because it only addresses documentation. The documentation gives an indication of how close a vendor's implementation is conformant to HL7 V2 requirements. The use of tools for creating such documentation becomes necessary at level 3. Furthermore, at the higher levels the documentation should provide a section listing the profiles; this will enable verification. It is recommended that both the documentation and profile be publicly

available—preferably on the vendor's website[10]. Following the documentation recommendations presented in this chapter will better inform consumers of vendor's products.

2B.12.11 Primary Focus of Requirement

It is recognized that for complex messages satisfying the highest level is challenging. As indicated above, testing profiles using operational implementations is sufficient for satisfying this requirement. A prerequisite is that the documentation reflects the interface behavior. Such verification is only enforceable if such documentation is part of the agreements.

It should be noted that we have discussed vendor claims through documentation; this is only a claim of what they implemented. It is not an indication that the vendor's interface implementation is conformant.

It is also recognized that we are making the assumption that the documented profile has been implemented as an interface. When a user buys an interface they should insist that the documentation be included into the agreement. This ensures that sufficient documentation of the interface functionality of the implementation exists. Such documentation and agreements enable testing. Several tools are available to test messages/interfaces against message profile[11].

2B.12.12 Advantages for Implementers

It is advantageous to test interfaces thoroughly; providing good documentation on the capabilities of the interface facilitates testing and ultimately expedites installation of the interface. A testable profile provides an effective way to check for compatibility problems in advance. Currently, many problems are undetected at installation and are discovered by the customer during use. Specific issues like the maximum length of data elements are generally not tested. Inclusion of robust documentation as described here will shift efforts from accidental testing and support difficulties to automated testing with will increase the likelihood of interoperability.

2B.13 TOOLS

The tools used for creation, sharing, re-use, reporting, analyzing, and comparing message profiles are outside the scope of the HL7 standard. Refer to the Conformance and Guidance for Implementation and Testing (CGIT) Work Group web site for useful links that are of widespread interest to, and in support of, message profiles and the (CGIT) Work Group.[12]

---

[10] DICOM provides an idea of how this can be done. The official IHE website (http://www.ihe.net/Resources/ihe_integration_statements.cfm) also lists the vendors having participated at the connect-a-thon. Each vendor provides a link to his homepage where the integration statements can be found.
It would be ideal, if the vendors could list their conformance statements there in parallel.

[11] For example, in the U.S. the National Institute of Standards and Technology (NIST) Meaningful Use Tools are used to certify products that conform to the HL7 Version 2.5.1 Electronic Laboratory Reporting to Public Health, CDC HL7 Version 2.5.1 Immunization Messaging, CDC HL7 Version 2.5.1 Syndromic Surveillance, and the HL7 Version 2.5.1 Laboratory Results Interface (LRI) Implementation Guides

[12] The Messaging Workbench (MWB) is the first freeware tool available for creating message profiles. This tool is neither developed nor supported by HL7 but its use is widespread among HL7 members.

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.

2B.14 CONFORMITY ASSESSMENT OF USAGE CODES

2B.14.1 Conformity Assessment of Usage Codes

The preceding sections describe conformance constructs used in message profiles. This section seeks to clarify the meaning of the conformance usage codes by providing a context in which the requirements can be assessed. That is, a conformity assessment value is given for each conformance construct while considering the possible states of data presence and conditional outcomes (if applicable). The result is a set of truth tables that will aid readers interpret the meaning of the conformance constructs. Where applicable the assessment is given for the sending and receiving perspective. The information provided in this section can also serve as a guide when conducting conformance testing.

2B.14.2 Usage – Sending Application

This section presents the truth tables for assessing usage code for receiving applications. The following provides an explanation of the heading columns.

Usage Indicator – Usage declared in the conformance profile for the element.

Test Data Provided – Indicates whether or not data was provided for this element in the test data set.

Conformity Assessment Indicator – Indicates the valid action the sender should take when populating the message element.

Actual Data Sent – Indicates the possible behavior, i.e., whether or not the sender populate the element with a value

Conformity Assessment – Indicates the result of the conformity assessment.

Comments – Provides additional insight of the evaluation scenario.

As an example the first entry in the first table can be interpreted as: The sender profile specified the element usage as required (R). Data is available to the sender; therefore the requirement for the sender is to populated the element (i.e., the element shall be present in the message). If the data is present for the element then the application is conformant with respect to the element usage. If the data is not-present for the element then the application is not conformant with respect to usage of the element.

Conformity Assessment of the Required Usage Code for Sending Applications

Usage Indicator	Test Data Provided	Conformity Assessment Indicator	Actual Data Sent	Conformity Assessment	Comments
R	Valued	Present	Present	Conformant	Affirmative test result
			Not-Present	Non-Conformant	Application does not send the required element when a value is provided.
R	Not-Valued	None—expected behavior is that no message is sent	Present	Non-Conformant	Application sends a value even though they do not have a valid value to send. [13]
			Not-Present	Non-Conformant	Application sends a message with a required element not populated.
			No message sent	Conformant	Application correctly detects that they don't have data for a required value and doesn't send the message.

[13] Such "negative" testing is necessary. For example, an application may incorrectly populate an element with a default value.

Conformity Assessment of the Required but may be Empty Usage Code for Sending Applications[14]

Usage Indicator	Test Data Provided	Conformity Assessment Indicator	Actual Data Sent	Conformity Assessment	Comments
RE	Valued	Present	Present	Conformant	Affirmative test result
			Not-Present	Non-Conformant	Application does not send the required element when a value is provided.
RE	Not-Valued	Not-Present	Present	Non-Conformant	Application sends a value even though they do not have a valid value to send. [15]
			Not-Present	Conformant	Application sends a message without the element populated. Affirmative test result.

Conformity Assessment of Not-Supported Usage Code for Sending Applications

Usage Indicator	Test Data Provided	Conformity Assessment Indicator	Actual Data Sent	Conformity Assessment	Comments
X	Valued	Not-Present	Present	Non-Conformant	Non-conformant because value was sent for a not-supported element.
			Not-Present	Conformant	Test case results confirm correct usage of X element by providing data and the application did not send value.
X	Not-Valued	Not-Present	Present	Non-Conformant	Non-conformant because value was sent for a not-supported element.
			Not-Present	Conformant	Confirms expected behavior.

2B.14.2.0 Conditional (C(a/b)) Usage Conformity Assessment

Conformance assessment for elements with conditional usage (i.e., C(a/b)) is dependent on the result of the condition predicate. For example, if conditional usage for an element is specified as C(R/X) and the result of the condition evaluation is true then the usage for the element is R and the conformity assessment table for R-required applies. Likewise, when the result of the condition evaluation is false then the usage for the element is X and the conformity assessment table for X-not supported applies.

When testing elements with conditional usage both the true and false scenarios need to be examined. The conformity assessment table below shows an example for the case C(R/X) for a sending application. Note that when the condition evaluates to true the assessment is the same as the R-required assessment table and when the condition evaluates to false the assessment is the same as the X-not supported. Similar tables can be built for the other combination such as C(R/RE), C(RE/X), etc.

[14] There are multiple interpretations of "RE" when a value is known. One is "the capability must always be supported and a value is sent if known", the other is "the capability must always be supported and a value may or may not be sent even when known based on a condition external to the profile specification. The condition may be noted in the profile but cannot be processed automatically". This is what can be interpreted from the "relevant" part of the definition. Regardless of the interpretation, for the permutations presented here external conditions that may affect a value being sent or not is not considered. Test cases can be developed such that these cases won't exist. For example, a common example of when an element may not be sent is when a patient doesn't authorize it to be sent; in this scenario a pre-condition to the test case is that the patient has authorized consent (Another—and preferred—option is to define separate profiles for the different contents.). Therefore, regardless of the interpretation the "RE" usage code, a set of test circumstances can be developed to sufficiently test the "RE" element. That is, the external condition can't always prevent an element from being sent; otherwise it is not a condition. Hence, "RE" elements can in fact be fully tested in the manner described.

[15] Such "negative" testing is necessary. For example, an application may incorrectly populate an element with a default value.

Example Conformity Assessment of Conditional Usage Code C (R/X) for Sending Applications

Usage Indicator	Test Data Provided	Condition Predicate Result	Conformity Assessment Indicator	Actual Data Sent	Conformity Assessment	Comments
C(R/X)	Valued	True	Present	Present	Conformant	Affirmative test result
				Not-Present	Non-Conformant	Application does not send the required element when a value is provided.
C(R/X)	Not-Valued	True	None—expected behavior is that no message is sent.	Present	Non-Conformant	Application sends a value even though they do not have a valid value to send. [16]
				Not-Present	Non-Conformant	Application sends a message with a required element not populated.
				No Message Sent	Conformant	Application correctly detects that they don't have data for a required value and doesn't send the message.
C(R/X)	Valued	False	Not-Present	Present	Non-Conformant	Non-conformant because value was sent for a not-supported element.
				Not-Present	Conformant	Test case results confirm correct usage of X element by providing data and the application did not send value.
C(R/X)	Not-Valued	False	Not-Present	Present	Non-Conformant	Non-conformant because value was sent for a not-supported element. Value sent when condition is false and no value provided.
				Not-Present	Conformant	Confirms expected behavior.

2B.14.2.1 Optional (O) Usage Conformance Implications

The usage indicator "O-Optional" must be further constrained to another value in order to allow for a conformity assessment. Therefore, no truth table is provided.

2B.14.3 Usage – Receiving Application

This section presents the truth tables for assessing usage code for receiving applications. Below provides an explanation of the heading columns.

Usage Indicator – Usage declared in the conformance profile for the element.

Test Data Sent – Indicates whether or not the element was populated in the test message.

Conformity Assessment Indicator – Indicates how the receiver should respond to the test message with regards to an element.

Receiver Action – Indicates what action the receiver took in response to the test message.

Conformity Assessment – Indicates the result of the conformity assessment.

Comments – Provides additional insight of the evaluation scenario.

[16] Such "negative" testing is necessary. For example, an application may incorrectly populate an element with a default value.

As an example, the first row in the first table can be interpreted as: The receiver profile has specified an element usage as required (R). A test message is created in which the element is populated with a value. The requirement for the receiver is to process the element. If the receiver processed the element then the application is conformant with respect to the element usage. If the application did not process the element then the application is not conformant with respect to the element usage.

Conformity Assessment of the Required Usage Code for Receiving Applications

Usage Indicator	Test Data Sent	Conformity Assessment Indicator	Receiver Action	Conformity Assessment	Comments
R	Valued	Process Element	Processed	Conformant	Affirmative test result
			Not-Processed	Non-Conformant	Application does not process the required element received.
R	Not-Valued	Raise Exception	Not-Processed; Application raises an exception	Conformant	The application shall raise an error when a required element is not sent. The application should not process the message.
			Not processed; no exception raised	Non-Conformant	The application shall raise an error when a required element is not sent. The application should not process the message.
			Processed; Application does not raise an exception	Non-Conformant	Application processes the message and does not raise an error for a required element.

Conformity Assessment of the Required but may be Empty Usage Code for Receiving Applications

Usage Indicator	Test Data Sent	Conformity Assessment Indicator	Receiver Action	Conformity Assessment	Comments
RE	Valued	Process Element	Processed	Conformant	Affirmative test result
			Not-Processed	Non-Conformant	Application does not process the required element received.
RE	Not-Valued	Process Message	Not-Processed and/or the application raises an exception	Non-Conformant	Application should process the message when this element is omitted. In this case the application did not process the message and/or raised an exception.
			Processed; Application does not raise an exception	Conformant	Application processes the message. Affirmative test result.

Conformity Assessment of Not-Supported Usage Code for Receiving Applications

Usage Indicator	Test Data Sent	Conformity Assessment Indicator	Receiver Action	Conformity Assessment	Comments
X	Valued	Don't process the element[17]; Exception may be raised	Element Processed	Non-Conformant	Non-conformant because data was processed for a not-supported element; correct behavior is to not process the element.
			Element Not-Processed	Conformant	Application did not process not-supported element.
X	Not-Valued	Process Message	Application raises an error	Non-Conformant	Application should process the message without raising an exception.
			Processed	Conformant	Confirms expected behavior.

2B.14.3.0 Conditional (C(a/b)) Usage Conformity Assessment

Conformance assessment for elements with conditional usage (i.e., C(a/b)) is dependent on the result of the condition predicate. For example, if conditional usage for an element is specified as C(R/X) and the result of the condition evaluation is true then the usage for the element is R and the conformity assessment table for R-required applies. Likewise, when the result of the condition evaluation is false then the usage for the element is X and the conformity assessment table for X-not supported applies.

When testing elements with conditional usage both the true and false scenarios need to be examined. Conformity assessment tables can be built for the various combinations such as C(R/X), C(R/RE), and so on. See the Conditional Usage Conformity Assessment section for sending application for an example.

2B.14.3.1 Optional (O) Usage Conformance Implications

The usage indicator "O-Optional" must be further constrained to another value in order to allow for a conformity assessment. Therefore, no truth table is provided.

2B.15 MESSAGE PROFILE DOCUMENT DEFINITION

The structure of the message profile document is expressed using the XML Schema Language.[18] The message profile schema is normative in order to express the rules by which the registry will validate .

2B.15.1 Message profile schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    == Changes from previous version (2.7.1)
```

[17] Note: stricter conformance may apply based on business rules associated with the element. For example, the trading partner's conformance agreement included a privacy requirement to not send the patient name for this application. The sender sent the patient name in error. If it is strictly prohibited to send the patient name in this agreement the receiver should reject the message and raise an exception to notify the sender of the error and that the message was not processed. In circumstances where business rules are indifferent for this element the receiver can choose to ignore the data with or without raising an exception. In either case, the receiver shall not process the information. It is important to recognize that for a receiver not-supported element there is no way to distinguish between a strict shall not receive and indifference within the framework of the message profile. Documented business rules are necessary. Such short-comings prevent straight forward analysis and processing.

[18] Refer to the World Wide Web Consortium web site for their recommendations for schema (http://www.w3.org/TR/xmlschema-0, http://www.w3.org/TR/xmlschema-1, and http://www.w3.org/TR/xmlschema-2).

```
    1. The schema has been updated to reflect the changes introduced in
       the version 2.8 of the standard.
            Below is a summary of the changes   (see the conformance section in the
            standard for more details)
           - Use Case element has been replaced with Interaction
           - Cardinality of  HL7v2xStaticDef element has been changed from
             minOccurs = 1 and maxOccurs = unbounded  to minOccurs = 1 and maxOccurs = 1
           - Role attribute has been removed from HL7v2xStaticDef element
    2. Field element has been modified to support
        a) profiling repeating elements (See Profiling Repeating Elements for details)
        b) profiling elements based on a specific component value
-->
<!--
    == Issues
    1. TableLibraryReference is defined as a black box (xsd:any) any valid xml can be
inserted. This may need to be detailed.
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
       <xs:element name="HL7v2xConformanceProfile">
              <xs:annotation>
                     <xs:documentation>An unambiguous specification of one or more
standard HL7 messages that have been analyzed for a particular use case. It prescribes a
set of precise constraints upon one or more standard HL7 messages.</xs:documentation>
              </xs:annotation>
              <xs:complexType>
                     <xs:sequence>
                            <xs:element name="MetaData" type="MetaDataType">
                                   <xs:annotation>
                                          <xs:documentation>Provides descriptive
information about the life-cycle of the HL7v2xConformanceProfile, as well as authorship
and control information.</xs:documentation>
                                   </xs:annotation>
                            </xs:element>
                            <xs:element name="Annotation" type="AnnotationType"
minOccurs="0" maxOccurs="unbounded">
                                   <xs:annotation>
                                          <xs:documentation>Annotations provide a
general description about how the profile is intended to be used, as well as hints on
using or interpreting the profile.</xs:documentation>
                                   </xs:annotation>
                            </xs:element>
                            <xs:element name="Encodings">
                                   <xs:annotation>
                                          <xs:documentation>Identifies all of the
message encoding mechanisms supported by the profile. Non-traditional encoding mechanisms
may be identified if desired.</xs:documentation>
```

```
                                    </xs:annotation>
                                    <xs:complexType>
                                            <xs:sequence>
                                                    <xs:element name="Encoding"
maxOccurs="unbounded">
                                                            <xs:annotation>

        <xs:documentation>Identifies one of the encoding mechanisms supported by the
profile.</xs:documentation>
                                                            </xs:annotation>
                                                            <xs:simpleType>
                                                                    <xs:union
memberTypes="xs:NMTOKEN">
                                                                            <xs:simpleType>

        <xs:restriction base="xs:NMTOKEN">

        <xs:enumeration value="ER7"/>

        <xs:enumeration value="XML"/>

        </xs:restriction>

        </xs:simpleType>
                                                                    </xs:union>
                                                            </xs:simpleType>
                                                    </xs:element>
                                            </xs:sequence>
                                    </xs:complexType>
                                    <xs:key name="EncodingUniqueInEncodings">
                                            <xs:selector xpath="Encoding"/>
                                            <xs:field xpath="."/>
                                    </xs:key>
                            </xs:element>
                            <xs:element name="Interaction">
                                    <xs:complexType>
                                            <xs:sequence>
                                                    <xs:element name="Annotation"
type="AnnotationType" minOccurs="0" maxOccurs="unbounded"/>
                                            </xs:sequence>
                                            <xs:attribute name="Sender" type="xs:string"
use="required"/>
                                            <xs:attribute name="Receiver"
type="xs:string" use="required"/>
                                            <xs:attribute name="Message"
type="MsgTypeType" use="required"/>
                                    </xs:complexType>
                            </xs:element>
```

```
                        <xs:element name="DynamicDef" maxOccurs="unbounded">
                            <xs:annotation>
                                <xs:documentation>The dynamic definition is
an interaction specification for a conversation between 2 or more
systems.</xs:documentation>
                            </xs:annotation>
                            <xs:complexType>
                                <xs:attribute name="AccAck"
type="AcknowledgementType" default="NE">
                                    <xs:annotation>
                                        <xs:documentation>Identifies
when and if HL7 'Accept' acknowledgements are required. Allowed values are: AL (always),
NE (never), SU (on success), ER (on error). Default is 'NE'.</xs:documentation>
                                    </xs:annotation>
                                </xs:attribute>
                                <xs:attribute name="AppAck"
type="AcknowledgementType" default="AL">
                                    <xs:annotation>
                                        <xs:documentation>Identifies
when and if HL7 'Application' acknowledgements are required. Allowed values are: AL
(always), NE (never), SU (on success), ER (on error).  Default is 'AL'.</xs:documentation>
                                    </xs:annotation>
                                </xs:attribute>
                                <xs:attribute name="MsgAckMode"
default="Immediate">
                                    <xs:annotation>
                                        <xs:documentation>Identifies
the type of acknowledgement expected by the sender of a message. Allowed values are:
Immediate and Deferred. Default is Immediate.</xs:documentation>
                                    </xs:annotation>
                                    <xs:simpleType>
                                        <xs:restriction
base="xs:NMTOKEN">
                                            <xs:enumeration
value="Immediate"/>
                                            <xs:enumeration
value="Deferred"/>
                                        </xs:restriction>
                                    </xs:simpleType>
                                </xs:attribute>
                                <xs:attribute name="QueryMessageType"
default="NonQuery">
                                    <xs:annotation>
                                        <xs:documentation>Identifies
whether the message is query-related, and if so, what type of query message it is. Allowed
values are: NonQuery, Query, Response and Publish. Default is NonQuery.
                                </xs:documentation>
                                    </xs:annotation>
                                    <xs:simpleType>
```

```
                                                    <xs:restriction
base="xs:NMTOKEN">
                                                        <xs:enumeration
value="NonQuery"/>
                                                        <xs:enumeration
value="Query"/>
                                                        <xs:enumeration
value="Response"/>
                                                        <xs:enumeration
value="Publish"/>
                                                    </xs:restriction>
                                                </xs:simpleType>
                                            </xs:attribute>
                                            <xs:attribute name="QueryMode"
default="RealTime">
                                                <xs:annotation>
                                                    <xs:documentation>Identifies
the type of query being performed. Allowed values are: Batch, RealTime or
Both.</xs:documentation>
                                                </xs:annotation>
                                                <xs:simpleType>
                                                    <xs:restriction
base="xs:NMTOKEN">
                                                        <xs:enumeration
value="Batch"/>
                                                        <xs:enumeration
value="RealTime"/>
                                                        <xs:enumeration
value="Both"/>
                                                    </xs:restriction>
                                                </xs:simpleType>
                                            </xs:attribute>
                                        </xs:complexType>
                                    </xs:element>
                                    <xs:element ref="HL7v2xStaticDef"/>
                                    <xs:choice>
                                        <xs:element name="TableLibrary"
type="TableLibraryType">
                                            <xs:annotation>
                                                <xs:documentation>The table library
specifies a standardized format to organize the vocabulary and provides support to
reference it (See section X.X.X ). In short, the table library is a container for a
collection of tables. </xs:documentation>
                                            </xs:annotation>
                                        </xs:element>
                                        <xs:element name="TableLibraryReference"
type="TableLibraryReferenceType">
                                            <xs:annotation>
                                                <xs:documentation/>
```

```
                                        </xs:annotation>
                                </xs:element>
                        </xs:choice>
                </xs:sequence>
                <xs:attribute name="HL7Version" default="2.8">
                        <xs:annotation>
                                <xs:documentation>Identifies the HL7 2.x version on
which the profile is based and with which it is expected to comply.</xs:documentation>
                        </xs:annotation>
                        <xs:simpleType>
                                <xs:union memberTypes="xs:NMTOKEN">
                                        <xs:simpleType>
                                                <xs:restriction base="xs:NMTOKEN">
                                                        <xs:enumeration value="2.0"/>
                                                        <xs:enumeration value="2.0D"/>
                                                        <xs:enumeration value="2.1"/>
                                                        <xs:enumeration value="2.2"/>
                                                        <xs:enumeration value="2.3"/>
                                                        <xs:enumeration
value="2.3.1"/>
                                                        <xs:enumeration value="2.4"/>
                                                        <xs:enumeration value="2.5"/>
                                                        <xs:enumeration
value="2.5.1"/>
                                                        <xs:enumeration value="2.6"/>
                                                        <xs:enumeration value="2.7"/>
                                                        <xs:enumeration
value="2.7.1"/>
                                                        <xs:enumeration value="2.8"/>
                                                </xs:restriction>
                                        </xs:simpleType>
                                </xs:union>
                        </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="ProfileType" use="required">
                        <xs:annotation>
                                <xs:documentation>Categorizes the profile into one
of 3 types: HL7 - represents a specific HL7 published standard (may only be submitted by
the HL7 Organization); Constrainable - May contain "Optional" elements which must be
further constrained in order to create implementation profiles; Implementation - Fully
constrained with no optionality (reflects the behavior of a runtime
system)</xs:documentation>
                        </xs:annotation>
                        <xs:simpleType>
                                <xs:restriction base="xs:NMTOKEN">
                                        <xs:enumeration value="Implementation"/>
```

```
                                        <xs:enumeration value="Constrainable"/>
                                </xs:restriction>
                        </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="Identiifer" type="IdentifierType">
                        <xs:annotation>
                                <xs:documentation>A unique identifier for this
specific version of this dynamic profile. If not specified, one will be assigned to the
profile upon submission to a registry.</xs:documentation>
                        </xs:annotation>
                </xs:attribute>
                <xs:attribute name="ProfileSchemaVersion" type="xs:float"
use="required" fixed="2.0">
                        <xs:annotation>
                                <xs:documentation>The schema version of the
profile.</xs:documentation>
                        </xs:annotation>
                </xs:attribute>
        </xs:complexType>
    </xs:element>
    <xs:element name="HL7v2xStaticDef">
        <xs:annotation>
            <xs:documentation>This represents a detailed profile of a single
message. It provides a detailed breakdown of exactly what the message may contain,
including optionality and cardinality.</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element name="MetaData" type="MetaDataType"
minOccurs="0">
                    <xs:annotation>
                        <xs:documentation>Provides descriptive
information about the life-cycle of the HL7 v2x Static Definition, as well as authorship
and control information.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:group ref="MessageGroup"/>
                <xs:element name="Segment" type="SegmentType">
                    <xs:annotation>
                        <xs:documentation>Documents the
characteristics of a single HL7 segment within the context of a particular message or
segment group.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:group ref="SegGroupOrSegmentGrouping"
maxOccurs="unbounded"/>
            </xs:sequence>
```

```
                        <xs:attribute name="MsgType" type="MsgTypeType" use="required">
                                <xs:annotation>
                                        <xs:documentation>The HL7 message type code, as
identified in MSH-9.1 (see HL7 Table 0076 - Message type).</xs:documentation>
                                </xs:annotation>
                        </xs:attribute>
                        <xs:attribute name="EventType" type="EventTypeType" use="required">
                                <xs:annotation>
                                        <xs:documentation>The HL7 event type code, as
identified in MSH-9.2 (see HL7 Table 0003 - Event type)</xs:documentation>
                                </xs:annotation>
                        </xs:attribute>
                        <xs:attribute name="MsgStructID" type="MsgStructIDType">
                                <xs:annotation>
                                        <xs:documentation>The HL7 message structure code, as
identified in MSH-9.3 (see HL7 Table 0354 - Message Structure Type). </xs:documentation>
                                </xs:annotation>
                        </xs:attribute>
                        <xs:attribute name="OrderControl" type="OrderControlType">
                                <xs:annotation>
                                        <xs:documentation>The HL7 Order control code, as
identified in ORC 1 (see HL7 Table 0119 - Order Control Codes).</xs:documentation>
                                </xs:annotation>
                        </xs:attribute>
                        <xs:attribute name="EventDesc" type="NonEmptyStringType"
use="required">
                                <xs:annotation>
                                        <xs:documentation>A description of the event carried
by this message.</xs:documentation>
                                </xs:annotation>
                        </xs:attribute>
                        <xs:attribute name="Identifier" type="IdentifierType"
use="optional">
                                <xs:annotation>
                                        <xs:documentation>A unique identifier for this
specific version of this static definition. If not specified, one will be assigned to the
profile upon submission to a registry.</xs:documentation>
                                </xs:annotation>
                        </xs:attribute>
                </xs:complexType>
        </xs:element>
        <xs:complexType name="MetaDataType">
                <xs:attribute name="Name" type="NonEmptyStringType" use="required">
                        <xs:annotation>
                                <xs:documentation>Provides a name that clearly and concisely
defines the message exchange being profiled.</xs:documentation>
```

```
                        </xs:annotation>
                </xs:attribute>
                <xs:attribute name="OrgName" type="NonEmptyStringType" use="required">
                        <xs:annotation>
                                <xs:documentation>Name of the organization that submitted
the profile.</xs:documentation>
                        </xs:annotation>
                </xs:attribute>
                <xs:attribute name="Version" type="NonEmptyStringType" use="optional">
                        <xs:annotation>
                                <xs:documentation>The version identifier assigned to this
profile by the author. There is no prescribed version numbering scheme.  However 'higher'
versions should generally be interpreted to be more resent.</xs:documentation>
                        </xs:annotation>
                </xs:attribute>
                <xs:attribute name="Status" type="NonEmptyStringType" use="optional">
                        <xs:annotation>
                                <xs:documentation>Status of this profile, as assigned by the
author. There is no prescribed status scheme at this time. Possible values might include:
'Draft', 'Active', 'Superceded', 'Withdrawn'</xs:documentation>
                        </xs:annotation>
                </xs:attribute>
                <xs:attribute name="Topics" type="NonEmptyStringType" use="optional">
                        <xs:annotation>
                                <xs:documentation>This provides a list of key-words that
relate to the profile and that may be useful in profile searches.</xs:documentation>
                        </xs:annotation>
                </xs:attribute>
                <xs:attribute name="MetaVersion" default="2.6">
                        <xs:annotation>
                                <xs:documentation>Identifies the Message Profile version on
which the profile is based and with which it is expected to comply.
                 </xs:documentation>
                        </xs:annotation>
                        <xs:simpleType>
                                <xs:union memberTypes="xs:NMTOKEN">
                                        <xs:simpleType>
                                                <xs:restriction base="xs:NMTOKEN">
                                                        <xs:enumeration value="2.5"/>
                                                        <xs:enumeration value="2.6"/>
                                                        <xs:enumeration value="2.7"/>
                                                        <xs:enumeration value="2.7.1"/>
                                                        <xs:enumeration value="2.8"/>
                                                </xs:restriction>
                                        </xs:simpleType>
```

```
                        </xs:union>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="Context" type="NonEmptyStringType" use="optional">
                    <xs:annotation>
                        <xs:documentation>As defined, in the HL7 Policies and
Procedures Manual, Affiliates will have decision-making authority.  HL7 Affiliates control
Realms.  Realms do not have decision-making authority.  Realms simply represent a
partition of the solution space.  Affiliates choose how the solution space is to be
partitioned by authorizing the creation of zero to many (0..*) Realms.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
            </xs:complexType>
            <xs:group name="SegGroupOrSegmentGrouping">
                <xs:choice>
                    <xs:element name="SegGroup">
                        <xs:annotation>
                            <xs:documentation>Documents the characteristics of a
grouping of HL7 segments within the context of a particular message or segment
group.</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:sequence>
                                <xs:group ref="MessageElementsGroup"/>
                                <xs:group ref="SegGroupOrSegmentGrouping"
maxOccurs="unbounded"/>
                            </xs:sequence>
                            <xs:attribute name="Name" use="required">
                                <xs:annotation>
                                    <xs:documentation>This is the short,
formal name for the group. It appears in the tag name when using the XML Encoding
syntax.</xs:documentation>
                                </xs:annotation>
                                <xs:simpleType>
                                    <xs:restriction base="xs:NMTOKEN">
                                        <xs:pattern value="([A-
Z]|_)+"/>
                                    </xs:restriction>
                                </xs:simpleType>
                            </xs:attribute>
                            <xs:attribute ref="LongName" use="required"/>
                            <xs:attribute ref="Usage" use="required"/>
                            <xs:attribute ref="PredicateTrueUsage"/>
                            <xs:attribute ref="PredicateFalseUsage"/>
                            <xs:attributeGroup
ref="RepeatableElementAttributes"/>
                        </xs:complexType>
```

```
                        </xs:element>
                        <xs:element name="Segment" type="SegmentType">
                                <xs:annotation>
                                        <xs:documentation>Documents the characteristics of a
single HL7 segment within the context of a particular message or segment
group.</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                </xs:choice>
        </xs:group>
        <xs:complexType name="SegmentType">
                <xs:sequence>
                        <xs:group ref="MessageElementsGroup"/>
                        <xs:element name="Field" maxOccurs="unbounded">
                                <xs:annotation>
                                        <xs:documentation>Documents the characteristics of a
single HL7 field within the context of a particular message segment.</xs:documentation>
                                </xs:annotation>
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:group ref="LeafMessageElementsGroup"/>
                                                <xs:element name="Occurrence" minOccurs="0"
maxOccurs="unbounded">
                                                        <xs:complexType>
                                                                <xs:sequence>
                                                                        <xs:element
name="Component" minOccurs="0" maxOccurs="unbounded">
                                                                                <xs:annotation>

        <xs:documentation>Documents the characteristics of a single component within the
context of a field.</xs:documentation>

        </xs:annotation>

        <xs:complexType>

        <xs:sequence>

        <xs:group ref="LeafMessageElementsGroup"/>

        <xs:element name="SubComponent" minOccurs="0" maxOccurs="unbounded">

        <xs:annotation>

                <xs:documentation>Documents the characteristics of a single sub-component
within the context of a component.</xs:documentation>

        </xs:annotation>
```

```
        <xs:complexType>
                <xs:group ref="LeafMessageElementsGroup"/>
                <xs:attributeGroup ref="LeafElementAttributes"/>
        </xs:complexType>
        </xs:element>
        </xs:sequence>
        <xs:attribute name="Number"
type="xs:nonNegativeInteger"/>
                                                        <xs:attribute name="Value"
type="xs:string"/>
                                                </xs:complexType>
                                        </xs:element>
                                </xs:sequence>
                                <xs:attributeGroup
ref="RepeatableElementAttributes"/>
                                <xs:attributeGroup ref="LeafElementAttributes"/>
                                <xs:attribute name="ItemNo">
                                        <xs:annotation>
                                                <xs:documentation>The HL7-assigned
item number corresponding with the semantic meaning of the field.</xs:documentation>
                                        </xs:annotation>
                                        <xs:simpleType>
                                                <xs:restriction base="xs:NMTOKEN">
                                                        <xs:pattern value="\d{5}"/>
                                                </xs:restriction>
                                        </xs:simpleType>
                                </xs:attribute>
                                <xs:attribute name="Order" type="xs:boolean">
                                        <xs:annotation>
                                                <xs:documentation>Specifies if order
apply or not when profiling repeating fields.</xs:documentation>
                                        </xs:annotation>
                                </xs:attribute>
                                <xs:attribute name="Position"
type="xs:nonNegativeInteger">
                                        <xs:annotation>
                                                <xs:documentation>Specifies the
```

```
position of the component which value is referred to in the Occurrence element
</xs:documentation>
                                        </xs:annotation>
                                    </xs:attribute>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                        <xs:attribute name="Name" type="SegmentNameType" use="required">
                            <xs:annotation>
                                <xs:documentation>This is the short, formal name for the
segment. It is used to identify the segment in both ER7 and XML
encodings.</xs:documentation>
                            </xs:annotation>
                        </xs:attribute>
                        <xs:attribute ref="LongName"/>
                        <xs:attribute ref="Usage" use="required"/>
                        <xs:attribute ref="PredicateTrueUsage"/>
                        <xs:attribute ref="PredicateFalseUsage"/>
                        <xs:attributeGroup ref="RepeatableElementAttributes"/>
                </xs:complexType>
                <xs:complexType name="AnnotationType">
                        <xs:simpleContent>
                            <xs:extension base="NonEmptyStringType">
                                <xs:attributeGroup ref="AnnotationAttributes"/>
                            </xs:extension>
                        </xs:simpleContent>
                </xs:complexType>
                <xs:attributeGroup name="LeafElementAttributes">
                        <xs:attribute name="Name" type="NonEmptyStringType" use="required">
                            <xs:annotation>
                                <xs:documentation>The descriptive name for the
field/component/sub-component</xs:documentation>
                            </xs:annotation>
                        </xs:attribute>
                        <xs:attribute ref="Usage" use="required"/>
                        <xs:attribute ref="PredicateTrueUsage"/>
                        <xs:attribute ref="PredicateFalseUsage"/>
                        <xs:attribute name="Datatype" type="DatatypeType" use="required">
                            <xs:annotation>
                                <xs:documentation>Identifies the HL7 datatype associated
with the element.</xs:documentation>
                            </xs:annotation>
                        </xs:attribute>
                        <xs:attribute name="MinLength" type="xs:positiveInteger">
                            <xs:annotation>
```

```
                        <xs:documentation>Identifies the minimum allowed length for
the content of the element.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="MaxLength" type="xs:positiveInteger">
                    <xs:annotation>
                        <xs:documentation>Identifies the maximum allowed length for
the content of the element.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="ConformanceLength" type="xs:positiveInteger">
                    <xs:annotation>
                        <xs:documentation>The minimum length an application must be
able to handle</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="Truncation" type="xs:boolean">
                    <xs:annotation>
                        <xs:documentation>whether the truncation pattern does/may
apply</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="Table" type="TableType">
                    <xs:annotation>
                        <xs:documentation>Identifies the name of the table
associated with the content of this element.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="ConstantValue" type="NonEmptyStringType">
                    <xs:annotation>
                        <xs:documentation>Identifies the fixed value associated with
this element</xs:documentation>
                    </xs:annotation>
                    <!-- Can only have constant values for leaf elements -->
                </xs:attribute>
        </xs:attributeGroup>
        <xs:attributeGroup name="RepeatableElementAttributes">
                <xs:attribute name="Min" type="xs:nonNegativeInteger" use="required">
                    <xs:annotation>
                        <xs:documentation>This identifies the minimum number of
repetitions of the element that are permitted in a message instance. This attribute should
only be specified if the minimum number of repetitions is greater than 1, as the minimum
for other elements is always '0'.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="Max" use="required">
```

```
                        <xs:annotation>
                                <xs:documentation>This identifies the maximum number of
repetitions of the element that are permitted in a message instance. This attribute should
only be specified if the maximum number of repetitions is greater than 1 and differs from
the minimum attribute (i.e. the maximum number of repetitions is greater than the minimum
number of repetitions). The special value '*' may be used to represent 'unlimited'
repetitions.</xs:documentation>
                        </xs:annotation>
                        <xs:simpleType>
                                <xs:union>
                                        <xs:simpleType>
                                                <xs:restriction base="xs:positiveInteger">
                                                        <xs:minInclusive value="1"/>
                                                </xs:restriction>
                                        </xs:simpleType>
                                        <xs:simpleType>
                                                <xs:restriction base="xs:string">
                                                        <xs:enumeration value="*"/>
                                                </xs:restriction>
                                        </xs:simpleType>
                                </xs:union>
                        </xs:simpleType>
                </xs:attribute>
        </xs:attributeGroup>
        <xs:attributeGroup name="AnnotationAttributes">
                <xs:attribute name="Type" use="required">
                        <xs:annotation>
                                <xs:documentation>
                 </xs:documentation>
                        </xs:annotation>
                        <xs:simpleType>
                                <xs:union memberTypes="xs:NMTOKEN">
                                        <xs:simpleType>
                                                <xs:restriction base="xs:NMTOKEN">
                                                        <xs:enumeration value="ImpNote">
                                                                <xs:annotation>

        <xs:documentation>Implementation Notes provide a general description about how the
element is intended to be used, as well as hints on using or interpreting the
it.</xs:documentation>
                                                                </xs:annotation>
                                                        </xs:enumeration>
                                                        <xs:enumeration value="Description">
                                                                <xs:annotation>
                                                                        <xs:documentation> An
explanation of the associated element.</xs:documentation>
```

```
                                                        </xs:annotation>
                                                </xs:enumeration>
                                                <xs:enumeration value="Definition">
                                                        <xs:annotation>
                                                                <xs:documentation> An
explanation of the meaning of the element.</xs:documentation>
                                                        </xs:annotation>
                                                </xs:enumeration>
                                                <xs:enumeration
value="DesignComment">
                                                        <xs:annotation>
                                                                <xs:documentation>
Internal development notes about why particular design decisions were made, outstanding
issues and remaining work.  They may contain formatting markup.  Not intended for external
publication.</xs:documentation>
                                                        </xs:annotation>
                                                </xs:enumeration>
                                                <xs:enumeration value="Other">
                                                        <xs:annotation>
                                                                <xs:documentation>
Additional content related to the element.</xs:documentation>
                                                        </xs:annotation>
                                                </xs:enumeration>
                                                <xs:enumeration value="Example">
                                                        <xs:annotation>
                                                                <xs:documentation> An
example instance of the element.</xs:documentation>
                                                        </xs:annotation>
                                                </xs:enumeration>
                                        </xs:restriction>
                                </xs:simpleType>
                        </xs:union>
                </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="OtherIdentifier" type="NonEmptyStringType">
                <xs:annotation>
                        <xs:documentation/>
                </xs:annotation>
        </xs:attribute>
</xs:attributeGroup>
<xs:group name="MessageGroup">
        <xs:sequence>
                <xs:element name="Annotation" type="AnnotationType" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:element name="Description" type="NonEmptyStringType"
minOccurs="0">
```

```
                        <xs:annotation>
                                <xs:documentation>Provides an explanation or
definition of what the element represents.</xs:documentation>
                        </xs:annotation>
                </xs:element>
                <xs:element name="Reference" minOccurs="0">
                        <xs:annotation>
                                <xs:documentation>Identifies external sources or
other locations within the profile where additional information can be found about this
item.</xs:documentation>
                        </xs:annotation>
                        <xs:simpleType>
                                <xs:restriction base="NonEmptyStringType"/>
                        </xs:simpleType>
                </xs:element>
        </xs:sequence>
    </xs:group>
    <xs:group name="MessageElementsGroup">
        <xs:sequence>
                <xs:group ref="MessageGroup"/>
                <xs:element name="Predicate" minOccurs="0">
                        <xs:annotation>
                                <xs:documentation>Identifies the conditionality rule
for this element, if applicable</xs:documentation>
                        </xs:annotation>
                        <xs:simpleType>
                                <xs:restriction base="NonEmptyStringType"/>
                        </xs:simpleType>
                </xs:element>
        </xs:sequence>
    </xs:group>
    <xs:group name="LeafMessageElementsGroup">
        <xs:sequence>
                <xs:group ref="MessageElementsGroup"/>
                <xs:element name="DataValues" minOccurs="0" maxOccurs="unbounded">
                        <xs:complexType>
                                <xs:attribute name="ExValue"
type="NonEmptyStringType">
                                        <xs:annotation>
                                                <xs:documentation>Identifies an
individual example value.</xs:documentation>
                                        </xs:annotation>
                                </xs:attribute>
                        </xs:complexType>
                </xs:element>
```

```
                </xs:sequence>
        </xs:group>
        <xs:attribute name="LongName" type="NonEmptyStringType">
                <xs:annotation>
                        <xs:documentation>This is the descriptive name for the element. It
does not appear in any encodings.</xs:documentation>
                </xs:annotation>
        </xs:attribute>
        <xs:attribute name="Usage">
                <xs:annotation>
                        <xs:documentation>Usage identifies the circumstances under which an
element appears in a message. Possible values are:
                 R - Required (must always be present);
                 RE - Required or Empty (must be present if available);
                 O - Optional (no guidance on when the element should appear);
                 C - Conditional (the element is required or allowed to be present when the
condition specified in the Predicate element is true);
                 X - Not supported (the element is not supported)
             </xs:documentation>
                </xs:annotation>
                <xs:simpleType>
                        <xs:restriction base="xs:NMTOKEN">
                                <xs:enumeration value="R"/>
                                <xs:enumeration value="RE"/>
                                <xs:enumeration value="O"/>
                                <xs:enumeration value="C"/>
                                <xs:enumeration value="X"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="PredicateTrueUsage">
                <xs:annotation>
                        <xs:documentation>PredicateTrueUsage is used in combination with a
conditional "C" usage . It specifies how the element usage should be interpreted when the
condition predicate evaluates to TRUE. Possible values are :
                 R - Required (must always be present);
                 RE - Required or Empty (must be present if available);
                 O - Optional (no guidance on when the element should appear);
                 X - Not supported (the element is not supported)
             </xs:documentation>
                </xs:annotation>
                <xs:simpleType>
                        <xs:restriction base="xs:NMTOKEN">
                                <xs:enumeration value="R"/>
```

```
                        <xs:enumeration value="RE"/>
                        <xs:enumeration value="O"/>
                        <xs:enumeration value="X"/>
                    </xs:restriction>
                </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="PredicateFalseUsage">
                <xs:annotation>
                    <xs:documentation>PredicateFalseUsage is used in combination with a
conditional "C" usage . It specifies how the element usage should be interpreted when the
condition predicate evaluates to FALSE. Possible values are :
                     R - Required (must always be present);
                     RE - Required or Empty (must be present if available);
                     O - Optional (no guidance on when the element should appear);
                     X - Not supported (the element is not supported)
                </xs:documentation>
                </xs:annotation>
                <xs:simpleType>
                        <xs:restriction base="xs:NMTOKEN">
                                <xs:enumeration value="R"/>
                                <xs:enumeration value="RE"/>
                                <xs:enumeration value="O"/>
                                <xs:enumeration value="X"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:attribute>
        <xs:simpleType name="NonEmptyStringType">
                <xs:restriction base="xs:string">
                        <xs:minLength value="1"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="AcknowledgementType">
                <xs:restriction base="xs:NMTOKEN">
                        <xs:enumeration value="AL"/>
                        <xs:enumeration value="NE"/>
                        <xs:enumeration value="SU"/>
                        <xs:enumeration value="ER"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="IdentifierType">
                <xs:restriction base="xs:NMTOKEN">
                        <xs:pattern value="(0|[1-9][0-9]*)(\.(0|[1-9][0-9]*))*"/>
                </xs:restriction>
```

```
        </xs:simpleType>
        <xs:simpleType name="MsgTypeType">
                <xs:restriction base="xs:NMTOKEN">
                        <xs:pattern value="[A-Z0-9]{3}"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="EventTypeType">
                <xs:restriction base="xs:NMTOKEN">
                        <xs:pattern value="[A-Z0-9]{3}"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="MsgStructIDType">
                <xs:restriction base="xs:NMTOKEN">
                        <xs:pattern value="[A-Z0-9]{3}(_[A-Z0-9]{3})?"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="OrderControlType">
                <xs:restriction base="xs:NMTOKEN">
                        <xs:pattern value="[A-Z]{2}"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="DatatypeType">
                <xs:restriction base="xs:NMTOKEN">
                        <xs:minLength value="1"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="SegmentNameType">
                <xs:restriction base="xs:NMTOKEN">
                        <xs:pattern value="[A-Z][A-Z0-9]{2}"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="TableType">
                <xs:restriction base="xs:NMTOKEN">
                        <xs:minLength value="1"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:complexType name="TableLibraryType">
                <xs:sequence>
                        <xs:element name="TableDefinition" type="TableDefinitionType"
maxOccurs="unbounded">
                                <xs:annotation>
                                        <xs:documentation>A table definition consists of
metadata describing the table and specifies a list of code/value pairs.  The table
definition metadata consists of a table identifier, OID, name, type, version, and code
```

```
system. </xs:documentation>
                        </xs:annotation>
                    </xs:element>
                </xs:sequence>
                <xs:attribute name="Name" type="xs:string" use="required">
                    <xs:annotation>
                        <xs:documentation>The name of the table
library.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="OrganizationName" type="xs:string">
                    <xs:annotation>
                        <xs:documentation>The organization that created the
library.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="TableLibraryVersion" type="xs:string">
                    <xs:annotation>
                        <xs:documentation>The version of the table
library.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="Status" type="xs:string">
                    <xs:annotation>
                        <xs:documentation>The status of the table
library.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="TableLibraryIdentifier" type="xs:string"
use="required">
                    <xs:annotation>
                        <xs:documentation>A unique identifier for the table
library.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="Description" type="xs:string">
                    <xs:annotation>
                        <xs:documentation>A text description of the table
library.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
            </xs:complexType>
            <xs:complexType name="TableLibraryReferenceType">
                <xs:sequence>
                    <xs:any maxOccurs="unbounded"/>
                </xs:sequence>
```

```
        </xs:complexType>
        <xs:complexType name="TableDefinitionType">
                <xs:sequence>
                        <xs:element name="TableElement" type="TableElementType"
maxOccurs="unbounded">
                                <xs:annotation>
                                        <xs:documentation>Table elements express code/value
pairs and descriptive information about the code/value pair</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                </xs:sequence>
                <xs:attribute name="Identifier" type="xs:string" use="required">
                        <xs:annotation>
                                <xs:documentation>The table identifier.</xs:documentation>
                        </xs:annotation>
                </xs:attribute>
                <xs:attribute name="OID" type="xs:string" use="required">
                        <xs:annotation>
                                <xs:documentation>An OID that identify the table, not the
codesystem.</xs:documentation>
                        </xs:annotation>
                </xs:attribute>
                <xs:attribute name="Name" type="xs:string" use="required">
                        <xs:annotation>
                                <xs:documentation>A descriptive name of the
table.</xs:documentation>
                        </xs:annotation>
                </xs:attribute>
                <xs:attribute name="Type" use="required">
                        <xs:annotation>
                                <xs:documentation>The type of the table as described in
section 2.6.3.6
                    Valid identifiers for a table type are HL7, User, Local, External, and
Imported.  </xs:documentation>
                        </xs:annotation>
                        <xs:simpleType>
                                <xs:restriction base="xs:string">
                                        <xs:enumeration value="HL7"/>
                                        <xs:enumeration value="User"/>
                                        <xs:enumeration value="Local"/>
                                        <xs:enumeration value="External"/>
                                        <xs:enumeration value="Imported"/>
                                </xs:restriction>
                        </xs:simpleType>
                </xs:attribute>
```

```
                <xs:attribute name="Version" type="xs:string" use="required">
                        <xs:annotation>
                                <xs:documentation>The version of the
table.</xs:documentation>
                        </xs:annotation>
                </xs:attribute>
                <xs:attribute name="CodeSystem" type="xs:string" use="required">
                        <xs:annotation>
                                <xs:documentation>A code system as specified in HL7 table
0396 </xs:documentation>
                        </xs:annotation>
                </xs:attribute>
        </xs:complexType>
        <xs:complexType name="TableElementType">
                <xs:attribute name="Code" type="xs:string" use="required">
                        <xs:annotation>
                                <xs:documentation>The code for the data
value.</xs:documentation>
                        </xs:annotation>
                </xs:attribute>
                <xs:attribute name="DisplayName" type="xs:string" use="required">
                        <xs:annotation>
                                <xs:documentation>The long description of the
code.</xs:documentation>
                        </xs:annotation>
                </xs:attribute>
                <xs:attribute name="Source" use="required">
                        <xs:annotation>
                                <xs:documentation>The source of the code/value
pair.</xs:documentation>
                        </xs:annotation>
                        <xs:simpleType>
                                <xs:restriction base="xs:string">
                                        <xs:enumeration value="HL7"/>
                                        <xs:enumeration value="Local"/>
                                        <xs:enumeration value="Redefined">
                                                <xs:annotation>
                                                        <xs:documentation>Redefined means the
code/display name has been changed from its original value.</xs:documentation>
                                                </xs:annotation>
                                        </xs:enumeration>
                                        <xs:enumeration value="SDO">
                                                <xs:annotation>
                                                        <xs:documentation>SDO means Standard
Development Organization.</xs:documentation>
```

```
                                        </xs:annotation>
                                    </xs:enumeration>
                                </xs:restriction>
                            </xs:simpleType>
                        </xs:attribute>
                    </xs:complexType>
                </xs:schema>
```

2B.15.2 Table Librarydocument definition

The structure of the table library document is expressedusing the XML Schema Language. The table library schema is normative in order to express the rules by which the registry will validate.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:element name="TableLibrary" type="TableLibraryType"/>
        <xs:complexType name="TableLibraryType">
                <xs:sequence>
                        <xs:element name="TableDefinition" type="TableDefinitionType"
maxOccurs="unbounded">
                            <xs:annotation>
                                    <xs:documentation>A table definition consists of metadata
describing the table and specifies a list code/value pairs. The table definition metadata
consists of a table identifier, name, type, version, and code system. </xs:documentation>
                            </xs:annotation>
                        </xs:element>
                </xs:sequence>
                <xs:attribute name="Name" type="xs:string" use="required">
                        <xs:annotation>
                                <xs:documentation>The name of the table library.</xs:documentation>
                        </xs:annotation>
                </xs:attribute>
                <xs:attribute name="OrganizationName" type="xs:string">
                        <xs:annotation>
                                <xs:documentation>The organization that created the
library.</xs:documentation>
                        </xs:annotation>
                </xs:attribute>
                <xs:attribute name="HL7Version" type="xs:string" use="required">
                        <xs:annotation>
                                <xs:documentation>The HL7 Version of the table
definitions.</xs:documentation>
                        </xs:annotation>
                </xs:attribute>
                <xs:attribute name="TableLibraryVersion" type="xs:string">
                        <xs:annotation>
```

```
                        <xs:documentation>The version of the table
library.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="Status" type="xs:string">
                    <xs:annotation>
                        <xs:documentation>The status of the table
library.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="TableLibraryIdentifier" type="xs:string" use="required"/>
                <xs:attribute name="Description" type="xs:string">
                    <xs:annotation>
                        <xs:documentation>A text description of the table
library.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
            </xs:complexType>
            <xs:complexType name="TableDefinitionType">
                <xs:sequence>
                    <xs:element name="TableElement" type="TableElementType"
maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Table elements express code/value pairs
and descriptive information about the code/value pair</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                </xs:sequence>
                <xs:attribute name="Identifier" type="xs:string" use="required">
                    <xs:annotation>
                        <xs:documentation>The table identifier.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="Name" type="xs:string" use="required">
                    <xs:annotation>
                        <xs:documentation>The name of the table.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="Version" type="xs:string" use="required">
                    <xs:annotation>
                        <xs:documentation>The version of the specific
table.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="CodeSys" type="xs:string" use="required">
```

```
                        <xs:annotation>
                                <xs:documentation>A code system as specified in HL7 table 0396
</xs:documentation>
                        </xs:annotation>
                </xs:attribute>
                <xs:attribute name="Type" use="required">
                        <xs:annotation>
                                <xs:documentation>The table type is a recognized HL7 table as
described in section 2.6.3.6. Valid identifiers for a table type are HL7, User, Local, External,
and Imported.  </xs:documentation>
                        </xs:annotation>
                        <xs:simpleType>
                                <xs:restriction base="xs:string">
                                        <xs:enumeration value="HL7"/>
                                        <xs:enumeration value="User"/>
                                        <xs:enumeration value="Local"/>
                                        <xs:enumeration value="External"/>
                                        <xs:enumeration value="Imported"/>
                                </xs:restriction>
                        </xs:simpleType>
                </xs:attribute>
        </xs:complexType>
        <xs:complexType name="TableElementType">
                <xs:attribute name="Code" type="xs:string" use="required">
                        <xs:annotation>
                                <xs:documentation>The code for the data value.</xs:documentation>
                        </xs:annotation>
                </xs:attribute>
                <xs:attribute name="DisplayName" type="xs:string" use="required">
                        <xs:annotation>
                                <xs:documentation>The long description of the
code.</xs:documentation>
                        </xs:annotation>
                </xs:attribute>
                <xs:attribute name="Source" use="required">
                        <xs:annotation>
                                <xs:documentation>The source of the code/value
pair.</xs:documentation>
                        </xs:annotation>
                        <xs:simpleType>
                                <xs:restriction base="xs:string">
                                        <xs:enumeration value="HL7"/>
                                        <xs:enumeration value="Local"/>
                                        <xs:enumeration value="Redefined"/>
```

```
                                    <xs:enumeration value="SDO"/>
                                </xs:restriction>
                            </xs:simpleType>
                        </xs:attribute>
                </xs:complexType>
</xs:schema>
```