



# **HL7 Templates Standard: Specification and Use of Reusable Information Constraint Templates, Release 1**

January 2014

HL7 DSTU

Sponsored by:  
Templates Work Group

Copyright © 2013 Health Level Seven International ® ALL RIGHTS RESERVED. The reproduction of this material in any form is strictly forbidden without the written permission of the publisher. HL7 and Health Level Seven are registered trademarks of Health Level Seven International. Reg. U.S. Pat & TM Off.

Use of this material is governed by HL7's IP Compliance Policy.

## Acknowledgments

The authors of this document wish to recognize the following participants who contributed their time and expertise to the development of this guide.

## Copyright

This document is © 2013 Health Level Seven International, All rights reserved.

This material includes SNOMED Clinical Terms ® (SNOMED CT®) which is used by permission of the International Health Terminology Standards Development Organization (IHTSDO). All rights reserved. SNOMED CT was originally created by The College of American Pathologists. "SNOMED ®" and "SNOMED CT ®" are registered trademarks of the IHTSDO.

This material contains content from LOINC® (<http://loinc.org>). The LOINC table, LOINC codes, and LOINC panels and forms file are copyright (c) 1995-2013, Regenstrief Institute, Inc. and the Logical Observation Identifiers Names and Codes (LOINC) Committee and available at no cost under the license at <http://loinc.org/terms-of-use>.

<b>Primary Editor/ Co-Chair</b>	Dr. Kai U. Heitmann Heitmann Consulting and Services, HL7 Germany hl7@kheitmann.de
<b>Co-Chair</b>	Mark Shafarman Shafarman Consulting mark.shafarman@earthlink.net
<b>Co-Chair</b>	John Roberts Tennessee Department of Health john.a.roberts@tn.gov
<b>Co-Editor</b>	Jane Curry Health Information Strategies janecurry@healthinfostrategies.com
<b>Co-Editor</b>	Lisa Nelson Life Over Time Solutions lisaRNelson@cox.net
<b>Contributor</b>	Andy Stechishin CANA Software & Services Ltd. andy.stechishin@gmail.com
<b>Contributor</b>	Keith W. Boone GE Healthcare keith.boone@ge.com
<b>Contributor</b>	Rick Smithies NProgram Ltd rik@nprogram.co.uk

# Table of Contents

<b>1. Introduction .....</b>	<b>8</b>
1.1. Purpose.....	8
1.2. Scope.....	8
1.3. Ballot Status of the Document.....	8
1.4. Audience .....	9
1.4.1. <i>Requisite Knowledge</i> .....	9
1.5. Organization of the Guide .....	9
1.6. Conventions.....	10
1.7. Contributions .....	10
<b>2. Templates and Template Versions .....</b>	<b>11</b>
2.1. What is a template? .....	11
2.2. What is a template version?.....	11
2.3. Why are templates needed?.....	13
2.4. Why is template versioning needed? .....	14
2.5. How does template versioning work? .....	14
2.6. Where do templates help? .....	15
2.7. How does a template design get defined? .....	15
2.7.1. <i>Template Designer</i> .....	15
2.7.2. <i>Governance Group</i> .....	16
2.7.3. <i>Template Repository</i> .....	16
2.7.4. <i>Template Registry</i> .....	16
2.8. Characteristics of a template and its versions .....	16
2.9. Template Meta data.....	17
2.9.1. <i>Identifier</i> .....	17
2.9.2. <i>Name</i> .....	17
2.9.3. <i>Effective Date</i> .....	17
2.9.4. <i>Version Label</i> .....	17
2.9.5. <i>Expiration Date</i> .....	18
2.9.6. <i>Official Release Date</i> .....	18
2.9.7. <i>Status</i> .....	18
2.9.8. <i>Additional meta data</i> .....	20
2.10. Template Body .....	22
2.10.1. <i>Types of constraints</i> .....	22
2.10.2. <i>Inclusion and Containment</i> .....	28
2.10.3. <i>Choice</i> .....	28
2.10.4. <i>Co-occurrence</i> .....	28
2.10.5. <i>STATIC vs DYNAMIC artifact binding</i> .....	29
2.11. Open versus Closed Templates.....	29
2.12. Types of Templates .....	30
2.12.1. <i>CDA Document Level</i> .....	30
2.12.2. <i>CDA Header Level</i> .....	30
2.12.3. <i>CDA Section Level</i> .....	30
2.12.4. <i>CDA Entry Level</i> .....	30
2.12.5. <i>Message Level</i> .....	30
2.12.6. <i>Clinical Statement Level</i> .....	30
2.12.7. <i>Data Type Level (= Data Type Flavors)</i> .....	30
2.12.8. <i>FHIR Resource Profiles</i> .....	31
2.12.9. <i>v2.xml Profiles</i> .....	31
2.13. Template Design Considerations.....	31
2.14. Template Item Table Example.....	31

<b>3. Template Relationships .....</b>	<b>33</b>
3.1. What are template relationships? .....	33
3.2. Why are template relationships useful? .....	33
3.3. How do template relationships work? .....	33
3.4. Types of relationships between templates.....	33
3.4.1. Replacement (replaces).....	33
3.4.2. Version.....	33
3.4.3. Specializations / Generalizations (specializes, generalizes).....	33
3.4.4. Design Copy (copies).....	34
3.4.5. Adaptation (based on) .....	34
3.4.6. Equivalency (equals).....	34
3.4.7. Backwards compatibility.....	34
3.4.8. Forward compatibility.....	34
3.4.9. Containment .....	34
3.4.10. Inclusion.....	35
3.5. Relationships to other artifacts.....	35
3.5.1. Relationship to Underlying model(s) .....	35
3.5.2. Subscribe with “Adopter” .....	35
<b>4. Using Templates to Create, Validate and Consume Instances .....</b>	<b>36</b>
4.1. Templates as Definition of Clinical Concepts.....	36
4.2. Construction of Instances .....	36
4.3. Validation of Instances.....	37
4.4. Processing of Instances.....	37
4.5. Instance Content Creators.....	37
4.6. Instance Content Consumers.....	37
4.7. Content Validator Methods .....	38
4.8. The use of multiple Template Ids .....	38
4.9. Implementation principles and use of template ids in instances.....	39
<b>5. Use Cases Demonstrating the Creation, Use, Maintenance and Governance of Templates.....</b>	<b>40</b>
5.1. Creation and life cycle of template “Estimated Delivery Date” .....	40
5.1.1. Step 1: initial draft.....	40
5.1.2. Step 2: review phase.....	40
5.1.3. Step 3: revision with a minor change.....	41
5.1.4. Step 4: revision with substantial changes.....	42
<b>6. Management and Governance of Templates.....</b>	<b>43</b>
6.1. Role of governance groups.....	43
6.2. Template Repository .....	43
6.3. Template creation.....	43
6.3.1. Use cases / business requirements .....	43
6.3.2. Template content provider skills.....	43
6.3.3. Mapping clinical requirements to technical artifacts.....	44
6.4. Template endorsement, publication, testing and maintenance.....	44
6.5. Re-use of templates.....	44
6.5.1. Referencing templates.....	44
6.5.2. Obtaining template designs.....	45
6.5.3. Publish and Subscribe.....	45
<b>7. Standards for Exchanging Template Definitions .....</b>	<b>46</b>
7.1. The Template Definition Exchange Standard.....	46
7.1.1. Objectives .....	46

7.1.2.	<i>Out-of-scope</i> .....	48
7.1.3.	<i>Reference Installation</i> .....	48
7.2.	Template Meta Data and Design Body.....	48
7.2.1.	<i>Data types used for this specification</i> .....	49
7.2.2.	<i>Overview</i> .....	49
7.3.	Template elements and their attributes .....	51
7.3.1.	<i>template/@id</i> .....	51
7.3.2.	<i>template/@name</i> .....	51
7.3.3.	<i>template/@effectiveDate</i> .....	51
7.3.4.	<i>template/@statusCode</i> .....	51
7.3.5.	<i>template/@displayName</i> .....	52
7.3.6.	<i>template/@versionLabel</i> .....	52
7.3.7.	<i>template/@expirationDate</i> .....	52
7.3.8.	<i>template/@officialReleaseDate</i> .....	52
7.3.9.	<i>template/@isClosed</i> .....	52
7.3.10.	<i>template/desc</i> .....	53
7.3.11.	<i>template/desc/@language</i> .....	53
7.3.12.	<i>template/classification</i> .....	53
7.3.13.	<i>template/classification/@type</i> .....	53
7.3.14.	<i>template/context</i> .....	55
7.3.15.	<i>template/item</i> .....	56
7.4.	Template design body definitions.....	57
7.4.1.	<i>element</i> .....	60
7.4.2.	<i>element/desc</i> .....	62
7.4.3.	<i>element/item</i> .....	63
7.4.4.	<i>element/example</i> .....	63
7.4.5.	<i>element/vocabulary</i> .....	63
7.4.6.	<i>element/property</i> .....	64
7.4.7.	<i>element/text</i> .....	66
7.4.8.	<i>Attribute</i> .....	66
7.4.9.	<i>attribute/desc</i> .....	67
7.4.10.	<i>attribute/item</i> .....	67
7.4.11.	<i>attribute/vocabulary</i> .....	67
7.4.12.	<i>choice</i> .....	68
7.4.13.	<i>include</i> .....	69
7.4.14.	<i>Schematron statements</i> .....	71
7.5.	How to Create, Validate and Consume Template Definitions.....	73
7.5.1.	<i>Creation</i> .....	73
7.5.2.	<i>Validation</i> .....	73
7.5.3.	<i>Consume</i> .....	73
7.6.	How to exchange Template Definitions between Systems .....	74
7.7.	Best practice examples for CDA definitions .....	74
7.7.1.	<i>Example of a CDA section level template</i> .....	75
7.7.2.	<i>Example of a CDA header level template</i> .....	77
7.7.3.	<i>Example of a CDA entry level template</i> .....	78
7.7.4.	<i>Example of a CDA document level template</i> .....	80
<b>Appendix A – Glossary</b> .....		<b>84</b>
<b>Appendix B – Acronyms and Abbreviations</b> .....		<b>86</b>
<b>Appendix C – Conventions and Best Practice for Template Documentation</b> .....		<b>87</b>
<b>Appendix D – References</b> .....		<b>94</b>



# 1. Introduction

## 1.1. Purpose

HL7 V3 provides a global framework for exchange of healthcare information as documents or messages or services by providing a framework for constructing instances of data according to agreed definitions in a standard fashion. The globally agreed definitions are often fairly general in nature due to the intention of their scope or the requirements to be globally suitable, and a framework for making additional rules for more specific knowledge models is required. Templates are used to provide this framework within the context of HL7 V3 and especially the Clinical Document Architecture.

Beside HL7 V3 artifacts there are other HL7 standards that can profit from using the same template framework, mainly v2.xml, the XML representation of v2 messages [v2xml], or FHIR [fhir]. In fact the template framework described here can be used for any XML based data representation.

This document arises from joint work of the HL7 Templates SIG and the Template Specifications project of the Modeling and Methodology TC. Templates have had a long genesis from the first identification of their need, and a series of draft documents have already been published, most of them incomplete in some fashion or other.

This document builds on the existing template specifications or drafts, e.g.

- *HL7 V3 Templates: HL7 Version 3 Standard: Specification and Use of Reusable Constraint Templates, Release 2* published in February 2008 (which is replaced by this specification),
- the recently balloted *HL7 Templates Registry Business Process Requirements Analysis Informative Ballot, Version 2*, published recently,
- various implementation guides published by HL7's Structured Document Working Group,
- various implementation guides published by IHE and by other governance groups outside HL7 International and
- other new work in the general interoperability methodology space to provide a complete specification for templates.

## 1.2. Scope

This implementation guide describes how templates are specified, registered, used and exchanged. It describes how to exchange template definitions using an Implementable Technology Specification (ITS). It covers template characteristics, versioning of templates and definitions of relationships that exist for templates. It also explains how to use and validate template references in XML instances.

At this point in time it primarily provides a framework within the context of HL7 V3 and especially the Clinical Document Architecture.


## 1.3. Ballot Status of the Document

This document is advanced for DSTU ballot. There is reference to several outstanding issues in this document, but it is believed that this document should be published as a



DSTU without waiting for these issues to be resolved. It is planned to resolve these issues before the templates specification becomes fully normative.

Outstanding issues are identified in the following box:

	This is an open issue or unanswered question.
---	---

## 1.4. Audience

The audiences for this implementation guide are architects and developers of healthcare information technology (HIT) systems anywhere in the world that exchange clinical data.

Business analysts and policy managers can also benefit from a basic understanding of the use of templates.

### 1.4.1. Requisite Knowledge

- HL7 V3 Interoperability Standards (Messaging)
- Clinical Document Architecture CDA Release 2
- SNOMED ([www. http://www.ihtsdo.org/snomed-ct](http://www.ihtsdo.org/snomed-ct))
- LOINC (<http://loinc.org>)
- UCUM (<http://unitsofmeasure.org>)
- OIDS (<http://www.hl7.org/oid>)

## 1.5. Organization of the Guide

This implementation guide is organized into the following chapters

- Chapter 1 is this **introduction**, defining scope, audience and prerequisites.
- Chapter 2 defines what **templates** are, how they function, their characteristics, and how they are defined.
- Chapter 3 covers **template versioning** to describe the life cycle of templates and the corresponding meta data used to identify and characterize versions of a template.
- Chapter 4 describes possible **relationships** between various templates and other artifacts like models.
- Chapter 5 provides **use cases** which demonstrate the creation, use, maintenance and governance of templates.
- Chapter 6 covers questions and answers around **management and governance** of templates.
- Chapter 7 defines the **Template Definition Exchange ITS**, an implementable technology specification for the exchange (and storage) format of template artifacts. Furthermore the **use of templates** to generate instances which conform to the template definition and tools and techniques for **validating template conformance** in generated instances are discussed.
- Appendix A is a **glossary** defining the most important terms used in the “Templates” domain.
- Appendix B includes a lookup reference for **acronyms and abbreviations**.
- Appendix C summarizes **conventions and best practice for template documentation** with examples from various governance groups.

- Appendix D records **additional references** which augment the topics covered in this document.

## 1.6. Conventions

This guide adheres to the following conventions:

- Examples are all non-normative
- If there are any differences between examples and the normative text, the examples are in error

### *Reference Implementation*

*The Template Definition and Exchange Format described in this guide is not only a specification but has already been implemented successfully and is used in real projects dealing with template creation, maintenance and use. Most of the described features and functionalities have undergone a proof-of-concept phase and were determined by typical practical use cases and refined repeatedly by experiences from practice.*

*Notes regarding this Reference Implementation appear in a green box.*

## 1.7. Contributions

In addition to the editors and authors mentioned above the following groups gave important input to this document.

- HL7 Working Groups
  - Templates (discussions running from October 2012 to October 2013)
  - Modeling and Methodology
  - Tooling
  - Structured Documents
- IHE (Integrating the Healthcare Enterprise)
- National Institute for Health IT (Nictiz), The Netherlands
- The ART-DECOR expert group
- HL7 Austria
- Austrian ELGA infrastructure project
- HL7 Germany (Interoperability Forum)
- IHE Germany
- The European epSOS Project (European Patients - Smart open Services)
- The European Semantic Health Net Initiative

## 2. Templates and Template Versions

### 2.1. What is a template?

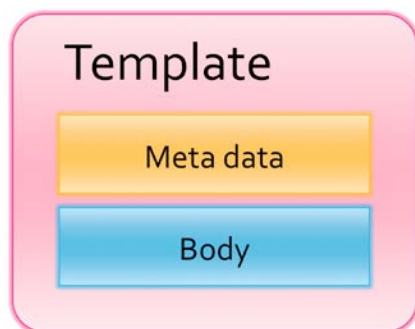
A template represents a formal definition of a set of constraints on a model, e.g. an HL7 R-MIM. The constraints are expressed as a formal definition, for example as a restriction on the attribute value domains, cardinality and optionality of the information model when it is applied to a particular use case or context,. The definition may be expressed in one or more human readable languages, or as a formal definition such as a model. For example, an HL7 V3 template is an expression of a set of constraints on a specific message R-RIM. Another example is a CDA header level template, which is an expression of a set of constraints on the CDA R-MIM header part.

Templates are used to further define and refine these existing models to specify a narrower and more focused scope. They function to apply additional constraints to an instance of data, or a portion of an instance of data. The definition represented in the template includes one or more implementation-specific representations, which can be used to validate instances in a particular context. As an example, the blood pressure template used by a intensive care specific template would be more focused than the blood pressure template used in other situations.

Templates often provide additional definition and documentary material that describe how the information models are applied to very specific use cases or contexts. This material needs to be consistent with the underlying model fragments to which it applies. It is typical for template to be defined as reusable modules, e.g. a template that contains other frequently used templates. As an example a vital signs template would typically include a blood pressure template.

A template is logically split into

- A meta data part such as an identifier, versioning information and a definition of the purpose etc. and
- A body part that contains the actual set of constraints.



*Figure 1: Template parts*

### 2.2. What is a template version?

Templates, like other specification artifacts, have a life-cycle. They come into existence with an initial design, and then over time, their design is modified and updated to match the continually evolving environment in which they are used.

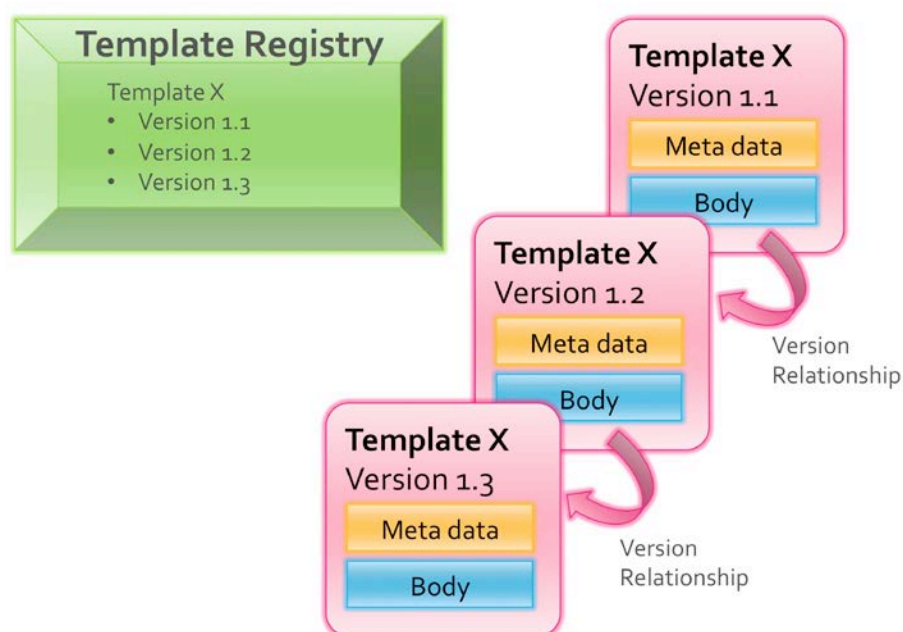
Each version of a template represents a distinct design for the associated definition. Meta data associated with the version describes the precise purpose and use of the particular definition for the template.

A governance group starts with the design of a template, and after use or review it may be necessary to develop and release a new, enhanced or corrected version of the initial template. The revision history of a template is managed through the creation of “versions”. Each version of a template represents a different design in its evolution.

Furthermore it must be possible to mark a version of a template as being in a certain state of maturity, so that its use in other designs or its use for content creators and content consumers can be controlled. Template versioning supports the management of specification artifacts at the design level. It is important that each version of a template contain meta data which associates it with the original intention of the template (which is bound to the template identifier) and meta data which uniquely identifies the version itself.

While multiple versions of a template can concurrently be useful, because they support instances of the template or they are used by systems that are not yet aware of an improved version of a template, the most recent version is thought to be the “current” version and should be used in creating new instances. If another use case or purpose is to be served a new version of a template is not used: a new template is created.

Each version of a set of versions of a template are handled as distinct and complete artifacts each with meta data and body. The fact that they belong to the set of versions is expressed by the (implicit or explicit) template version relationship. This is because also parts of the meta data may change over time. Please note that a template registry that has a list of templates may present the set of versions appropriately, e.g. summarizing all versions “under” the name of the template (see figure).



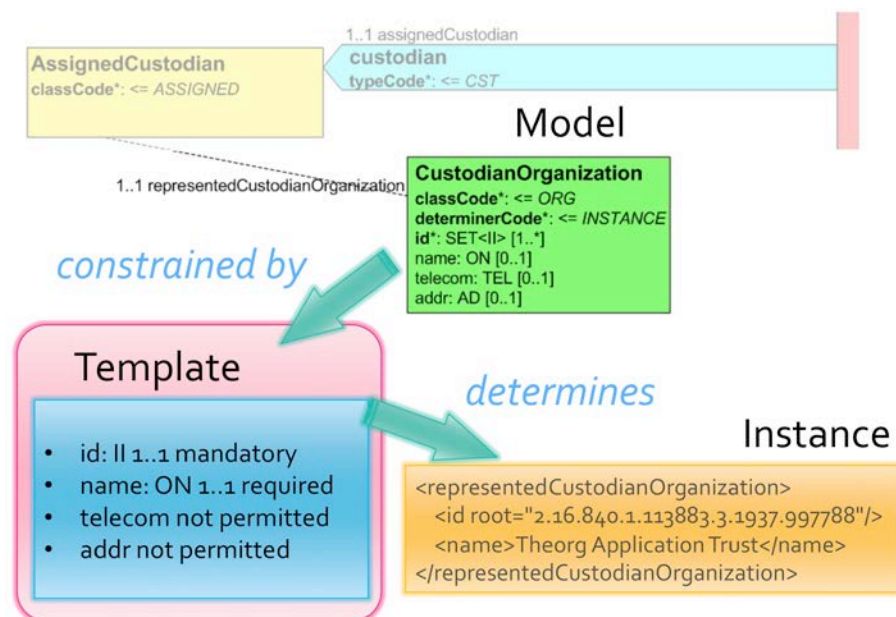
**Figure 2: Template versions are handled as complete artifacts including meta data and body. They have an implicit or explicit version relationship. Template Registries/Repositories may list all versions of a set of templates for example in a hierarchical list.**

## 2.3. Why are templates needed?

In order for data to be interoperable, it must conform to expressions of the base object model which is commonly understood. In HL7 V3, instances of data will firstly conform the HL7 Reference Information Model (RIM). The data may also conform to other models (R-MIMs) which are derived from the RIM and meet specific use case requirements. Templates are a documentation of the conformance rules associates with various models. The rules are applied when generating, validating (or processing) messages, documents and services.

Other models may exist at many levels of abstraction. This includes static models, which describe a set of constraints on the RIM that clarifies how some real world concept is properly described in the RIM. However these static models are generally rather broad and generic. Other more detailed models may conform to these static models refining the generic concepts to more specific ones such as particular laboratory tests.

Templates are used to constrain and define the structures of these more atomic concepts, and are able to be reused in multiple different contexts wherever the real world concept they describe occurs, such as a laboratory report in a CDA document or quoted within a pharmacy message as supporting evidence. With templates one is able to express additional set of conformance rules beyond the pure static model to ensure interoperable data is created and shared.



*Figure 3: Templates typically constrain underlying models, e.g. a model fragment defining a generic Custodian Organization is further restricted by a template that restricts the identifier to be exactly 1 mandatory id, exactly one required name and does not allow telecom or addr in the instance. The template definition determines how a conformant instance looks like. An instance may be validated against the template definition (and the underlying model).*

## 2.4. Why is template versioning needed?

Template versioning is needed to enable template designs to evolve over time.

Template versioning enables template designers to control and shape the conformances that make up their designs, tailoring the design to fit an intended purpose. Depending on the way a template is used, the design can include conformances that are specified in a particular version of the template's design specified at the time of definition, or at the time of use.

## 2.5. How does template versioning work?

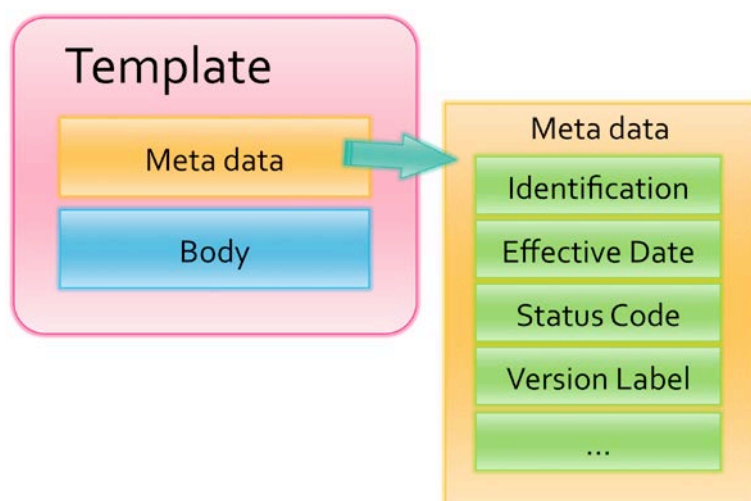
Each version of a template is associated with a particular template. The template has a mandatory globally unique, non-semantic, identifier. It serves as the identifier of the original intent of the template and as the identifier of the “family” of versions.

Templates also have a mandatory timestamp (date and optional time), called the “effective date”. The date can be seen as the point in time when the template “came into being”, i.e. was relevant at least to the governance group. Use of the template prior to this date would be considered an invalid use of the template.

It can be expected that a template designer always knows when a version of a template is intended to come into being, regardless of the status of the template, i.e. whether it is a draft or an active (officially published) template. The effective date functions in conjunction with the other version identification properties to uniquely identify a template at the version-level.

In addition to the identifier and the date a status code shall be maintained. It reflects the status of the design, e.g. that it is still in draft, or may be used or its use is no longer recommended (retired).

A template version may also have a version label which is actually a label for the effective date and is used to further distinguish one version of a template from another.



*Figure 4: Template meta data (partially)*

Some governance groups consider the combination of the template identity and “effective date” to reflect the “version” of a template. Other groups prefer the template identity and version label to uniquely identify the version of the template.



## 2.6. Where do templates help?

To facilitate interoperability, instances of data, either documents, messages or service payload fragments, need to conform to some reference model or some set of reference models, such as the CDA, a document specific constraint on CDA, or a message specification defined for a particular interaction. Further, portions of the data may conform to additional, even more specific models, as specified by one or more lower level templates.

Templates can be seen as an “instruction” for the creation of proper and conformant instances. Despite the fact that an instance must always carry all semantics so it must be understandable even without any template ids in the instance itself, template ids in documents or messages can aid data validation and processing, for example for more efficient parsing (avoiding deep look-aheads) and they can also allow for quick routing and “easy” validation. When some definitional constructs establish that an instance of data conforms to a template, the instance of data is said to “conform to” a template.

## 2.7. How does a template design get defined?

A template design can be defined in a number of ways. They can be created as human-written specifications or they can be developed using modeling / design tools which automate the process for greater precision and consistency. Templates can be published in human-readable or in machine processable forms. They can be managed in documents where they are defined or in repositories deployed through technology tools developed specifically for the management of templates.

Each template represents a set of constraints on a model. A template could constrain a base model, such as CDA R2. Templates designed for one purpose may be usable for another purpose. For example, a section level template defined for the HL7 Continuity of Care Document may also be usable for a Discharge Document or an IHE Patient Care Coordination Patient Summary document. A template can be used or reused in for any purpose which it is fit to serve

To explain how templates get defined, it is helpful to understand the roles played by various “actors” involved in the processes associated with defining templates. These actors may represent an individual, an organization, or a system contributing a particular function.

### 2.7.1. Template Designer

A template designer is a person who develops the set of conformance statements which make up a template. The template designer considers the requirements for the template based on its intended purpose. A template designer may make use of previously defined templates in the construction of a new template. A template designer also may create a new template by further constraining a previously defined template.

Choices about the reuse of existing templates or creation of new templates is a part of the what a template designer considers when developing template designs. The underlying design of the templates plays an important role in creating interoperable and meaningful data. The development of high quality template designs requires a great deal of specialized knowledge about the subject matter where the template will be employed

(domain expertise), the applicable interoperability standard, and the universe of existing template designs.

### **2.7.2. Governance Group**

Template designers often create their designs within the context of a community. Often these communities are involved with standards creation, like HL7 and IHE, but they don't have to be. A template governance group establishes the rules and best practices associated with their creation and management of templates. The group establishes practices that are used to manage the intellectual property rights associated with the templates they create and use, and provide for governance functions. Some governance groups develop templates with the intent to register and share template designs. Others intend to maintain their template definitions as private works.

### **2.7.3. Template Repository**

A template repository houses the definition of template designs.

### **2.7.4. Template Registry**

A template registry tracks the existence of templates and their designs from one or more template repositories. It typically represents the point of view of a single governance group and manages the status of the templates and designs based on the practices and policies of that governance group.

## **2.8. Characteristics of a template and its versions**

Each template has a mandatory globally unique, non-semantic, identifier. It serves as the identifier of the original intent of the template.

Templates typically have a life-cycle like other specification artifacts. A governance group starts with the design of a template, and after use or review it may be necessary to develop and release a new, enhanced or corrected version of the initial template. The revision history of a template is managed through the creation of "versions".

Each version of a template represents a different design in its evolution. Templates can be managed at the template level, and at the design version level. Meta data exists for each template, as well as each design version of each template. Meta data at the template level permits the templates as a whole to be managed and tracked within template registries and template repositories. For more information about the functions performed by a template registry and a template repository, see section 7.6.

When does a new design represent a new template as opposed to a new version of an existing template? If the intent/purpose of the template changes substantially or the design of template is substantively different, a new template needs to be created. Typical situations where a new template is required include:

- Different purpose, where the difference can be either a narrower or broader purpose from an existing template's definition, or a substantively different purpose
- Value Set Bindings,
- Cardinality / Conformance,
- Containment of other templates



- Backward compatibility invalidation (see also section 3.4.7 about “backward compatibility”).

Templates have a variety of relationships to other templates. A template may be defined to inherit all the conformances of a previously defined template. A template may incorporate another template into its design by including that prior design as a component. A template could have an identical structure, but be used for a different purpose. These all represent different ways one template may be related to another. Backward compatibility is a relationship between different versions of a particular template. It is a complex notion which considers aspects of the design as well as issues of template governance. For template relationships see section 3.

## 2.9. Template Meta data

The following meta data exists for each design or version, as noted, of a template.

### 2.9.1. Identifier

Each template design is identified using a unique identifier. The identifier is a mandatory globally unique, non-semantic, identifier and is associated with the unique intent or purpose and acts as the primary identifier of the template definition. It serves as the identifier of the original intent of the template and as the identifier of the “family” of design versions which may exist to express the template over time.

### 2.9.2. Name

A template has a required name for the template as a secondary identifier. Please note that there is no guarantee that the name is globally unique but it shall be unique within a governance group. The name identifies the template. If a template has multiple designs, they are all called by the name of the associated template.

### 2.9.3. Effective Date

The template has a mandatory timestamp (date and optional time) after which the template existed regardless of its state (e.g. still in design phase or ready for use). The date can be seen as the point in time “when the template came into being” or when it became “in effect” for the governance group or a broader audience. Use of the template prior to this date would be considered an invalid use of the template.

The rationale behind a mandatory effective date as a version identification property is that it can be expected that a template designer always knows when “his” template first came into being, i.e. when he starts the work on template meta data or design body. It is important to distinguish between the effective date (in effect for the governance group) and an “official release date” (see later).

### 2.9.4. Version Label

A version of a template may contain an optional human readable version label for the template to be able to determine the correct version of a template.

Some governance groups may make this label to be mandatory, IHE for example requires the use as a version label as “version identifier” which is used in conjunction with the template identifier to uniquely identify a version of a template.

During the development of template designs requirements may lead to create a new version of the initially published active template. The criteria that it is a version of the former active template, the intent/purpose of the new design must be the same as the intent/purpose of the original template.

Further, if an application using the template either as a content creator or consumer does not need to make any substantial changes to their code to process data conforming to the new design, then the new version is said to constitute a **minor revision**.

If the application using the template either as a content creator or consumer will need to make substantial changes to their code for processing data conformant to the new design, then the new version is said to constitute a **major revision**. Some governance groups may choose to indicate major or minor revisions in the naming scheme used for the version label.

*Table 1: Recommended properties of a “version” of a template*

Version of a template
<b>Keeps the same identifier as the former template</b> <b>Shall have a new effective start date</b> <b>Shall have a status code</b> <b>May have a new version label</b>

In some cases, error correction may lead to the need to create a new version of a template design. The governance group may decide whether a new version is required, as either a major or minor revision, to replace the former template design, or if the former template design can be modified to correct the errata.

#### 2.9.5. Expiration Date

A template may have an optional date at which the concept represented by this template becomes stale, and should be reviewed for (clinical) relevance and accuracy. The expiration date is set to indicate that it should/shall no longer be used for new designs (by anyone), i.e. another template design that uses this template by inclusion or containment after this date would be considered ill-advised.

#### 2.9.6. Official Release Date

A template may be released (published) by the governance group in any status it might have. However it may be useful to set an “official” date since when the template is ready for use (trial implementation, production etc.). This is done by populating the optional “official release date”.

#### 2.9.7. Status

Every version of a template has a status. The status indicates the level of maturity of the design and may be used to manage the use of the design. While different template governance organizations may establish additional status states to support local template management practices, or chose to reduce the number of statuses it uses, the follow set of statuses have been defined:

*Table 2: Status states for a template version*

Status code	Description
<b>draft</b>	Design is under development (nascent).
<b>active</b>	Design has been deemed fit for use and published by the governance group.
<b>review</b>	Design is completed and under review.
<b>retired</b>	A previously active design which is targeted for discontinuation and should no longer be used, but for historical purposes. A newer design may or may not exist. The template is still active.
<b>cancelled</b>	Design is determined to be erroneous or not fit for intended purpose and is discontinued before ever being published in an active state.
<b>rejected</b>	A previously active design which has been determined to be erroneous or not fit for the intended purpose and should no longer be used, even for historical purposes. A newer design likely exists to have replaced this design. The associated template is still active.
<b>terminated</b>	A previously active design which has been determined to be erroneous or not fit for the intended purpose and should no longer be used, even for historical purposes. A newer design does not exist to replace this design. The associated template is retired.

It must be noted that from a governance perspective it may be appropriate to use the full range of status codes. From an implementers perspective only the status codes “draft”, “active”, “review” and “retired” may be of interest.

#### 2.9.7.1. Draft

Each template design begins as a draft design. It is assigned its own meta data and a version of the associated template. Especially an effective date (start date) is assigned to the design version when it is created.

A governance group may decide to have also different versions of a draft template, i.e. different effective dates for a draft template in order to version the template.

As long as a template version has the status of “draft” the design may change, even substantially. Once the status is advanced to “review” or the template becomes “active”, etc. no further changes can be made to this version of the template’s design. Additional changes necessitate creation of a new version.

#### 2.9.7.2. Active

This is the state of an approved template. The governance group has decided that this template is fit for purpose. When propagating a template to “active”, the effective date is left as is (e.g., like the draft version).

#### 2.9.7.3. Review

This status indicates that the template is undergoing a review process, either within a governance group for an internal review or published for an outside review.

This is typically a transition from the status “draft” and would include a (public) comment phase.

After the review phase the template may become endorsed (=active), or rejected/inactive/retired with the possibility of crafting a new version for this template in status draft.

#### **2.9.7.4. Retired**

Once a template design should not be used any longer or is no longer recommended, the expiration date is set and the status is set to retired/deprecated. This design version should no longer be used for new instances, or new definitions (template designs with containment or inclusion).

A retired design should not be used for validation unless it is used for a historical validation (i.e. the date of the creation of the instance is falling between effective and expiration date).

#### **2.9.7.5. Cancelled**

If the template was never used and published, a template may be cancelled, e.g. if the template is found to be a bad idea to create or a duplicate effort. Other such situations include if one governance group decides to abandon their template (and its id) and, for example, join another governance group or if one governance group decides to use a template from another governance group.

#### **2.9.7.6. Rejected**

During some review, a template design maybe found inadequate or erroneous by the governance group, and therefore it becomes rejected, may be even before publication. In this case the expiration date (end date) is set (to the current date) indicating that the use of this template is no longer recommended. This is the end of the life cycle of this template design. However, a new design for this template may be created.

This status is typically used within a governance group.

#### **2.9.7.7. Terminated**

If an already published template design was found a bad idea to create, duplicate effort, etc., the status can be moved to “terminated”. This is the end of the life cycle of this template concept as a whole and no new design version will be published for the associated template. If a document contains a terminated template design, the instance is invalid from a machine processability perspective (although it still may contain some reasonable or useful clinical information).

If a document contains a terminated template design, the instance is invalid from a machine processability perspective (although it still may contain some reasonable or useful clinical information).

#### **2.9.8. Additional meta data**

In addition to the version-related meta data the following items are part of a template definition.

### 2.9.8.1. Description

A description that defines the purpose / intend and the scope of the template.

### 2.9.8.2. Classification

An optional classification of the template informing about

- The type of the template, e.g. CDA Section Level Template, Document Level Template etc. (see also section 2.12).
- Possible search tags to easily find templates in large registries, e.g. tagging the template with “blood pressure” or “diabetes regime” would enable a search engine to find the template by this term.
- The format of the template; as of now this is “HL7 Version 3”, but future developments may make other options possible.

### 2.9.8.3. Relationships

This covers an optional list of relationships to other templates or model artifacts (see chapter 3).

### 2.9.8.4. Context

An optional context of a template , i.e. where in an XML instance the template rules are considered to apply to. From a practical viewpoint templates may have no context. In this case the template is not “exposed” for external use, but rather used for internal inclusion in other templates (see also section 2.10.2).

If the template is intended to be published for external purposes (for containment in other template designs) and for example be triggered by a template id in an instance, it shall have a context. Typically there are three types of context specifications.

#### **Sibling node context**

The template rules apply to all sibling elements and descendent nodes in the instance. This means that the context for validation is: all sibling nodes of any templateId element in an instance where the @root is populated with the same id as the template id.

#### **Parent node context**

The template rules apply to the siblings of the parent node and all descendent nodes in the instance.

#### **Pathname specified context**

A pathname (making use of XPath expressions) is specified, which allows to activate templates in an instance without the need to have template id elements in an instance.

### 2.9.8.5. Open vs. closed

An indication whether a template is considered to be closed or open. For further discussion on open vs. closed templates see section 2.11.

### 2.9.8.6. Examples

It is best practice that a template contains one or more examples or example fragments to illustrate valid (or expose invalid) examples.

#### 2.9.8.7. Publishing Authority

The authoritative body who has reviewed the template for (clinical) accuracy and relevance, and authorized it for publication. Template registries may require templates to populate this element. It typically contains identifications, names and contact information.

#### 2.9.8.8. Endorsing Authority

A list of bodies who have reviewed the template for (clinical) accuracy and relevance, and endorsed it for use. It typically contains identifications, names and contact information.

#### 2.9.8.9. Revision History

The free text description describing the changes in this version of the template as compared to the previous version; since template versions are built off of previous versions, the net effect of this field is to function as a comprehensive historical reference of the template, so this field should be populated if the template is not the first version.

### 2.10. Template Body

The actual template design, also seen as the “body” of the template, is a collection of constraints that ideally describes the structure and semantics of all instance elements. It is typically presented as a template item table, see section 2.14 for an example.

#### 2.10.1. Types of constraints

Constraints express a template’s design in several different ways:

- Data Type constraints
- The cardinality
- Mandatory elements, e.g. if data may be absent (nullFlavor)
- The conformance, i.e. whether a system must support an element or not
- Vocabulary bindings and coding strengths
- Possible fixed values
- The data type or a data type flavor of an element where appropriate
- Requirements for additional attributes or properties such as units (measurements), ranges, fraction digits
- Containment of additional components which may or may not be templated

In general, the constraints in the design of the template define the properties of a conformant XML element or an XML attribute. In addition, co-occurrences of items or conditional requirements may also be defined.

##### 2.10.1.1. Data Types constraints

The data type of an element is simply determined by specifying a data type name or a data type flavor name.

*Table 11: Examples of Version 3 data type assignments / constraints*

Item	Description	Data Type (Flavor)
<b>id</b>	An instance identifier	II
<b>code</b>	A coded element	CE
<b>effectiveTime</b>	A time stamp	TS
<b>validityPeriod</b>	A time interval	IVL_TS
<b>value</b>	A measurement of a physical quantity	PQ
<b>repeatNumber</b>	Repetitions, non-negative numbers	INT.NONNEG
<b>date</b>	A time stamp precise to the day	TS.DATE

Note that the constraint shall be equal to or a valid demotion of the corresponding data type of the underlying model (if present).

#### 2.10.1.2. Cardinalities

The cardinality indicator (0..1, 1..1, 1..\*, etc.) specifies the allowable occurrences within a document instance. The cardinality indicators are interpreted with the following format “m..n” where m represents the least and n the most:

- 0..1 zero or one
- 1..1 exactly one
- 1..\* at least one
- 0..\* zero or more
- 1..n at least one and not more than n

This guide uses cardinalities for element and attribute definitions where attribute definitions can be either 0..0 (prohibited, not allowed), 0..1 (optional), and 1..1 (required).

#### 2.10.1.3. Mandatory/Conformance

Elements that are defined to be mandatory must be populated with valid data. If an element is not flagged as mandatory, data may be absent, e.g. by using a nullFlavor or the element may be simply omitted if optional.

The conformance expresses whether a system must support an element or not.

The following terms are used in this guide to express conformance criteria.

##### Mandatory (M)

The attribute is mandatory, i.e. a valid value shall be provided and no null value is allowed. The minimum cardinality is at least 1. This also implies that if the sender has no valid value for such an attribute, the message cannot be sent.

It is indicated as “M” in the conformance column of the template item table, a shorthand for “mandatory” with required conformance.

A sender must support elements with “mandatory” conformance, a receiver must understand these element.



## Required (R)

The item is required, i.e. a valid value should be provided or if missing a null value is allowed if its minimum cardinality is 1, or may be omitted if its minimum cardinality is zero.

In messages, the element must be communicated if its minimum cardinality is one. In the case where the element is not mandatory, it may be communicated with a null value. Note that any element declared to be "Mandatory" must also be "Required" and have a minimum cardinality of one. If the minimum cardinality is zero, and the element is "Required", conforming applications need not send the element if data does not exist. In some governance groups this is referred to as "Required but may be empty".

For required elements, conforming applications must demonstrate their ability to provide and communicate not null values. Receiving applications must demonstrate their ability to receive and process (eg. store, display to users) not null values for required elements.

It is indicated as "R" in the conformance column of the template item table.

A sender must support elements with "required" conformance, a receiver must understand these elements.

## Optional (O)

The item is truly optional, i.e. a valid value may be provided or if missing may be omitted.

It is indicated as "O" in the template item table, a shorthand for an unspecified conformance with a minimum cardinality of zero.

## Not permitted (NP)

The item is not allowed, i.e. the number of allowable occurrences is 0. It is indicated as "NP" in the template item table.

## Conditional (C)

This item has an associated condition predicate and may depend on the co-occurrence of other elements or properties of the instance or situations. The condition must be testable in the context of an instance (see also section 2.10.4).

The conditional conformance criteria are mentioned in a corresponding table like this:

*Table 3: Example of a Condition Predicate Table*

Card	Conf	Predicate
1..1	M	If number of gravidities greater than 0
0..1	O	otherwise

In this example the following situation is covered: if the predicate is satisfied then

- a conformant sending application must always send the element
- a conformant receiving application must process or ignore data in the element; it may raise an error if the element is not present.



If the predicate is not satisfied then

- a conformant sending application should not (if conformance is optional) or must not (if conformance is NP = not present) send the element.
- a conformant receiving application must not raise an error if the condition predicate is false and the element is not present, though it may raise an error if the element is present.

A conditional attribute is indicated as “C” in the template item table, followed immediately by the condition predicate table.

#### Fixed (F)

This is used for attributes only and indicates that an attribute has a fixed value. Its fixed value shall appear in an XML instance.

It is indicated as “F” in the template item table, a shorthand for a mandatory element with required conformance with a fixed value. The cardinality should be 1..1.

#### 2.10.1.4. Conformance Verbs

The keywords SHALL, SHOULD, MAY, NEED NOT, SHOULD NOT, and SHALL NOT in this document are to be interpreted as described in the HL7 Version 3 Publishing Facilitator's Guide (<http://www.hl7.org/v3ballot/html/help/pfg/pfg.htm>):

- SHALL: an absolute requirement
- SHALL NOT: an absolute prohibition against inclusion
- SHOULD/SHOULD NOT: best practice or recommendation. There may be valid reasons to ignore an item, but the full implications must be understood and carefully weighed before choosing a different course
- MAY/NEED NOT: truly optional; can be included or omitted as the author decides with no implications

The keyword "SHALL" allows the use of nullFlavor unless the requirement is on an attribute or the use of nullFlavor is explicitly precluded.

The objective of the Templates Exchange Format is to allow machine processable template representation. The representation with conformance verbs and natural language like used in CDA implementation guides is not subject of this specification. It has been observed that there is a variation on natural language representation of conformance statements. Please refer to Appendix C to see the recommendations and best practice for template documentation.

There is a relationship between conformance verbs and the cardinality/conformance indicators used in this guide. A summarization with these relationships can be found in the next table.

*Table 4: Relationship between Conformance verbs and Cardinality/Conformance properties used in this guide*

Shown as	Mandatory?	Conformance	Possible Cardinality	nullFlavor allowed?	Description
<b>M</b>	Yes	R	1..1 1..*	No	Element SHALL be present with a proper value
<b>R</b>	No	R	1..1 1..*	Yes	Element SHALL be present in instance and SHOULD be populated with a proper value; if no proper value is present, a nullFlavor attribute SHALL be present
<b>R</b>	No	R	0..1 0..*	No	Element SHOULD be present in instance and SHALL be populated with a proper value; if no proper value is present, the element SHALL NOT be present
<b>O</b>	No	O	0..1 0..*	Yes	Element is MAY BE present (is truly optional)
<b>C</b>		C			Element is conditional and may depend on the co-occurrence of other elements or properties of the instance or situations. The conditional conformance criteria are mentioned in a corresponding table.
<b>NP</b>	No		0..0	No	Element SHALL NOT be present

#### 2.10.1.5. Fixed Values

This allows expressing the intended fixed (prescribed) value for conforming instances.

The most frequent use of fixed values is with a structural attribute. From a model perspective they are fixed and default to a single value (code). The cardinality of such structural attributes is handled to be either optional (as is mostly the case in CDA) or required (as in V3 messages).

Table 5: Example of fixed values for structural attributes

Examples
<b>observation classCode="OBS" moodCode="EVN"</b>

Fixed values for attributes are denoted as “F” in the template item table.

#### 2.10.1.6. Quantity Ranges

For quantities like real and integer it may be useful to specify allowed ranges in a template. This is typically done by defining a lower (minInclude) and a higher (maxInclude) boundary.

#### 2.10.1.7. Units

For physical quantities (HL7 data type PQ) it is useful to specify allowed units. For HL7 v3 PQ items this has to be a UCUM expression.

Please note that quantity ranges are related to their units and therefore need to be grouped appropriately if there is more than one allowed unit.

#### 2.10.1.8. Fractional Digits

In some cases it makes sense to constrain the number of fraction digits for physical quantities / real number. This may include a required number of fraction digits or an indication of the maximum number of fraction digits.

#### 2.10.1.9. Vocabulary Binding

For coded elements a template design may restrict the use of terms from a specific code system. These vocabularies are defined in various supporting specifications and may be maintained by other bodies, as is the case for the LOINC® and SNOMED CT® vocabularies.

There are three types of vocabulary bindings.

- A coded element can be bound to a **specific code** from a specific code system and the population of the element can be considered as a “constant”.
- A coded element can be bound to a specific **value set** by value set id or name.
- A coded element can be bound to a certain **concept domain** by domain id or name.

Note that value set identifiers, e.g. value set with OID 2.16.840.1.113883.1.11.78 / name *HL7ObservationInterpretation* do not appear in messages/documents. They tie the appropriate value set to the coded item, for example to perform proper validation.

Concept domain indications are normally not testable in an instance as they indicate the binding to a possible abstract conceptual domain. In a template they are typically replaced by value set references by specializing the original template for certain use cases.

Value set bindings adhere to HL7 Vocabulary Working Group best practices, see section about DYNAMIC vs. STATIC artifact binding, see later.

### 2.10.2. Inclusion and Containment

One of the design principles of templates is the re-usability: a template, once defined, may be used again in any context wherever appropriate. From practice, two kinds of “re-use” mechanism are known.

An **inclusion** within a template design makes use of another template by “virtually” copying the included template definitions, also known as transclusion. In essence this means that template definitions are included by reference and shown as-is on demand, i.e. at time of displaying the template or using it for the creation of validation scripts. Inclusion is automatic and transparent to the user.

Example: a CDA document level template includes the definitions of a CDA *typeId*, *effectiveTime*, *title* and *setId+versionNumber*.

A **containment** within a template design is a reference to another template without actually showing the contained definitions. A typical situation is a CDA section that may contain entries. The reference only is part of the section definition. The definition of the entry itself remains part of the entry level template.

The containment relationship constraints between a specific structure (context) in an XML instance and sub-structures in that context (child elements).

### 2.10.3. Choice

In some cases a designer wants to offer a choice of template elements and restrict the choice to be one out of n (cardinality constraint). A typical example is the CDA header author definition where in the underlying model a choice is defined between an *assignedPerson* and an *assignedAuthoringDevice* playing the role of the author and where 0..1 playing entities may be chosen.

### 2.10.4. Co-occurrence

Co-occurrence means the presence of some data depending on the presence or value of some other data, also referred to as conditional data.

A condition has a predicate that falls “true” or “false” and an associated cardinality and conformance statement. A typical example is the co-occurrence definition of an observation “pregnancy”: if the “gender of the patient is female” (predicate) then the pregnancy observation has the cardinality / conformance 1..1 R, otherwise (default predicate) it is not present (NP). The following table illustrates this example.

*Table 6: Example of the Condition Predicate Table for Observation “pregnancy”*

Card	Conf	Predicate
1..1	R	If patient is female
	NP	otherwise

Co-occurrences are often expressed in natural language, in other cases a formal expression is possible that can be tested against the instance by means of validation mechanisms, such a schematron.

Typically there are co-occurrences within an instance where a condition applies to one single instance and rely of properties on that instance. As an example, a documentation about female patient shall contain information whether she is pregnant or not.

A co-occurrence may relate to external data, i.e. where the presence of data in an instance is influenced by external factors (outside the very same instance). As an example a required lab result that is expected based on the history of the patient that has to be evaluated on a regular base.

In this specification, only co-occurrences constraints within an instance are handled.

#### 2.10.5. STATIC vs DYNAMIC artifact binding

For the binding of artifacts within a template definition, especially

- Vocabularies (value sets)
- Templates as Inclusions or Containments

the same mechanics are applied: an artifact is bound to an element either as

- STATIC, meaning that they are bound to a specified version (date) of the artifact,
- or DYNAMIC, meaning that they are bound to the most current version of the artifact.

Value set bindings adhere to HL7 Vocabulary Working Group best practices.

A STATIC binding is a fixed binding at design time whereas a DYNAMIC binding implies a need for a look-up of the (most recent) artifact at runtime.

It can also be found in practice to “freeze” any binding defined as “dynamic” to the most recent artifact at the time of the official publication and set the binding in the official publication itself to “static” for the most recent version.

#### 2.11. Open versus Closed Templates

A **closed template** defines what's relevant for that template and leaves no room for any other contents in the instance. An instance is considered non-conformant if it contains anything beyond the template definition.

An **open template** defines what's relevant for that template while allowing undefined contents in the instance. From the perspective of the template this undefined content is ignored. This may have benefits if it is to be expected that implementations add additional content, or when you want to cater backward compatibility in a next version of the template.

Open templates allow designers to develop additional structured content not constrained within for example an implementation guide. However, implementers are encouraged to bring their use cases forward to the governance group as candidate requirements to be formalized in a subsequent version of the standard to maximize the use of shared semantics.

Another typical situation is that templates in a repository for re-use are defined as open as when they are used within a document definition (document level template) a governance group may decide to use all templates as closed, i.e. no other content then specified is allowed. The same may temporarily apply during conformance testing for example a connect-a-thon where it may be required to detect undefined content.

## 2.12. Types of Templates

According to their “position” or function within a message or a document the following types of templates are distinguished.

### 2.12.1. CDA Document Level

These templates constrain fields in the Clinical Document (CDA) header, and define containment relationships to CDA sections.

For example, a History-and-Physical document-level template might require that the patient’s name be present, and that the document contain a Physical Exam section.

### 2.12.2. CDA Header Level

These templates constrain fields for parts of the CDA header, like the patient, the author or the service event.

### 2.12.3. CDA Section Level

These templates constrain fields in the CDA section, and define containment relationships to CDA entries.

For example, a Physical-exam section-level template might require that the section/code be fixed to a particular LOINC code, and that the section contain a Systolic Blood Pressure observation.

### 2.12.4. CDA Entry Level

These templates constrain the CDA clinical statement model in accordance with real world observations and acts.

For example, a Systolic-blood-pressure entry-level template defines how the CDA Observation class is constrained (how to populate observation/code, how to populate observation/value, etc.) to represent the notion of a systolic blood pressure.

### 2.12.5. Message Level

These templates constrain the HL7 V3 message model and may define containment relationships to wrapper and payload components. This is the counterpart to CDA document level templates.

### 2.12.6. Clinical Statement Level

These templates constrain the HL7 V3 clinical statement model in accordance with real world observations and acts, used for example in Patient Care Messages. This is similar to CDA entry level templates. Please note that Clinical Statement Level templates used in some V3 messaging artifacts may be equivalent to CDA entry level templates but not necessarily identical.

### 2.12.7. Data Type Level (= Data Type Flavors)

A data type may also be constrained by means of a template. In effect the mechanism to handle data type constraints since Data Types Release 2 (ISO 21090) is to create and use a data type flavor, for example instead of “INT” for an integer use “INT.POS” if a positive integer is required.

### 2.12.8. FHIR Resource Profiles

The corresponding artifact to a template in the FHIR world is a profile. Expressing FHIR profiles as “templates” as defined here has been explored by both HL7 core groups but is not within the scope of this document yet.

### 2.12.9. v2.xml Profiles

The XML representation of HL7 v2 messages called v2.xml can be constrained by the definitions made in this “template” specification. The actual ITS (see below) allows constraining any XML. However, the practical implications have not yet been examined and the coverage of v2.xml here is considered to be out of scope yet.

## 2.13. Template Design Considerations

An important consideration about a template design is how deep the template structure should go. A model is a collection of classes and the associations between them. When a model is expressed as a set of conformances in a template, the structure could be infinitely deep. To date, most templates are rather shallow, but some may include 3 or 4 levels of nested conformances. If one of the levels includes a previously defined template, then the nesting continues deeper based on the levels defined for that template.

This design feature creates a modularly recursive or “fractal-like” characteristic to a template. Being able to break a large model into a variety of smaller reusable sets of constraints has proven to be useful. However, it does add to the complexity of the template designs.

### 2.14. Template Item Table Example

The following table is an example for a template item table. It is the start of the “Age Observation” Template from C-CDA [ccdar1].

*Table 7: Example of a Template Item Table*

Item	DT	Card	Conf	Description/Value
<b>observation</b>				
@classCode		1..1	F	OBS
@moodCode		1..1	F	EVN
templateId	II	1..1	M	Template Id of this template
@root		1..1	F	2.16.840.1.113883.10.20.22.4.31
code	CE	1..1	F	
@code		1..1	F	445518008
@codeSystem		1..1	F	2.16.840.1.113883.6.96 (Snomed-CT)
value	PQ	1..1	R	
@value		1..1	R	The actual age
@unit		1..1	R	Code from value set : AgePQ_UCUM

Please note that from the table it is immediately obvious that all structures have fixed values, except the *value* itself. In other words: a corresponding instance can be created with most of the “HL7” structures already determined and the focus is on the value only, which carries the actual “age”. Setting the focus on elements and attributes that actual carries a variable value can make implementation easier.



## 3. Template Relationships

### 3.1. What are template relationships?

The purpose of one template may depend on, build upon, or further constrain the purpose of another. Also, design interdependencies can define one template by referencing conformance expressed in another template or by including a component which has been defined separately in different template. One template may have had several designs over time as the exchange requirements for the information evolved. A subsequent design of a template can include conformances that affect the ways in which a new design is similar or different from its prior version.

### 3.2. Why are template relationships useful?

Template relationships help us understand how information specifications are similar or different from each other and enable us to logically deduce how data conforming to a template may or may not interoperable. The relationships between templates allow us to reason about the data that is exchanged using the templates.

### 3.3. How do template relationships work?

Template relationships enable processing logic to be applied to the data that is represented. If Template B is a further constraint on Template A, then by deductive reasoning, an instance of data that conforms to Template B can be processed according to rules established for processing data that conforms to Template A or Template B.

Also “versioning” is an aspect of relationships between templates, e.g. template K is the predecessor of template L.

### 3.4. Types of relationships between templates

Based on the design techniques used to define templates, several types of relationships exist between different templates and between design versions of a single template.

#### 3.4.1. Replacement (replaces)

A replacement is used when a new version of a template replaces older ones. With a replacement, the status of the older one typically gets retired (or inactive). This is inherent to versioning and could also indicate a new design replacing an old design (with a new id).

#### 3.4.2. Version

A “version” relationship may be automatically derived (in a proper registry) or explicitly set to express the relationship to other (older or newer) versions of a template design. Versions typically have the same id but a different effective dates.

Multiple versions of a template can be “active” or “draft” or “review” at the same time.

#### 3.4.3. Specializations / Generalizations (specializes, generalizes)

A specialized template is a narrower, more explicit, more constrained template based on a “parent” template.

Examples: A “Body Weight Observation” is a specialization of a “Simple Observation”

#### **3.4.4. Design Copy (copies)**

Used to express that a copy of a template design from another governance group has been made without change, except the id.

This implies always the creation of a new: the copied template gets a new id, the effective date is copied to the new effective date.

Foreign governance groups typically do this for later adaptation.

#### **3.4.5. Adaptation (based on)**

Copying a template design from another governance group (i.e. design copy) and change it, i.e. change both id and design is called adaptation. The adapted template is “based on” the original template which means it can be an extension or a specialization (restriction) of the original template design.

This implies always the creation of a new: the adapted template gets a new id and a new effective date.

#### **3.4.6. Equivalency (equals)**

“Equivalency” indicates that two templates have the same purpose and the same design, however their governance and/or meta data and/or details of their design may be different. They are meant to be semantically equivalent because a transformation exist without addition or loss of meaning.

Examples: use of different vocabulary (LOINC vs SNOMED), slightly different XML representation of CDA entries vs. Clinical Statements in Patient Care Messages.

#### **3.4.7. Backwards compatibility**

Indicates that a template is considered to be backward compatible to a previous version of the template. A template is backwards compatible if it validates all instances of instances produced by former versions of the template.

Determining true backward compatibility can be complex. A new version of a template defined as “open” may reject instances constructed using the older version as “open” means it allows other elements in an instance as defined. In other words, older instances may contain legal additional input that fails to validate against a newer version of the template.

True backward compatibility can be determined more reliable with templates defined as “closed” only.

#### **3.4.8. Forward compatibility**

Indicates that a template is considered to be forward compatible to a previous version of the template. Forwards compatibility means that a newer template can be rolled out in a way that does not break existing implementations.

#### **3.4.9. Containment**

This relationship indicates that a template has a reference to a contained template. In a proper registry/repository this relationship may be determined automatically (implicit relationship), see also section 2.10.2.

Examples: a CDA section level template has an entry that contains an “Age Observation” template.

#### **3.4.10. Inclusion**

This relationship indicates that a template makes use of virtually copying another template design into the template. In a proper registry/repository this relationship may be determined automatically (implicit relationship), see also section 2.10.2.

Examples: a CDA document level template includes a “Record Target” definition that is used in multiple templates.

### **3.5. Relationships to other artifacts**

#### **3.5.1. Relationship to Underlying model(s)**

This relationship is used to express the link to an underlying model. Examples are conceptual models, Detailed Clinical Models, HL7 R-MIMs, HL7 CMETs. In other words, the template is based on / derived from a (more generic) model.

#### **3.5.2. Subscribe with “Adopter”**

An “adopter” relationship expresses a kind of relationship to other individuals or governance groups in the sense of a subscription (a “template follower”) on any changes or improvements of a template or to proactively offer participation in testing and reviewing new versions. Governance of the template remains untouched.

It is a template registry matter to record those interests and to signal the “adopters” on changes or the start of review phases.

## 4. Using Templates to Create, Validate and Consume Instances

### 4.1. Templates as Definition of Clinical Concepts

HL7 Models serve as a structured formalism through which human beings can unambiguously exchange agreed knowledge models that describe concepts they share. In this wider context, templates are often used to define knowledge models at a finer level of granularity than the HL7 interoperability definitions.

Since all HL7 v3 templates rely on Static Models, all templates are a formal definition of a particular concept. Templates may be used to further refine any models in any domain of interest to HL7, including messaging infrastructure, healthcare administration and especially for clinical concepts.

In essence templates are used for three areas in information exchange:

- Construction of Instances
- Validation of Instances
- Processing of Instances

These activities are done by systems/individuals in the following roles:

- Instance Content Creator
- Instance Content Consumer

Applications may need to use other resources to define how the information model described in the template is properly presented or applied in other contexts, such as input screens. These things are out of scope of this template specification.

However some vendors already use templates expressed in the templates exchange format (see chapter 7) to build reference user input interfaces or to generate code for proper instance generation etc.

### 4.2. Construction of Instances

Models and templates are used to guide and direct information input for a message or document. This may take the form of a specification for a user input interface, or where a set of templates is used to guide an application in constructing a proper instance. Templates can be used to provide knowledge models that are applied on a context sensitive basis to a consistently applied more general model, and provide a coherent framework for customizing application behavior.

A good example is the Continuity of Care Document Definition by HL7 as part of the Consolidated CDA Templates for Clinical Notes, where specific templates are used to specify header parts and a collection of sections with their corresponding entries.

It is important to recognize that templates do not carry semantics. An instance that is build using a templates shall carry all semantics on its own and must be understandable even without any reference to or knowledge of the underlying templates.

Applications using templates for instance creation may choose to reference the template in the instance to support validation and processing and it is current practice to include all template references in an instance.

### 4.3. Validation of Instances

Templates are used to support validation of instances. When a template is applied, either by definition or by reference, the instance can be validated against the constraints expressed in the template in addition to the base static model.

Template definitions may overlap as they apply to an instance. More than one template may be applied to a given class, or a template defined elsewhere in the instance may still apply when a new template is applied. Where more than one model (templates and the underlying model) apply to a feature, the possible valid set of instances for the feature is the set intersection of the set of instances described by each applicable model. Note that it may be that there is no instance which can satisfy all the applied templates if the models contain incompatible constraints (the set of valid instances is the empty set).

### 4.4. Processing of Instances

Template ids in validated instances may also be used to process the instance properly. The presence of a specific template reference makes sure that the document conforms to the specification of the asserted template and it can be used to trigger the extraction of the information accordingly.

### 4.5. Instance Content Creators

Instance Content Creators use templates to assert the patterns and semantics present in the data instances they create.

If a system creates a document instance that conforms to all the specification of Template A, then the Instance Content Creator can assert Template A in the document. This assertion can be used by Instance Content Consumers to process the data contained in the document. The assertion can also be used by Content Validators to confirm that the document does in fact meet all of the constraint requirements specified for the template.

### 4.6. Instance Content Consumers

If a document is valid, then a Content Consumer can be certain that the data in the document conforms to the specification of the asserted templates (see also section on Content Validators below).

If the Content Consumer application has the ability to process data that conforms to a certain template, and the document contains that template, then the Content Consumer can receive that information.

If the document contains information that conforms to a template, which the Content Consumer does not have the capability to process, then data conforming to that template may not get processed.

If there is no expressed requirement for a Content Consumer to process certain templates, then it is permissible for the Content Consumer to process only that data which conforms to the templates it claims to support.

## 4.7. Content Validator Methods

Regarding the validation of instances in a certain context and again the underlying models and templates it must be mentioned that some of the constraints are testable in the scope of an instance, while others require examination of the inputs and outputs of a system by humans to determine conformance.

For the testable constraints of an instance validation methods are available to do the test whether an instance is conformant to definitions. In essence two methodologies are mainly used in practice:

- W3C schema validation, mainly derived from the underlying static model
- ISO schematron validation, derived from the template definitions.

For the Template exchange format in chapter 7, a transformation exists to create schematrons directly out of the template definitions (see section 7.5). The underlying schematron engine creates ISO schematron rules and also compiles all value set references that are needed for a proper validation.

According to the definition of the nature of a constraint, “errors”, “warnings”, or just “information” can be the output of a validation process against an instance. Any type of constraint described above may determine the nature of the validation message.

A governance group determines the behavior of the validation process if an instance fragment is not conformant. They may decide to issue errors, or just warnings or informational text.

Regarding the use of vocabulary, a test of an instance fragment whether it uses the correct vocabulary depends on the type of binding. If a value set binding is specified and the value set is completely available for the validation process, it is no problem to check codes in instances. In all other cases it might be difficult to check codes against value sets or even abstract conceptual domains that are not physically available.

## 4.8. The use of multiple Template Ids

More than one template can be used at the same time in the same document, so long as all the rules defined by each template hold true in the document where the templates are asserted. The layering of templates can be used by Instance Content Creators when they create a document in the same way that Template Designers can layer templates to define a specific template.

Template layering supports incremental interoperability. Layers can be added when data needs to be further constrained for more specific purposes. The templates can be used to meet very specific exchange requirements without additional negotiation. By establishing standard templates to meet the data interoperability needs, Instance Content Creators and Instance Content Consumers simply utilize the templates to generate and process the documents as defined by those standard templates. The inherent nesting properties of template designs make it possible for the data to be created and understood at varying levels of specificity.

The order of multiple template identifiers for one instance fragment supplied in an instance is not significant. However, to understand the combined effect, it is useful to envision the data as constrained initially by the most constraining template. Assuming

that the set of templates have designs that “nest together”, it then follows that the data also conforms to templates which are increasingly more relaxed, or generalized.

For example, if a system supports template A and template B which is a further constraint on template A, then it can create a document which includes data that conforms to both Template B and Template A. A system which understands only template A could still process the data, but not at the level of specificity that could have been achieved if it understood template B.

When creating or processing a list of templates, it may be unclear which template is the most constrained or it might be even none of them being the “most” constraining template. Currently, there is no computational algorithm for determining this, but it can be deduced through analysis of the template definitions.

It is not required that a document include all templates to which it conforms. The Instance Content Creator asserts the intended templates in a document at the time it is created. It may be possible for a receiver to determine that there are other models to which the instance of data is also conformant, even though they are not identified in the instance.

#### 4.9. Implementation principles and use of template ids in instances

A template is referenced in an XML instance by the XML element `templateId` wherever appropriate along with at least the template id in the `@root` attribute, e.g.

---

##### Example

```
<templateId root="1.2.3.4.5"/>
```

---

There is the possibility to reference to a specific version of a template (for example review or trial use cases, connect-a-thons using previews of templates). In this case the `@extension` attribute of the `templateId` element extends the id to a specific version, e.g.

---

##### Example

```
<templateId root="1.2.3.4.5" extension="2013-10-12"/>
```

---

or

---

##### Example

```
<templateId root="1.2.3.4.5" extension="v1.0"/>
```

---

If `@extension` is omitted then the binding to the template is as defined by the governance group. This accommodates the migration phase of various versioning practices to this new specification.



## 5. Use Cases Demonstrating the Creation, Use, Maintenance and Governance of Templates

This chapter is non-normative and shows typical use-cases of the Creation, Use, Maintenance and Governance of Templates.

### 5.1. Creation and life cycle of template “Estimated Delivery Date”

This example is about the life cycle of template definitions to represent data about an Estimated Delivery Date (EDD).

#### 5.1.1. Step 1: initial draft

*Designer perspective:* The governance group / template designer decides to create a brand new (nascent) template for the purpose of representing data regarding the “Estimated Delivery Date (EDD)”.

The template creation was started a 2013-05-02 (effective start date) and the initial template got the OID 1.2.3.7 with status “new”. After a while they decided to publish the first draft of the template, still finalizing some details on the design.

Table 8: Example template

Template “Estimated Delivery Date”
<b>Meta data</b> <ul style="list-style-type: none"><li>• Identifier: 1.2.3.7</li><li>• Effective Date: 2013-05-02</li><li>• Status code: draft</li><li>• Name: EstimatedDeliveryDate</li><li>• Version label: 1.0</li></ul> <b>Design</b> <ul style="list-style-type: none"><li>• Observation<ul style="list-style-type: none"><li>• @classCode = OBS</li><li>• @moodCode = EVN</li><li>• code = LOINC 11778-8</li><li>• value to carry a valid “estimated delivery date” (demotion to xsi:type = TS)</li></ul></li></ul>

*Content creator perspective:* An application may start with the implementation of the template and test/review it but must be aware that a template with status “draft” may change, even substantially.

#### 5.1.2. Step 2: review phase

*Designer perspective:* The designer decides to publish the draft and start the test phase to get feed-back from the governance group, external reviewers, or content creators or consumers.

While the design remains the same, the meta data status code may be changed to reflect the new status of the template.

Table 9: Example template



## Template “Estimated Delivery Date”

### Meta data

- Identifier: 1.2.3.7
- Effective Date: 2013-05-02
- Status code: **review**
- Name: EstimatedDeliveryDate
- Version label: 1.0

*Reviewer group / content creator or consumer perspective:* The template should be reviewed / tested.

### 5.1.3. Step 3: revision with a minor change

During the test/review phase it became obvious that an optional date of determination of the EDD is needed.

The original template version was designed as “open”, so an optional additional element can be added without “breaking” the former template design, i.e. instances based on the new template version are still valid against the former version.

*Designer perspective:* At 2013-05-20, the designer adds an *effectiveTime* as TS to the design, the rest is unchanged. Because the template was in state “review” he needs to create a new version of the template.

The use of the former template version may be declared as “discouraged” with the changing the meta data status code to “inactive”.

Table 10: Example template

## Template “Estimated Delivery Date”

### Meta data

- Identifier: 1.2.3.7
- Effective Date: 2013-05-02
- Status code: **inactive**
- Name: EstimatedDeliveryDate
- Version label: 1.0

The new version has the following meta data and design

Table 11: Example template

## Template “Estimated Delivery Date”

### Meta data

- Identifier: 1.2.3.7
- Effective Date: **2013-05-20**
- Status code: **draft**
- Name: EstimatedDeliveryDate
- Version label: **1.1**
- **Relationships: replaces template 1.2.3.7 as of 2013-05-02**

## Design

- Observation
  - @classCode = OBS
  - @moodCode = EVN
  - code = LOINC 11778-8
  - **effectiveTime of type TS as “date of determination”**
  - value to carry a valid “estimated delivery date” (demoted to xsi:type = TS)

Content creator / consumer perspective: Due to the notification that the original template got a new version that does not necessarily imply a required change of their application code, they may or may not adopt the new version of the template.

### 5.1.4. Step 4: revision with substantial changes

The test and review phase came up with the change proposal to

- make a change to the allowed *Observation.code* so that it not only allows LOINC 11778-8 but also Snomed-CT code 161714006 (change of vocabulary binding)
- allow to convey the method of evaluation of the EDD as a required item, with a proposed list: last ovulation, last conception, last menstrual period, quickening, ultrasound.

The governance group decides to follow the suggestions.

**Designer perspective:** At 2013-05-30, the designer makes the two suggested changes by creating a new (major) version of the template, leaving the rest of the design unchanged. He also compiles a new value set *EDDMethod* to reflect the concepts in the proposed list.

Table 12: Example template

## Template “Estimated Delivery Date”

### Meta data

- Identifier: 1.2.3.7
- Effective Date: **2013-05-30**
- Status code: **draft**
- Name: EstimatedDeliveryDate
- Version label: **2.0**
- Relationships: replaces template 1.2.3.7 as of 2013-05-02
- **Relationships: replaces template 1.2.3.7 as of 2013-05-20**

### Design

- Observation
  - @classCode = OBS
  - @moodCode = EVN
  - code = LOINC 11778-8 or Snomed-CT 161714006
  - effectiveTime of type TS as “date of determination”
  - **methodCode** as “method of evaluation of the EDD”, shall be drawn from value set *EDDMethod*.
  - value to carry a valid “estimated delivery date” (demoted to xsi:type = TS)

## 6. Management and Governance of Templates

### 6.1. Role of governance groups

#### **What is the role of a governance group in the management and governance of templates?**

A governance group is the custodian of a set of templates. A governance group can act in any of these three roles: Template Creator, Template Adopter, and Template Adapter. It establishes a Template Repository to hold the template designs it creates and uses. It may maintain a Template Registry to track the template designs it uses. It set the policies and procedures it will follow when using, adding, and addressing the life-cycle of each template design under its control (see also 2.7.2 Governance Group).

### 6.2. Template Repository

#### **What is a Template Repository?**

A template repository houses the definition of template designs. As templates are added or modified, their designs remained documented in the Template Repository maintained by the governance group which is the custodian of the designs (see also 2.7.3 Template Repository).

### 6.3. Template creation

#### **How does a new template get created?**

To create a new template or a new design for an existing template, a template designer works with a subject matter expert, or a group of subject matter experts, to consider a specific use case and representational requirements for the information. If no existing template can be used to meet the requirements, then a new design is developed. The template designer documents the needed conformance statements that make up the template (see 2.7.1 Template Designer).

#### 6.3.1. Use cases / business requirements

#### **What role do use cases or business requirements play in the management of templates?**

Use cases and business requirements are used to establish the purpose of a template. As the use cases expand, or the business requirements change, the purpose that a template serves may change, a new template design may need to be developed, or a new template may need to be developed.

#### 6.3.2. Template content provider skills

#### **Who is the template content provider to provide the functional requirements for the purpose and clinical information design?**

Template designers have the technical expertise to analyze use cases and business requirements in order to construct the needed conformance statements to constrain and apply an information model to meet the information exchange requirements. However, it requires subject matter expertise to provide the functional requirements which inform the design. It often requires a great deal of input from multiple clinical subject

matter experts and multiple technical template designers to establish the templates needed for a complex use case.

### **6.3.3. Mapping clinical requirements to technical artifacts**

#### **Based on existing HL7 models and representations, how do the clinical requirements map to the technical artifacts of the template design?**

Each clinical requirement may generate one or more technical requirement for a template's design. Information models and the conformance statements which constrain them can be very complex. Multiple dimensions must be considered including aspects such as but not limited to the structure and semantics associated with representing the information, the relationships across information, the provenance of the data, and the longitudinal nature of the information.

### **6.4. Template endorsement, publication, testing and maintenance**

#### **How do templates get endorsed, published, tested, maintained over time?**

Each governance group establishes the rules and best practices for reviewing, approving, publishing, testing and maintaining the set of templates it governs. This process often follows well established practices that align with broader governance policies for the organization. The governance of versioning is part of how an organization manages the information it controls as it changes over time.

### **6.5. Re-use of templates**

#### **How do existing templates get reused or adapted to create a new design?**

When an information model is first developed, the number of templates in existence to constrain and use it is small. As the information model becomes more adopted and more uses are identified, the number of templates increases. This creates the need to reuse and adapt existing templates, as a natural maintenance requirement for interoperability. Prior designs must always be considered when creating new designs or else the information conforming to these established patterns will not benefit from the comparability implied by the use of the templates.

#### **6.5.1. Referencing templates**

##### **How do you reference an existing template?**

When the design of a new template is related to the design of another template, it is best to reference the prior template rather than copying its conformance into the new design. This practice prevents the need to make a cascading set of changes if a future change to the underlying template is made. Referencing the prior template using its id to inherit the prior design without restating the prior conformance in the new template creates maintenance efficiencies. It also creates implied relationships which makes the information conforming to the new template meaningful relative to information conforming to the prior template. This practice also makes it easier to understand a large body of templates. When one template's design is defined as an adaptation or further constraint of a prior template design, only the incremental differences need to be understood, assuming an understanding of the prior template already exists.

## 6.5.2. Obtaining template designs

### How do you obtain template designs?

Template definitions are published and can be read to gain an understanding of the design, but this is not an efficient way to work with the definition when developing additional templates. To exchange a set of template definitions and the detailed conformance statements each template comprises, a standard has been devised. The implementable technology specification (ITS) for this exchange standard allows systems that maintain a template repository to export the template designs it manages and import template designs which have been created in other systems (see chapter 7).

#### 6.5.2.1. Licensing issues

### What are the template licensing issues?

Like other types of designs, templates are intellectual property. As intellectual property, copyrights and other forms of licensing and legal protections may be applied.

#### 6.5.2.2. Use of template design

### How can I make use of the design?

Prior templates designs can be referenced in accordance with the rules established by the governance group that controls them. Referencing a prior template using its identifier rather than copying its conformances into a new design is the best practice.

## 6.5.3. Publish and Subscribe

### How does Publish and Subscribe functionality support template management?

When template designs include interdependencies, it becomes necessary to stay informed of changes in the related templates. Template Registries may support functionality which permits a system to declare its interest in a template or set of templates so that it may be informed when changes occur. Once a system has subscribed, the Template Registry can publish a change, and the needed notification happens automatically.

## 7. Standards for Exchanging Template Definitions

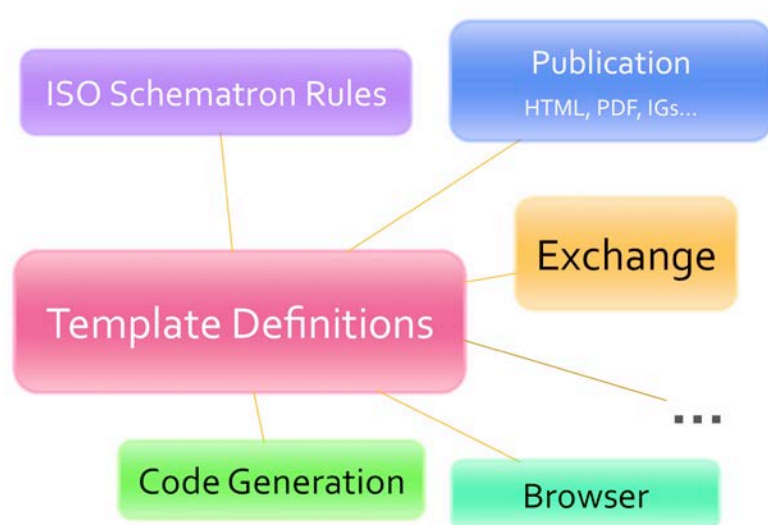
### 7.1. The Template Definition Exchange Standard

The **Template Definition Exchange Standard** is an implementable technology specification (ITS) for the exchange and storage format of template artifacts.

#### 7.1.1. Objectives

The main objective of the Templates ITS is to provide machine processable template definitions for various purposes:

- Publication and Documentation, Browsing
- Exchange, Exposition by Template Repositories
- Generation of Validation Methods, e.g. ISO Schematron
- Others, e.g. Code Generation



*Figure 5: Template definitions and derivations for publication, validation, browsing, exchange and possible other purposes.*

##### 7.1.1.1. Template Publication and Documentation

Main objective is a proper documentation of the template definition. This means for example the availability of a template design browser that shows selected templates. The template definitions should also be available in other visual formats such as HTML or PDF or should be easily embeddable in implementation guides.

During the last years different governance groups published template definitions in various formats (see also Appendix C), using also different types of expressions for conformance requirements.

The definitions of a template are typically transformed into a tabular view (see example in section 2.14), but a governance group may decide to present them also in other formats or transform them into human language based conformance statements.

For the best practice examples in section 7.7 possible tabular view renditions are shown. In Appendix C other representations are shown.

### 7.1.1.2. Exchange Format

During the last years a few template authoring tools have been developed, e.g. Model Driven Health Tool MDHT, Lantana's Trifolia Workbench, ART-DECOR etc. While the underlying rationale and the details of how template design is acquired, presented and stored may be different, an exchange of template definitions between these tools is useful if not required.

This specification provides a standardized exchange and storage mechanism. In several proof-of-concept projects prior to the publication of this specification it could be shown, that in most cases conversion into the ITS here is possible with only little corrections by hand. It can be expected that – once this ITS is endorsed – tools may converge to read and write template specifications following this ITS without any manual interaction. This might include a transition to typical concepts described here, for example the choice mechanism or that a template must include all defined items that are allowed in order to be able to handle “closed” templates.

### 7.1.1.3. Source for Generation of Validation Methods

Template definitions serve three purposes regarding instances: creation, validation and processing of instances. Providing a standardized documentation format for template definitions allows to apply transformations on the templates to generate validation methods for instance testing.

During the last years schematron [isosch] has been established as the mainly used constraints language. It allows the definition of a set of rules which are applied to an XML instance. As a result errors, warnings and informational notes may be presented coming from detected non-conformant fragments of the tested instance or an indication that the instance (fragment) is conformant (valid).

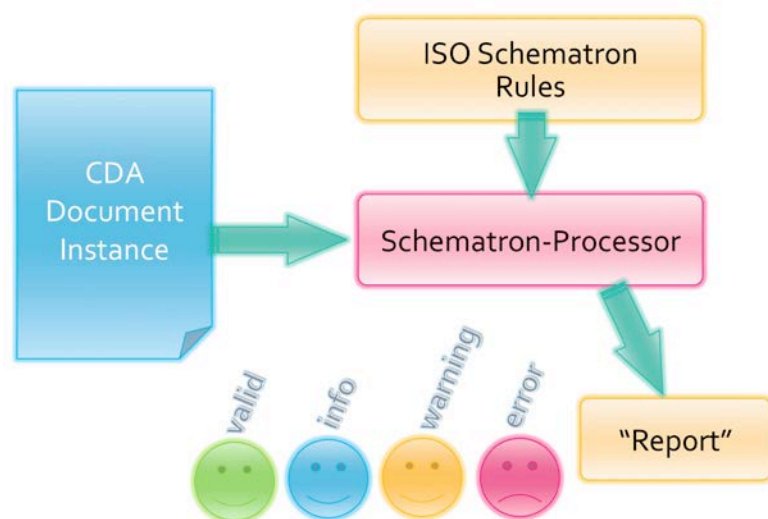


Figure 6: Schematic overview of a CDA instance validation with ISO schematron

For this ITS a transformation suite (schematron engine) is available that converts the template definitions into schematrons. The results can be used to validate instance and test their conformance.



#### 7.1.1.4. Easy Access to Templates by Repositories

Once a set of templates is ready to be published the aspect of a proper template registry and/or repository comes into play. This implies to not keep the template definitions locked up in a proprietary tool or published them in paper format only, but to allow other groups, e.g. template consumers (implementers), to access the template definitions electronically along with all other needed artifacts (e.g. value sets) for machine processing, code generation, validation and testing etc.

For that purpose templates – once endorsed for use – ideally are exposed in a template repository/registry to allow access for the purposes mentioned above. In addition to that, a governance group may decide to make use of a template definition of another governance group by including endorsed templates in their specifications or to clone and refine or adapt them. This is easy by offering template definitions in a template repository/registry in a standardized format.

#### 7.1.2. Out-of-scope

Other constraint languages than schematron, for example OCL or GELLO, are not covered in this guide.

#### 7.1.3. Reference Installation

##### *Reference Implementation*

*The Template Definition Exchange Standard (ITS) specified here has already been implemented in an open source project call ART-DECOR (see [art-decor.org](http://art-decor.org)) since the year 2009 and it could be proven that it fulfills all practical use cases of more than 20 clinical templates specification and implementation projects regarding template design, template versioning, template relationship and deployment as well as use of templates in creation, validation and processing XML instances including the usage in Template Repositories.*

*It was used in CDA R2 projects as well as in Clinical Statement (Care Provision) messages and non-HL7 XML specifications. Also the following transformation scripts (XSLT and Xquery) and services are available*

- *Template ITS → HTML tabular view (with a further path to PDF representation)*
- *Template ITS → ISO Schematron (“schematron-engine”, including open / closed template behavior)*
- *RESTful repository functionalities*

## 7.2. Template Meta Data and Design Body

According to the previous chapters in this specification, templates have meta information about their identity, effective date, status etc. referred to as the **Template Meta Data** and the actual set of constraint definitions referred to as the **Template Design Body**.

The following sections specify the template meta data and design body elements that are used to define a whole template.



### 7.2.1. Data types used for this specification

The elements and attributes of the ITS are associated with data types. Most of these data types are drawn from the MIF data types [mif] and are mainly restrictions of string types. The list of used data type definitions is as follows.

*Table 13: Used data types used for templates definition elements and attributes*

Data type	Description
<b>AnyURI</b>	Equivalent to the W3C schema type xs:anyURI
<b>BasicIdOrOid</b>	Either a string or an OID in order to identify the object
<b>Boolean</b>	Boolean “true” or “false”
<b>StaticOrDynamic Flexibility</b>	Either the fixed string "dynamic" or a valid W3C schema type xs:dateTime as an effective date of the version of the object
<b>Enumeration</b>	Used as the base class for all “codes” which qualify or shall be handled as a W3C schema NMTOKEN
<b>FreeText</b>	Allows unconstrained markup to identify the language in which it is expressed in the @language attribute of the element of that type
<b>ShortFormalName</b>	Used when names are intended to be non-empty, short and without “weird” characters
<b>NonEmptyString</b>	A non-empty string
<b>Oid</b>	ISO Object Identifier, used for identifiers
<b>SelfReference TemplateIdOrOid</b>	A string “*” or “**” or a valid OID
<b>TimeStamp</b>	A time stamp according W3C schema type xs:dateTime
<b>SmallNonNegative Integer</b>	Integer number up to 6 characters long
<b>UnlimitedInteger</b>	Integer number up to 6 characters long or “*” to indicate “unbounded”, used for multiplicity definitions

### 7.2.2. Overview

The following table gives an overview of the template meta data structure.

*Table 14: Template Meta data elements overview*

Item	DT	Card	Conf	Description
<b>template</b>				Root element
@id	Oid	1..1	M	Identification
@name	ShortFormal Name	1..1	M	Business name

Item	DT	Card	Conf	Description
@effectiveDate	TimeStamp	1..1	M	Start Date
@statusCode	Enumeration	1..1	M	Status
@displayName	NonEmpty String	0..1	R	Print Name
@versionLabel	NonEmpty String	0..1	R	Version Label
@expirationDate	TimeStamp	0..1	R	Expiration Date
@officialReleaseDate	TimeStamp	0..1	R	Official Release Date
@isClosed	Boolean	0..1	R	Open/Closed Template
<b>desc</b>	FreeText	0..*	R	Multilingual description of purpose and scope
<b>classification</b>		0..*	R	Classification
@type	Enumeration	0..1	R	Type of Template
@format	Enumeration	0..1	R	Format of Template
<b>tag</b>	NonEmpty String	0..*	R	Tags for search purposes
<b>relationship</b>		0..*	R	Relationships
@type	Enumeration	0..1	C	Type of Relationship
@template	BasicIdOrOid	0..1	C	Related Template name or id
@model	NonEmpty String	0..1	C	Related Template name or id
@flexibility	StaticOr Dynamic Flexibility	0..1	C	Static or dynamic flexibility indicator
<b>context</b>		0..1	R	
@id	SelfReference TemplateIdOr Oid	0..1	C	Type of template id as context, either "*" or "**" or an OID
@path	AnyURI	0..1	C	
<b>item</b>		0..1	R	Template item label and description
@label	NonEmpty String	1..1	M	String to identify item (template) during validation

Item	DT	Card	Conf	Description
desc	FreeText	0..1	R	Further explanation of the item label
<b>example</b>		0..*	R	XML example
@type	Enumeration	0..1	R	Type of example, valid or error
@caption	NonEmpty String	0..1	R	Caption for the example
(element content)	(any well formed XML)	1..1	M	The example XML fragment

### 7.3. Template elements and their attributes

#### 7.3.1. `template/@id`

A mandatory globally unique, non-semantic, identifier for the template as the primary identifier. It is the identification of the purpose or intent of the template and is typically an OID. The id or the name of a template is chosen when referencing the template in another design, such as inclusion or containment.

##### Example fragment

```
<template id="2.16.840.1.113883.10.20.22.4.31" ...
```

#### 7.3.2. `template/@name`

A required name as a business name for the template as a secondary identifier. Please note that there is no guarantee that the name is globally unique but it shall be unique within a governance group. Name or id of a template is chosen when referencing the template in another design, such as inclusion or containment.

##### Example fragment

```
<template ... name="AgeObservation" ...
```

#### 7.3.3. `template/@effectiveDate`

The template has a mandatory timestamp (date and optional time) after which the template existed regardless of its state (e.g. still in design phase or ready for use). See also definitions in section 2.9.3.

##### Example fragment

```
<template ... effectiveDate="2013-11-13T00:00:00" ...
```

#### 7.3.4. `template/@statusCode`

The template has a mandatory status code. It is one of the following codes.

Table 15: Enumeration values for the status code of templates

Code	Description
<b>draft</b>	draft
<b>active</b>	active
<b>review</b>	review
<b>retired</b>	retired
<b>cancelled</b>	cancelled
<b>rejected</b>	rejected
<b>terminated</b>	terminated

---

#### Example fragment

```
<template ... statusCode="active" ...
```

See also definitions in section 2.9.7

#### 7.3.5. **template/@displayName**

The optional human readable print name for the template for orientation purposes. This is not intended for machine processing and typically created in the project's language or in English.

#### 7.3.6. **template/@versionLabel**

A version of a template may contain an optional human readable version label for the template to be able to determine the correct version of a template.

See also definitions in section 2.9.4.

---

#### Example fragment

```
<template ... versionLabel="v2.0" ...
```

#### 7.3.7. **template/@expirationDate**

The optional date at which the concept represented by this template becomes stale, and should be reviewed for (clinical) relevance and accuracy.

See also definitions in section 2.9.5.

#### 7.3.8. **template/@officialReleaseDate**

An optional official release date of the template.

See also definitions in section 2.9.6.

#### 7.3.9. **template/@isClosed**

An optional Boolean value that indicates whether the template is defined as closed or not. The default is "false" (i.e. template is defined as "open").

An “open” template should not have @isClosed or @isClosed set to “false”. A “closed” template shall have an @isClosed set to “true”.

---

#### Example fragment

```
<template ... isClosed="true" ...
```

On discussion and further explanation on “open” vs. “closed” template definitions refer to section 2.11.

#### 7.3.10. **template/desc**

An optional free text natural language description of the intent and scope of the template. The purpose is to provide the author’s initial intent for the template. The format is string with optional HTML tagging (XHTML).

#### 7.3.11. **template/desc/@language**

The language indicator in which language the description is done. Examples are “en-US”, “de-DE” etc.

#### 7.3.12. **template/classification**

The classification is an optional and repeatable element that allows the classification of the template (see also section 2.9.8.2).

#### 7.3.13. **template/classification/@type**

This attribute is an optional indication of the type of the template. For types of template refer to section 2.12. The type is a code (enumeration) with a choice one of the following items:

*Table 16: Enumeration values for the types of templates*

Code	Description
<b>cdadocumentlevel</b>	CDA document level
<b>cdaheaderlevel</b>	CDA header level
<b>cdasectionlevel</b>	CDA section level
<b>cdaentrylevel</b>	CDA entry level
<b>messagelevel</b>	V3 Message level
<b>clinicalstatementlevel</b>	In a V3 message, a clinical statement constraint

---

#### Example fragment

```
<classification type="cdaheaderlevel"/>
```

#### 7.3.13.1. **template/classification/@format**

The format of the instance the template constrains.

As of now the only format that is supported is HL7 V3 with XML ITS 1, i.e. "hl7v3xml1". This is the default.

In the future there might be other types of instances, e.g. "fhirxml" for FHIR-XML, "greencda" for green CDA instances or "genxml" for generic XML etc.

### 7.3.13.2. template/classification/tag

Tag elements could be incorporated to specify any number of tags for search purposes.

#### Example fragment

```
<classification type="cdaentrylevel">
  <tag>Blood pressure</tag>
</classification>
```

### 7.3.13.3. template/relationship

This optional and repeatable element defines relationships of the template regarding other templates or model artifacts. The type of relationship is indicated in the @type attribute.

The relationship element shall have a value in either @template or @model, but not both. The attribute @flexibility reflects the binding type for @template or @model, depending on which is valued.

### 7.3.13.4. template/relationship/@type

The type of the relationship of the template, at this point in time one of the following.

Table 17: Enumeration values for the types of templates

Code	Description
<b>Template-Template-Relationships</b>	
<b>REPL</b>	This template replaces @template
<b>VERSION</b>	This template is a version of @template*
<b>SPEC</b>	This template specializes @template
<b>GEN</b>	This template generalizes @template
<b>COPY</b>	This template is a design copy of @template
<b>ADAPT</b>	This template is an adaptation of @template
<b>EQUIV</b>	This template is equivalent to @template regarding design
<b>BACKWD</b>	This template is backward compatible to @template
<b>FORWD</b>	This template is forward compatible to @template
<b>CONTAINS</b>	This template contains (uses) @template*
<b>INCLUDES</b>	This template includes (uses) @template*
<b>Template-Model-Relationships</b>	

Code	Description
<b>DRIV</b>	This template is derived from @model
* in a proper registry, this relationship is derived automatically	

#### Example fragment

```
<relationship type="SPEC" template="2.16.840.1937.10.101"
  flexibility="dynamic"/>
<relationship type="ADAPT" template="1.3.1937.10.102"
  flexibility="2013-02-10T00:00:00"/>
<relationship type="DRIV" model="some-model-name-or-id"/>
```

For further discussion refer to chapter 3.

#### 7.3.13.5. **template/relationship/@template**

A reference by name or id to a template identifier, used for template-template relationships

#### 7.3.13.6. **template/relationship/@model**

A reference to a model artifact identifier or description specifying from what this template is derived from or based on, e.g. an HL7 R-MIM, a DCM, used for template-model relationships

#### 7.3.13.7. **template/relationship/@flexibility**

Static or dynamic binding for the template or model that is referred to. Default value is "dynamic".

#### 7.3.14. **template/context**

An optional context of a template, i.e. where in an XML instance the template rules are considered to apply to.

For further discussion refer to section 2.9.8.4. Typically there are three types of context specifications.

##### 7.3.14.1. **Sibling node context**

The template rules apply to all sibling elements and descendent nodes in the instance.

#### Example fragment

```
<context id="*" />
```

##### 7.3.14.2. **Parent node context**

The template rules apply to the siblings of the parent node and all descendent nodes in the instance. CDA entry level templates are often defined in parent node context.

#### Example fragment

```
<context id="**"/>
```

#### 7.3.14.3. Pathname specified context

A pathname (making use of XPath expressions) is specified, which allows to trigger templates in an instance without the need to have template id elements in the instance.

##### Example fragment

```
<context path="cda:recordTarget" />
```

#### 7.3.15. template/item

The optional item is used as an identification mechanism when it comes to error or warning or information messages by derived validation scripts. For example if each constraint has a (unique) number, it may be used to precede every validation message.

##### Example

A CDA “encounter” section level template defines the section code to be 46240-8 drawn from LOINC. This constraint is uniquely identified by an identifier at or near the end of the constraint (e.g., CONF:7941). In the template definition this is done specifying:

```
<item label="CONF:7941" />
```

Every error, warning or information that is raised during validation carries this constraint identifier.

##### 7.3.15.1. template/item/@label

Template item label as string. The default shall be the template name or id.

##### 7.3.15.2. template/item/desc

Further explanation of the item label.

##### 7.3.15.3. template/example

Templates may have zero to many example fragments to illustrate valid (or explicitly invalid) instances. In addition, each template element definition (see below) may have also an example fragment.

##### 7.3.15.4. template/example/@type

The optional type of the example, see the following table for valid codes.

*Table 18: Enumeration values for the types of template examples*

Code	Description
<b>valid</b>	The example is explicitly marked as valid. This is the default.
<b>invalid</b>	The example is explicitly marked as invalid



### 7.3.15.5. template/example/@caption

An optional text used as caption for the example.

The following statement could for example be rendered appropriately in the template documentation

```
<example type="valid" caption="A valid example with a piece of text">
  <root>
    <element value="abc"/>
    <text mediaType="text/plain">I am the text example</text>
  </root>
</example>
```

Example	A valid example with a piece of text
	<pre>&lt;root&gt;   &lt;element value="abc"/&gt;   &lt;text mediaType="text/plain"&gt;I am the text example&lt;/text&gt; &lt;/root&gt;</pre>

## 7.4. Template design body definitions

The actual template design, also seen as the “body” of the template, is a collection of constraints that ideally describes the structure and semantics of all instance elements. It contains the following items

- template **element definitions** with possible vocabulary, property or element content (text) constraints,
- template element **attribute definitions** with possible vocabulary constraints,
- **choices** of template elements,
- **inclusions** of other templates and
- to support at least one typical constraint language **schematron** rules (assert, report, let) are allowed.

The following table gives an overview over all possible template design body elements.

Table 19: Template Design Body elements overview

Item	DT	Card	Conf	Description
<b>element</b>				An XML element definition
@name	AnyURI	1..1	M	The name of the XML element
@minimum Multiplicity	SmallNon Negative Integer	1..1	M	The minimum multiplicity of the element
@maximum Multiplicity	Unlimited Integer	1..1	M	The maximum multiplicity of the element or “*” to express unbounded

Item	DT	Card	Conf	Description
multiplicity				
@isMandatory	Boolean	0..1	R	Whether the element is mandatory
@conformance	Enumeration	0..1	R	The conformance type of the element
@datatype	Enumeration	0..1	R	The data type of the element
@contains	AnyURI	0..1	R	A containment of another template
@flexibility	StaticOr Dynamic Flexibility	0..1	R	The associated flexibility of the contained template
@isClosed	Boolean	0..1	R	Whether the element and all subsequent definitions are considered to be defined as open or closed
desc		0..*	R	Multilingual description of the element
item		0..1	R	Element item label and description
example		0..*	R	XML example
vocabulary		0..*	R	Vocabulary binding specification for that element
property		0..*	R	Property specification for that element
text		0..*	R	Element content specification for that element
<p>Following allowed child elements of “element” are</p> <ul style="list-style-type: none"> <li>• <b>element</b></li> <li>• <b>attribute</b></li> <li>• <b>choice</b></li> <li>• <b>include</b></li> <li>• <b>assert</b></li> <li>• <b>report</b></li> <li>• <b>let</b></li> </ul>				
<b>attribute</b>				An XML attribute definition

Item	DT	Card	Conf	Description
@name	Enumeration Token	1..1	M	
@value	NonEmpty String	1..1	M	
@isOptional	Boolean	0..1	R	
@prohibited	Boolean	0..1	R	
@datatype	Enumeration	0..1	R	The data type of the attribute
desc		0..*	R	Multilingual description of the attribute
item		0..1	R	Attribute item label and description
vocabulary		0..*	R	Vocabulary binding specification for that attribute
<b>choice</b>				A choice
@minimum Multiplicity	SmallNon Negative Integer	1..1	M	The minimum multiplicity of the elements in choice
@maximum Multiplicity	Unlimited Integer	1..1	M	The maximum multiplicity of the elements in choice or “*” to express unbounded multiplicity
Following allowed child elements of “element” are <ul style="list-style-type: none"> <li>• <b>element</b></li> <li>• <b>choice</b></li> <li>• <b>include</b></li> </ul>				
<b>include</b>				Inclusion of another template
Following allowed child elements of “element” are <ul style="list-style-type: none"> <li>• <b>element</b></li> <li>• <b>attribute</b></li> <li>• <b>choice</b></li> <li>• <b>include</b></li> </ul>				
<b>assert</b>				Schematron assert
@flag	NonEmpty String	0..1	R	Schematron flag: to convey state or severity information to a subsequent

Item	DT	Card	Conf	Description
				process
@see	NonEmpty String	0..1	R	URI value of the external information of interest
@role	Enumeration	1..1	R	“error”, “warning” or “information”
@test	NonEmpty String	1..1	R	The test expression
(element content)		0..1	R	The schematron message
<b>report</b>				Schematron report
@flag	NonEmpty String	0..1	R	Schematron flag: to convey state or severity information to a subsequent process
@see	NonEmpty String	0..1	R	URI value of the external information of interest
@role	Enumeration	1..1	R	“error”, “warning” or “information”
@test	NonEmpty String	1..1	R	The test expression
(element content)		0..1	R	The schematron message
<b>let</b>				Schematron let
@name	NonEmpty String	1..1	R	Schematron let name of the variable
@value	NonEmpty String	1..1	R	Schematron let value

#### 7.4.1. element

##### 7.4.1.1. element/@name

The @name attribute carries the name of the element, typically prefixed by the default project namespace (e.g. “hl7:” or “cda:”).

##### Example fragment

```
<element name="hl7:recordTarget" ... >
```

The name may be an Xpath expression with predicates.

##### 7.4.1.2. element/@minimumMultiplicity

This optional attribute identifies the minimum number of repetitions of this element that may occur. This is a small non-negative integer.

#### 7.4.1.3. `element/@maximumMultiplicity`

This optional attribute identifies the maximum number of repetitions of this element that may occur. This is either a small non-negative integer or “\*” to indicate “unbound” repetitions.

##### Example fragments

```
<element ... maximumMultiplicity="2" ... />
<element ... maximumMultiplicity="*" ... />
```

#### 7.4.1.4. `element/@isMandatory`

This optional boolean attribute identifies that this element is mandatory, i.e. in an instance it shall be populated with a valid value.

#### 7.4.1.5. `element/@conformance`

This optional attribute identifies the conformance of this element. Allowed values are shown in the following table.

*Table 20: Enumeration values for the conformance of an element*

Code	Description
<b>R</b>	Required
<b>NP</b>	Not present
<b>C</b>	Conditional

#### 7.4.1.6. `element/@datatype`

This optional attribute specifies the data type or a data type flavor for this element.

##### Examples

**CD**  
**CE**  
**ST**  
**TS**  
**IVL\_TS**  
**II.BSN.NL**  
**TS.DATE.MIN**

Please note that this specifies the data type or flavor of the element for documentation purposes but schematron engines may produce appropriate tests on the definitions per element based on this declarations.

#### 7.4.1.7. `element/@id`

This optional attribute is an id that allows the element to be uniquely identified. This is typically done by an OID. The id can be used link between a template element and an associated concept for example in a data set or functional model.

#### Examples

A CDA recordTarget has an element "cda:birthTime" that is associated with the birth time of the patient. If the birthTime template element gets an id, it can be linked to a concept representation for example in a data set that shows the functional perspective.

#### 7.4.1.8. `element/@contains`

A @contains attribute indicates that the element contains the template specified by @contains. This may be a valid template name or template id. Please note that a contained template typically influences the predicate of the element. See sections about re-use of templates with includes and contains in section 2.10.2 for further information.

---

Example fragment: the cda entry contains a template named AgeObservation

```
<element name="cda:entry" contains="AgeObservation" ...
```

#### 7.4.1.9. `element/@flexibility`

An optional static or dynamic binding for the rule set that is referred to by @contains. Default value is "dynamic", i.e. use the most recent version of the specified contained template.

---

Example fragment: the cda entry contains a template with id 1.2.3.4.5 as of 2013-11-23 (the version of that template)

```
<element name="cda:entry" contains="1.2.3.4.5" flexibility="2013-11-23"...
```

#### 7.4.1.10. `element/@isClosed`

This optional boolean attribute allows to specify that the element and subsequent child nodes are considered as open (other elements than specified allowed) or closed (no other elements than specified allowed). The rationale is the same as for open or closed templates, except that this definition applies to element and its child elements only.

See also section 2.11 on open vs. closed templates.

#### 7.4.2. `element/desc`

This optional sub-element carries the multilingual textual description or further explanation on the use of the element.

### 7.4.3. element/item

Indicates the item label of the element for errors, warnings etc., see also section 7.3.15 about Template items.

### 7.4.4. element/example

An template element may have zero or more examples or example fragments for the element, see also section 7.3.15.3 about Template examples.

### 7.4.5. element/vocabulary

This optional sub-element is used for elements with "coded" data types. It allows the assertion of one or more codes, code systems, value sets or concept domains.

Valid combinations

- A @valueSet (by name or id) plus an optional @flexibility attribute which is either the fixed string "dynamic" or a valid date and time as an effective date of the version of the value set (static).
- A @code and (depending on the data type) a @codeSystem; it may contain human readable attributes such a @displayName and @codeSystemName which, if present, constraints the presence of these attributes in instances.
- An @domain specification by name, defining the abstract concept domain(s) from which proper codes may be drawn.

---

#### Example of value set bindings, dynamic and static

```
<element name="hl7:code" ... datatype="CV">
  <vocabulary valueSet="MyValueSet" flexibility="dynamic"/>
</element>

<element name="hl7:code" ... datatype="CE">
  <vocabulary valueSet="MyValueSet" flexibility="2013-11-24"/>
</element>
```

---

#### Example of a fixed code/code system binding

```
<element name="hl7:code" ... datatype="CE">
  <vocabulary code="1234" codeSystem="1.2.3.4.5"/>
</element>
```

If multiple vocabulary elements are specified the definitions are logically connected by an "OR".

---

#### Example: Code must come from Value Set AAA or BBB or shall be code '1234' from codeSystem '1.2.3.4.5'

```
<element name="hl7:code" ... datatype="CE">
  <vocabulary valueSet="AAA" flexibility="dynamic"/>
  <vocabulary valueSet="BBB" flexibility="dynamic"/>
  <vocabulary code="1234" codeSystem="1.2.3.4.5"/>
</element>
```

```
<vocabulary code="1234" codeSystem="1.2.3.4.5" />
</element>
```

#### 7.4.5.1. **element/vocabulary/@valueSet**

This attribute specifies the value set by name or id where codes shall be drawn from.

#### 7.4.5.2. **element/vocabulary/@flexibility**

This attribute specifies the flexibility for the value set.

#### 7.4.5.3. **element/vocabulary/@code**

This attribute specifies the code that shall be used in the instance.

#### 7.4.5.4. **element/vocabulary/@codeSystem**

This attribute specifies the code system (e.g. an OID) that shall be used in the instance.

#### 7.4.5.5. **element/vocabulary/@displayName**

This attribute specifies the human readable display name that shall be used in the instance.

#### 7.4.5.6. **element/vocabulary/@codeSystemName**

This attribute specifies the human readable code system name that shall be used in the instance.

#### 7.4.5.7. **element/vocabulary/@domain**

This attribute specifies the conceptual domain by name where codes shall be drawn from.

#### Example of a conceptual domain binding

```
<element name="hl7:code" ... datatype="CE">
  <vocabulary domain="TheConceptDomain" />
</element>
```

Please note that specifying a domain typically leads to no testable properties in an instance. A conceptual domain is used for coded elements in templates that are typically further constrained or that leaves a concrete binding to a vocabulary open by intention.

#### 7.4.6. **element/property**

The property sub-element is used for elements of type quantity, string or boolean. It allows the assertion of one or more units, ranges, fraction digits or fixed values.

*Table 21: Property definitions for an element*

Property attribute	Attribute type	Description	Examples
<b>@unit</b>	string	A proper (UCUM) unit.	cm



Property attribute	Attribute type	Description	Examples
	(code)		mm[Hg]
<b>@currency</b>	string (code)	A currency unit.	EUR
<b>@minInclude</b>	integer real	An integer or decimal number to specify the lower inclusive interval boundary	1.00
<b>@maxInclude</b>	integer real	An integer or decimal number to specify the upper inclusive interval boundary	38
<b>@fractionDigits</b>	positive integer	A small positive integer that specifies the number of fractional digits.  A fraction digit "2" requires at least two fraction digits to be present in the instance, otherwise the instance is in error. If fraction digit is suffixed with a "!", e.g. "2!" this means the presence of exactly two fraction digits is required	2 2!
<b>@minLength</b>	positive integer	A positive integer number to specify the minimal length of a string	1
<b>@maxLength</b>	positive integer	A positive integer number to specify the maximal length of a string	10
<b>@value</b>	string	A string as a fixed value.	XYZ

Valid combinations of attributes are:

- @unit @minInclude @maxInclude @fractionDigits (for physical quantities in @value)
- @currency @minInclude @maxInclude @fractionDigits (for monetary amounts in @value)
- @minLength @maxLength (for strings)
- @value (fixed values in @value)

Example of specifying that the value PQ shall be 0..200 and unit shall be 'cm'

```
<element name="hl7:value" ... conformance="R" datatype="PQ">
  <property minInclude="1" maxInclude="200" unit="cm" />
</element>
```

If multiple property elements are specified the definitions are logically connected by an "OR".

---

Example of specifying that the value PQ shall be 1 .. 200 cm or 0.01 .. 2.00 m with exactly 2 fraction digits

```
<element name="hl7:value" ... datatype="PQ">
  <property minInclude="1" maxInclude="200" unit="cm"/>
  <property minInclude="0.01" maxInclude="2.00" unit="m"
    fractionDigits="2!"/>
</element>
```

#### 7.4.7. element/text

This optional sub-element specifies the element content in an instance. This is used for elements of type ST, ED etc.

---

Example of specifying a fixed text for the element content in the instance

```
<element name="hl7:text" ... datatype="ST">
  <text>Fixed text to be used in the instance<text/>
</element>
```

If multiple text elements are specified the definitions are logically connected by an "OR".

#### 7.4.8. Attribute

This definition allows to specify attributes for template elements and their properties.

##### 7.4.8.1. attribute/@name

Specifies the name of the XML attribute. It is typically used in conjunction with a @value.

---

Example: a template element definition with attribute 'classCode' to be valued 'OBS'

```
<element name="hl7:observation" ... >
  <attribute name="classCode" value="OBS"/>
</element>
```

---

Example: attribute 'negationInd' shall be set to 'true'

```
<attribute name="negationInd" value="true"/>
```

##### 7.4.8.2. attribute/@value

Specifies the value of the XML attribute given in @name.

##### 7.4.8.3. attribute/@isOptional

This option attribute determines that the attribute is required to be valued in the instance. Default is 'false', meaning that the attribute is NOT optional, and thus required.

---

Example: Defining @classCode to be 'OBS' if present

```
<attribute name="classCode" value="OBS" isOptional="true"/>
```

#### 7.4.8.4. attribute/@prohibited

Determines that the attribute is prohibited to be in the instance.

Example: A nullFlavor is not allowed

```
<attribute name="nullFlavor" prohibited="true" />
```

#### 7.4.8.5. attribute/@datatype

This optional attribute allows to specify the data type of the XML attribute. Valid types are shown in the table below.

Table 22: Valid data types for an attribute of an element

Attribute data type	Meaning	Attribute Value Example
<b>st</b>	string, the default	"a string"
<b>bl</b>	boolean	"true"
<b>ts</b>	timestamp	"20130630"
<b>int</b>	integer	"1"
<b>real</b>	real	"0.4"
<b>cs</b>	code	"B64"
<b>set_cs</b>	set of codes	"HP WP"

Example

```
<attribute name="mediaType" datatype="cs" />
```

#### 7.4.9. attribute/desc

This optional sub-element carries the multilingual textual description or further explanation on the use of the element.

#### 7.4.10. attribute/item

Indicates the item label of the element for errors, warnings etc., see also section 7.3.15 about Template items.

#### 7.4.11. attribute/vocabulary

This optional sub-element is used for attributes with "coded" data types, i.e. "cs" or "set\_cs". It allows the assertion of one or more value sets.

In validation it is assumed that multiple codes may be used as is possible in e.g. @use on datatypes AD, EN and TEL. In order to constrain to only one possible code, the "@datatype" attribute must be set to "cs" (only one single code allowed). Alternatively, and also the default, use "set\_cs" (one or multiple space delimited codes allowed).

#### 7.4.11.1. attribute/vocabulary/@valueSet

The value set where the code shall be drawn from, referenced by name or id of the value set.

---

Example: required attribute containing one code from a value set

```
<attribute name="mediaType" datatype="cs">
  <vocabulary valueSet="MyMediaTypes" flexibility="dynamic"/>
</attribute>
```

---

Example: optional attribute containing a list of one to many codes from a value set

```
<attribute name="use" datatype="set_cs" isOptional="true">
  <vocabulary valueSet="AddressUse" flexibility="dynamic"/>
</attribute>
```

Another example is how to determine the set of allowed nullFlavors for any element. Assume the following definition of a *doseQuantity* element, then the included attribute definition constraints the set of allowed nullFlavors (defined in a value set shown below as well).

---

Example: null flavors for the *doseQuantity* element must be drawn from value set 'AllowableNulls'

```
<element name="hl7:doseQuantity" minimumMultiplicity="1"
  maximumMultiplicity="1" conformance="R" datatype="IVL_PQ">
  <attribute name="nullFlavor" datatype="cs" isOptional="true">
    <vocabulary valueSet="AllowableNulls"/>
  </attribute>
</element>
```

#### 7.4.12. choice

In some cases a designer wants to offer a choice of template elements and restrict the choice to be n out of m (cardinality constraint). In order to be able to specify choices of elements the choice element can be used.

##### Examples

A typical example is the CDA header author definition where in the underlying model a choice is defined between an *assignedPerson* and an *assignedAuthoringDevice* playing the role of the author and where 0..1 playing entities may be chosen.

This can be expressed by specifying the following constraint

---

```
<choice minimumMultiplicity="0" maximumMultiplicity="1">
  <element name="assignedPerson">...</element>
```

```
<element name="assignedAuthoringDevice">...</element>
</choice>
```

This offers the choice to select 0 or 1 out of the element *assignedPerson* or *assignedAuthoringDevice*. Their definitions are not shown in this case.

#### 7.4.12.1. choice/@minimumMultiplicity

Optional, identifies the minimum number of elements out of the choice.

#### 7.4.12.2. choice/@maximumMultiplicity

Optional, identifies the maximum number of elements out of the choice.

#### 7.4.13. include

A template may reference another defined template by inclusion.

An inclusion within a template design makes use of another template by “virtually” copying the included template definitions, also known as transclusion. In essence this means that template definitions are included by reference and shown as-is on demand, i.e. at time of displaying the template or using it for the creation of validation scripts. Inclusion is automatic and transparent to the user.

#### Examples

A CDA document level template includes the definitions of a CDA *typeId*, *effectiveTime*, *title* and *setId+versionNumber*. These items are defined as “includable” templates because they may be used in other document level templates as well.

#### 7.4.13.1. include/@ref

Mandatory reference to the template to be included, either by name or id.

Example: inclusion of a template named ‘MyEffectiveTime’

```
<include ref="MyEffectiveTime"/>
```

Example: Typical start of a Clinical Document definition with includes

```
<element name="hl7:ClinicalDocument">
  <include ref="CDATypeId"/>
  <element name="hl7:templateId" minimumMultiplicity="1"
    maximumMultiplicity="1" isMandatory="true" datatype="II">
    <desc language="en-US">CDA document template id
      for this kind of document</desc>
    <attribute root="2.16.840.1.113883.3.1937.99.60.3.10.1"/>
  </element>
  <include ref="CDAid"/>
  <element name="hl7:code"
```

```

        minimumMultiplicity="1" maximumMultiplicity="1"
        isMandatory="true" datatype="CE">
        <example>
            <!-- document type -->
            <code code="11524-6" codeSystem="2.16.840.1.113883.6.1"
                codeSystemName="LOINC" displayName="EKG study report"/>
        </example>
        <vocabulary code="11524-6" codeSystem="2.16.840.1.113883.6.1"/>
    </element>
    <include ref="CDAtitle" minimumMultiplicity="0"
        maximumMultiplicity="1">
        <example>
            <title>EKG Report as of 1 February 2013</title>
        </example>
    </include>
    <include ref="CDAeffectiveTime"/>
    <include ref="CDAconfidentialityCode"/>
    <include ref="CDAreCORDTarget"
        minimumMultiplicity="1" maximumMultiplicity="1"/>
    <include ref="CDAauthor"
        minimumMultiplicity="1" maximumMultiplicity="1"/>
    <include ref="CDAcustodian"/>
    ...
</element>

```

#### 7.4.13.2. include/@minimumMultiplicity

The optional attribute identifies the minimum number of repetitions of all elements at top level of the included template.

The include statement may specify overriding cardinalities/conformances. If the included template has only one root element the cardinality/conformance of this element gets overridden by the cardinality/conformance specified in the calling include statement. If there are more than one root element in the included template, all element's cardinalities/conformances are overridden.

#### Examples

In the following situation a *templateX* is included. Please note that regardless of the cardinality/conformance definitions of the root element *hl7:code* in *templateX*, it gets overridden by 0..1 as specified in the include statement. Also note that this is an override and not a formally checked restriction.

```
<template name="someTemplate" ... >
```

```

<element name="hl7:id" ... />

<include ref="templateX"
    minimumMultiplicity="0" maximumMultiplicity="1">
    ...
</template>

<template name="templateX" ... >
    <element name="hl7:code"
        minimumMultiplicity="1" maximumMultiplicity="*" ... />
</template>

```

#### 7.4.13.3. **include/@maximumMultiplicity**

The optional attribute identifies the maximum number of repetitions of all elements at top level of the included template.

#### 7.4.13.4. **include/@isMandatory**

This optional boolean attribute identifies that all elements at top level of the included template are mandatory, i.e. in an instance it shall be populated with a valid value.

#### 7.4.13.5. **include/@conformance**

This optional attribute identifies the conformance of all elements at top level of the included template.

#### 7.4.13.6. **include/desc**

This optional sub-element carries the multilingual textual description or further explanation on the use of the include.

#### 7.4.13.7. **include/item**

Indicates the item label of the include statement for errors, warnings etc., see also section 7.3.15 about Template items.

#### 7.4.13.8. **include/example**

An include may have zero or more examples or example fragments for the element, see also section 7.3.15.3 about Template examples.

### 7.4.14. **Schematron statements**

To support at least one typical constraint language schematron rules (assert, report, let) are allowed in template definitions. They may be interspersed at defined locations.

#### 7.4.14.1. **assert**

This statement will be converted to an appropriate schematron assert statement. They are often used to express co-constraints.

---

### Example

---

```
<assert role="error" test="../hl7:observation/@negationInd or hl7:value">
  If the observation is not null a value shall be present
</assert>
```

The assert statement is identically defined to the original schematron assert.

#### **assert/@flag**

Schematron flag to convey state or severity information to a subsequent process.

#### **assert/@see**

URI value of the external information of interest.

#### **assert/@role**

One of the following: “error”, “warning” or “information”.

#### **assert/@test**

Schematron test expression.

#### **7.4.14.2. report**

This statement will be converted to an appropriate schematron report statement. They are often used to express co-constraints.

The report statement is identically defined to the original schematron report.

#### **report/@flag**

Schematron flag to convey state or severity information to a subsequent process.

#### **report/@see**

URI value of the external information of interest.

#### **report/@role**

One of the following: “error”, “warning” or “information”.

#### **report/@test**

Schematron test expression.

#### **7.4.14.3. let**

A let statement defines a schematron runtime variable that can be used in assert or report or let statements.

---

Example: get the @code in the value element of the observation classified by code/@code 123-4

```
<let name="lepcode"
  value="//hl7:observation[hl7:code[@code='123-4']]/hl7:value/@code"/>
```

#### **let/@name**

Schematron let name of the variable



**let/@value**

Schematron let value

## 7.5. How to Create, Validate and Consume Template Definitions

The following section describes actions on Template Definition Instances not the XML instances of e.g. a Clinical Document.

### 7.5.1. Creation

There are multiple ways to create a Template Definition Instance.

The simplest way is to craft the definitions using an XML editor. To support this method, a W3C schema and accompanying schematron rules are available (see following section).

Meanwhile template authoring tools are available, e.g. Model Driven Health Tool MDHT, Lantana's Trifolia Workbench, ART-DECOR etc. The underlying rationale and the details of how template design is acquired, presented and stored may be different in these tools, but it is expected that – once this specification has been endorsed – an exchange of template definitions between these tools is easily possible.

#### *Reference Implementation*

*The Template Definition Exchange Standard (ITS) described here has successfully been implemented in the ART-DECOR (see [art-decor.org](http://art-decor.org)) tool suite. A template editor is available to create new and change existing templates and maintain their meta data and design. An exchange mechanism with template repository characteristics is also implemented.*

### 7.5.2. Validation

For the validation of the template definitions itself a W3C schema with embedded schematron is available.

A Wiki page has been established in order to point to all recommended artifacts, see

[http://wiki.hl7.org/index.php?title=Template\\_Instance\\_Validation](http://wiki.hl7.org/index.php?title=Template_Instance_Validation)

### 7.5.3. Consume

Regarding consuming template definitions several areas have to be mentioned:

- Documentation and publication of template definitions
- Using template definitions for XML instance validation including testing and certification purposes
- Creation of XML instance (fragments) according to the template definition, e.g. code generation for interfaces of applications
- Exposing template definitions in appropriate Template Repositories.

It is obvious that the template design has to be presented in an appropriate format for through the tools they were designed with. In addition, for **documentation and publication** purposes textual representations like in implementations guides may be required. Some governance groups use wiki pages to publish their template designs, in

other cases extracts of the definitions are done by generating documents for text processors or HTML pages.

As of today the most prominent purpose of templates beside documentation and publication in various formats, is XML **instance validation**.

In order to be able to validate instances against a template, a transformation exists from the template exchange format to schematrons directly out of the template definitions. The underlying schematron engine creates ISO schematron rules and also compiles all value set references that are needed for a proper validation.

In some projects the template definitions have also been used to create XML instance fragments by means of XSLT transformations or to generate code out of the definitions.

#### *Reference Implementation*

*At least the ART-DECOR tool suite has the ability to expose template definitions in a repository so that templates can be referred to or re-used in other template definitions, even between different governance groups. There are also several repository functions implemented regarding exchange.*

## 7.6. How to exchange Template Definitions between Systems

One major objective of this ITS is to enable the exchange of template definitions between systems, either in terms of an export and import functionality or to expose template definitions in a Template Repository.

There are some typical functions, a Template Repository must provide. The following list is an example.

### Examples of Template Repository Functions regarding Exchange of Template Definitions

#### Typical Functions

- Provide a **list of all templates** available in the Repository
- **Search** for a specific Template based on Meta Data information
- **Retrieve** a complete Template in the Exchange Format

#### Additional Functions

- **List Template Meta Data**
- **Allocate Template Id** for a new Template
- **Submit Template** to the Repository
- **Update Template** in the Repository
- **Remove Template** from the Repository

## 7.7. Best practice examples for CDA definitions

The following examples are derived from real projects using this ITS and show best practice for simple CDA template definitions. This document starts with simple section level template definitions, continues with entry and header level templates and ends with a whole document level template specification. If any examples are interspersed in

the definition they are shown in italic font to make it easier to identify these parts as examples.

#### 7.7.1. Example of a CDA section level template

A section level template “EKG Impression Section” is defined to capture the narrative results of an EKG study. The definition includes the corresponding *templateId*, the *section.code* as a LOINC code and declares both *section.title* and *section.text* as mandatory.

---

##### Example: EKG Impression Section

```
<template id="2.16.840.1.113883.3.1937.99.61.3.10.3001"
  name="EKGImpressionSection" displayName="EKG Impression Section"
  effectiveDate="2013-02-10T00:00:00" statusCode="active">

  <desc language="en-US">This section describes the impression
    (findings) of an EKG study of a patient</desc>

  <classification type="cdasectionlevel"/>

  <context id="**"/>

  <example>

    <section classCode="DOCSECT">

      <!-- Template id for EKG measurements -->

      <templateId root="2.16.840.1.113883.3.1937.99.61.3.10.3001"/>

      <code code="18844-1" codeSystem="2.16.840.1.113883.6.1"
        codeSystemName="LOINC"/>

      <title>Impression</title>

      <text>Normal sinus rhythm<br/> Ischemic ST-T changes in
        anterior leads<br/> Poor R Progression in right
        precordial leads</text>

    </section>

  </example>

  <element name="hl7:section">

    <attribute classCode="DOCSECT" isOptional="true"/>

    <!-- Element templateId -->

    <element name="hl7:templateId" minimumMultiplicity="1"
      maximumMultiplicity="1" datatype="II">

      <attribute root="2.16.840.1.113883.3.1937.99.61.3.10.3001"/>

    </element>

    <!-- Element code -->

    <element name="hl7:code" minimumMultiplicity="1"
      maximumMultiplicity="1" isMandatory="true" datatype="CD">

      <vocabulary code="18844-1"
        codeSystem="2.16.840.1.113883.6.1"/>

    </element>

    <!-- Element title -->

    <element name="hl7:title" minimumMultiplicity="1"
```

```

        maximumMultiplicity="1" isMandatory="true" datatype="ST"/>
    <!-- Element text -->
    <element name="hl7:text" minimumMultiplicity="1"
        maximumMultiplicity="1" isMandatory="true" datatype="SD.TEXT"/>
</element>
</template>

```

A possible rendition of the definition could look like the following (drawn from the ART-DECOR reference implementation). This is only one of the possible visualizations.

EKG Impression Section

Version

2013-12-05

Version Label

Name

EKGImpressionSection

Status

Active

Id

2.16.840.1.113883.3.1937.99.61.3.10.3001

Display Name

EKG Impression Section

Description, model, context and examples

Description

This section describes the impression (findings) of an EKG study of a patient

Context

Parent nodes of template element with id 2.16.840.1.113883.3.1937.99.61.3.10.3001

Classification

cdasectionlevel

Used by / Uses

Used by Template Id

As

Name

as of

2.16.840.1.113883.3.1937.99.61.3.10.1

Containment

MinimalCDAAdocument

Minimal CDA document

2013-12-05

Example

```
<section classCode="DOCSECT">
  <!-- Template id for EKG measurements -->
  <templateId root="2.16.840.1.113883.3.1937.99.61.3.10.3001"/>
  <!-- Section code -->
  <code code="18844-1" codeSystem="2.16.840.1.113883.6.1" codeSystemName="LOINC" />
  <title>Impression</title>
  <text>
    Normal sinus rhythm<br/>Ischemic ST-T changes in anterior leads<br/>Poor R Progression in right precordial leads
  </text>
</section>
```

Item

DT

Card

Conf

Description

Label

<> h17:section

EKGImpressionSection

@ classCode

0..1

F

DOCSECT

EKGImpressionSection

<> h17:templateId

II

1..1

EKGImpressionSection

@ root

1..1

F

2.16.840.1.113883.3.1937.99.61.3.10.3001

EKGImpressionSection

<> h17:code

CD

1..1

M

EKGImpressionSection

CONF

@code = 18844-1

@codeSystem = 2.16.840.1.113883.6.1

<> h17:title

ST

1..1

M

EKGImpressionSection

<> h17:text

SD.TEXT

1..1

M

EKGImpressionSection

If a section contains an entry this is typically expressed as the following.

Example fragment: the cda entry contains a template named AgeObservation

```
<element name="cda:entry" contains="AgeObservation" ...
```

It is also possible to state static or dynamic binding for the template that is referred to by @contains.

Example fragment: the cda entry contains a template with id 1.2.3.4.5 as of 2013-11-23 (the version of that template)

```
<element name="cda:entry" contains="1.2.3.4.5" flexibility="2013-11-23"...
```

The template itself has to start as the example for an entry level template below shows.

### 7.7.2. Example of a CDA header level template

A header level template “CDA custodian” is defined to capture the custodian of a document. It is defined as a re-usable component that is normally used by inclusion in a document level template (see below). Therefore it has no “context” as the context is determined by the point of inclusion in the including template.

The items defined for the custodian in this example includes an *assignedCustodian.representedCustodianOrganization.id* only.

---

#### Example: CDA custodian

```
<template id="2.16.840.1.113883.3.1937.99.61.3.10.2003"
  name="CDAcustodian" displayName="CDA custodian"
  effectiveDate="2013-12-05T00:00:00" statusCode="active">

  <desc language="en-US">Custodian of the document</desc>

  <classification type="cdaheaderlevel"/>

  <example>
    <custodian>
      <assignedCustodian>
        <representedCustodianOrganization>
          <id root="2.16.840.1.113883.3.1937.99.3.2.997788"/>
        </representedCustodianOrganization>
      </assignedCustodian>
    </custodian>
  </example>

  <element name="hl7:custodian" minimumMultiplicity="1"
    maximumMultiplicity="1" isMandatory="true">
    <attribute typeCode="CST" isOptional="true"/>
    <element name="hl7:assignedCustodian" minimumMultiplicity="1"
      maximumMultiplicity="1" isMandatory="true">
      <attribute classCode="ASSIGNED" isOptional="true"/>
      <element name="hl7:representedCustodianOrganization"
        minimumMultiplicity="0" maximumMultiplicity="1">
        <element name="hl7:id" minimumMultiplicity="1"
          maximumMultiplicity="1" conformance="R" datatype="II"/>
      </element>
    </element>
  </element>

</template>
```

A possible rendition of the definition could look like the following (drawn from the ART-DECOR reference implementation). This is only one of the possible visualizations.

CDA custodian				
Version	2013-12-05	Status	Active	
Version Label		Id	2.16.840.1.113883.3.1937.99.61.3.10.2003	
Name	CDAcustodian	Display Name	CDA custodian	
Description, model, context and examples				
Description	Custodian of the document			
Classification	cdaheaderlevel			
Used by / Uses	Used by Template Id	As	Name	as of
	2.16.840.1.113883.3.1937.99.61.3.10.1	Inclusion	MinimalCDAdocument Minimal CDA document	2013-12-05
Relations	Type	Model / Template	as of	
	Version 2.16.840.1.113883.3.1937.99.61.5.10.2003	CDAcustodian CDA custodian	2013-10-10	
Example	<pre> &lt;custodian&gt;   &lt;assignedCustodian&gt;     &lt;representedCustodianOrganization&gt;       &lt;id root="2.16.840.1.113883.3.1937.99.3.2.997788"/&gt;     &lt;/representedCustodianOrganization&gt;   &lt;/assignedCustodian&gt; &lt;/custodian&gt; </pre>			

Item	DT	Card	Conf	Description	Label
<> hl7:custodian		1..1	M		CDAcustodian
@ typeCode		0..1	F	CST	CDAcustodian
<> hl7:assignedCustodian		1..1	M		CDAcustodian
@ classCode		0..1	F	ASSIGNED	CDAcustodian
<> hl7:representedCustodianOrganization		0..1	O		CDAcustodian
hl7:id	II	1..1	R		CDAcustodian

### 7.7.3. Example of a CDA entry level template

An entry level template “Age Observation” is defined to capture the age of a patient. The only variable spot in the template is the actual value of the observation and the @unit is also bound to the value set *AgePQ\_UCUM* that contains typical units for age like “a” for year or “mo” for month.

#### Example: Age Observation

```

<template id="2.16.840.1.113883.3.1937.99.61.3.10.4001"
  name="AgeObservation" effectiveDate="2013-01-31T00:00:00"
  statusCode="draft" displayName="Age Observation">

  <desc language="en-US">This Age Observation represents...</desc>
  <classification type="cdaentrylevel"/>
  <context id="**"/>
  <example>
    <observation classCode="OBS" moodCode="EVN">
      <templateId root="2.16.840.1.113883.10.20.22.4.31"/>
      <code code="397659008" codeSystem="2.16.840.1.113883.6.96"
        displayName="Age"/>
      <statusCode code="completed"/>
      <value xsi:type="PQ" value="57" unit="a"/>
    </observation>
  </example>

```

```

<element name="hl7:observation">
  <attribute name="classCode" value="OBS"/>
  <attribute name="moodCode" value="EVN"/>
  <element name="hl7:templateId" minimumMultiplicity="1"
    maximumMultiplicity="1" isMandatory="true">
    <attribute name="root"
      value="2.16.840.1.113883.10.20.22.4.31"/>
  </element>
  <element name="hl7:code" minimumMultiplicity="1"
    maximumMultiplicity="1" isMandatory="true">
    <vocabulary code="445518008"
      codeSystem="2.16.840.1.113883.6.96"/>
  </element>
  <element name="hl7:statusCode" minimumMultiplicity="1"
    maximumMultiplicity="1" isMandatory="true">
    <vocabulary code="completed"/>
  </element>
  <element name="hl7:value" minimumMultiplicity="1"
    maximumMultiplicity="1" isMandatory="true" datatype="PQ">
    <attribute name="unit">
      <vocabulary valueSet="Age_PQ_UCUM"/>
    </attribute>
  </element>
</element>
</template>

```

A possible rendition of the definition could look like the following (drawn from the ART-DECOR reference implementation). This is only one of the possible visualizations.

Age Observation

Version

2013-01-31

Status

Draft

Version Label

Id

2.16.840.1.113883.3.1937.99.61.3.10.4001

Name

AgeObservation

Display Name

Age Observation

Description, model, context and examples

Description

This Age Observation represents the subject's age at onset of an event or observation. The age of a relative in a Family History Observation at the time of that observation could also be inferred by comparing RelatedSubject/subject/birthTime with Observation/effectiveTime. However, a common scenario is that a patient will know the age of a relative when the relative had a certain condition or when the relative died, but will not know the actual year (e.g., "grandpa died of a heart attack at the age of 50"). Often times, neither precise dates nor ages are known (e.g. "cousin died of congenital heart disease as an infant").

Context

Parent nodes of template element with id 2.16.840.1.113883.3.1937.99.61.3.10.4001

Classification

cdaentrylevel

Example

```
<observation classCode="OBS" moodCode="EVN">
  <templateId root="2.16.840.1.113883.10.20.22.4.31"/>
  <code code="397659008" codeSystem="2.16.840.1.113883.6.96" displayName="Age"/>
  <statusCode code="completed"/>
  <value xsi:type="PQ" value="57" unit="a" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
</observation>
```

Item	DT	Card	Conf	Description	Label
h17:observation					AgeObservation
@ classCode		1..1	F	OBS	AgeObservation
@ moodCode		1..1	F	EVN	AgeObservation
h17:templateId		1..1	M		AgeObservation
@ root		1..1	F	2.16.840.1.113883.10.20.22.4.31	AgeObservation
h17:code		1..1	M		AgeObservation
			CONF	@code = 445518008 @codeSystem = 2.16.840.1.113883.6.96	
h17:statusCode		1..1	M		AgeObservation
			CONF	@code = completed	
h17:value	PQ	1..1	M		AgeObservation
@ unit		1..1			AgeObservation
			CONF	Code from value set : AgePQ_UCUM (dynamic)	

#### 7.7.4. Example of a CDA document level template

A document level template "Minimal CDA" is defined. It defines the minimal requirements of a CDA document. A *CDATypeId* template is included (re-usable template for all kinds of document level templates), along with a template id, document id, document type code, title, effective time and confidentiality code. Subsequently the record target, author and custodian definitions are included, followed by the structured body that contains a single section template.

##### Example: Document Level Template of a minimal CDA document

```
<template id="2.16.840.1.113883.3.1937.99.61.3.10.1"
  name="MinimalCDAdocument" displayName="Minimal CDA document"
  effectiveDate="2013-12-05T00:00:00" statusCode="active">
  <desc language="en-US">A minimal CDA Release 2 document, ...</desc>
  <classification type="cdadocumentlevel"/>
  <context path="/" />
  <example>
    <ClinicalDocument xmlns="urn:hl7-org:v3">
```



```

    <!-- CDA Header -->
    <typeId root="2.16.840.1.113883.1.3"
      extension="POCD_HD000040"/>
    <templateId root="2.16.840.1.113883.3.1937.99.61.3.10.1"/>
    <id extension="123456789"
      root="2.16.840.1.113883.3.1937.99.3.2.997788.1"/>
    <code code="11524-6" codeSystem="2.16.840.1.113883.6.1"
      codeSystemName="LOINC" displayName="EKG study report"/>
    <effectiveTime value="20131020122709"/>
    <confidentialityCode code="N"
      codeSystem="2.16.840.1.113883.5.25"/>
    <recordTarget><!-- .. --></recordTarget>
    <author><!-- .. --></author>
    <custodian><!-- .. --></custodian>
    <!-- CDA Body -->
    <component>
      <structuredBody>
        <component><!-- .. --></component>
      </structuredBody>
    </component>
  </ClinicalDocument>
</example>
<element name="hl7:ClinicalDocument">
  <include ref="CDATypeId"/>
  <!-- template id for this document -->
  <element name="hl7:templateId" minimumMultiplicity="1"
    maximumMultiplicity="1" isMandatory="true" datatype="II">
    <desc language="en-US">CDA document template id for
      this kind of document</desc>
    <attribute root="2.16.840.1.113883.3.1937.99.61.3.10.1"/>
  </element>
  <!-- document id -->
  <element name="hl7:id" minimumMultiplicity="1"
    maximumMultiplicity="1" isMandatory="true" datatype="II"/>
  <!-- document type code -->
  <element name="hl7:code" minimumMultiplicity="1"
    maximumMultiplicity="1" isMandatory="true" datatype="CE">
    <example>
      <code code="11524-6" codeSystem="2.16.840.1.113883.6.1"
        codeSystemName="LOINC" displayName="EKG study report"/>
    </example>
    <vocabulary code="11524-6"

```

```

        codeSystem="2.16.840.1.113883.6.1"/>
    </element>
    <include ref="CDAtitle" minimumMultiplicity="0"
        maximumMultiplicity="1">
        <example>
            <title>EKG Report as of 1 February 2013</title>
        </example>
    </include>
    <element name="hl7:effectiveTime" minimumMultiplicity="1"
        maximumMultiplicity="1" isMandatory="true"
        datatype="TS.DATETIME.MIN"/>
    <element name="hl7:confidentialityCode" minimumMultiplicity="1"
        maximumMultiplicity="1" isMandatory="true" datatype="CE">
        <example>
            <confidentialityCode code="N"
                codeSystem="2.16.840.1.113883.5.25"/>
        </example>
        <vocabulary valueSet="BasicConfidentialityKind"/>
    </element>
    <include ref="CDArecordTarget"/>
    <include ref="CDAauthor"/>
    <include ref="CDAcustodian"/>
    <element name="hl7:component">
        <attribute typeCode="COMP" contextConductionInd="true"
            isOptional="true"/>
        <element name="hl7:structuredBody">
            <attribute classCode="DOCBODY" moodCode="EVN"
                isOptional="true"/>
            <!-- EKG Impression section, reuquired
                (otherwise this report is useless) -->
            <element name="hl7:component" minimumMultiplicity="1"
                maximumMultiplicity="1" isMandatory="true"
                contains="EKGImpressionSection">
                <attribute typeCode="COMP"
                    contextConductionInd="true" isOptional="true"/>
            </element>
        </element>
    </element>
</element>
</template>

```



## Appendix A – Glossary

Term	Definition
<b>Template</b>	A template is a set of conformance statements which further constrain an existing information model.
<b>Template Governance Group</b>	A template governance group establishes the rules and best practices associated with their creation and management of templates.
<b>Template Design</b>	A template design can be defined in a number of ways. They can be created as human-written specifications or they can be developed using modeling / design tools which automate the process for greater precision and consistency. Templates can be published in human-readable or in machine processable forms. They can be managed in documents where they are defined or in repositories deployed through technology tools developed specifically for the management of templates.
<b>Template Metadata</b>	Version-related meta data that exists for each design of a template. Ex: Identifier, Name, Effective Date, Version Label, Expiration Date, Official Release Date, Status, and Additional meta data.
<b>Template Body</b>	A collection of constraints that ideally describes the structure and semantics of all instance elements.
<b>Template Registry</b>	A template registry tracks the existence of templates and their designs from one or more template repositories.
<b>Template Types</b>	Document level, Header level, Section level, Entry level, etc.
<b>Open Template</b>	An open template defines what's relevant for that template while allowing undefined contents in the instance.
<b>Closed Template</b>	A closed template defines what's relevant for that template and leaves no room for any other contents in the instance.
<b>Cardinalities</b>	The cardinality indicator (0..1, 1..1, 1..*, etc.) specifies the allowable occurrences within a document instance.
<b>Conformance Verbs</b>	SHALL, SHALL NOT, SHOULD, SHOULD NOT, MAY, MAY NOT
<b>Vocabulary Binding</b>	<p>There are three practically used types of vocabulary bindings.</p> <ul style="list-style-type: none"> <li>• A coded element can be bound to a specific code from a specific code system and the population of the element can be considered as a “constant”.</li> <li>• A coded element can be bound to a specific value set by value set id or name.</li> <li>• A coded element can be bound to a certain concept domain by domain id or name.</li> </ul>

<b>Template Inclusion</b>	"Inheritance" of the conformances defined in another template
<b>Template Containment</b>	Inclusion in a template definition of a component defined by another template.
<b>Static Binding</b>	A STATIC binding is a fixed binding at design time.
<b>Dynamic Binding</b>	A DYNAMIC binding implies a need for a look-up of the (most recent) artifact at runtime.
<b>Template Repository</b>	A template repository houses the definition of template designs.
<b>Template Registry</b>	When template designs include interdependencies, it becomes necessary to stay informed of changes in the related templates. Template Registries may support functionality which permits a system to declare its interest in a template or set of templates so that it may be informed when changes occur. Once a system has subscribed, the Template Registry can publish a change, and the needed notification happens automatically.
<b>Template Designer</b>	a person who develops the set of conformance statements which make up a template.
<b>Content Creator</b>	Content Creators use templates to assert the patterns and semantics present in the data instances they create.
<b>Content Consumer</b>	Content Consumers use templates to receive the patterns and semantics presented to them by Content Creators.
<b>Content Validator</b>	An assertion that confirms that the document does in fact meet all of the constraint requirements specified for the template.

## Appendix B – Acronyms and Abbreviations

CDA	Clinical Document Architecture
DAM	Domain Analysis Model
DSTU	Draft Standard for Trial Use
H&P	History and Physical
HIT	healthcare information technology
HL7	Health Level Seven
HTML	Hypertext Markup Language
IG	implementation guide
IHE	Integrating the Healthcare Enterprise
IHTSDO	International Health Terminology Standard Development Organisation
LOINC	Logical Observation Identifiers Names and Codes
MDHT	Model-Driven Health Tools
PDF	portable document format
RIM	Reference Information Model
SDWG	Structured Documents Working Group
SDO	Standards Development Organization
SNOMED CT	Systemized Nomenclature for Medicine – Clinical Terms
UCUM	Unified Code for Units of Measure
URL	Uniform Resource Locator
XPath	XML Path Language

## Appendix C – Conventions and Best Practice for Template Documentation

The pages that follow show how the information contained in template definitions is used differently by different governance groups. While the underlying metadata about and constraint information contained within the template may be the same, each governance group may decide to apply it differently.

For interoperability, best practice suggests that differences in the interpretation of the standard “model-native” template definitions should be minimized. Differences in the way a governance group chooses to render the information contained in the template may also exist. However, these differences may suit the reading preferences of the governance group’s constituency and these differences do not cause interoperability problems, as long as the native language representations do not alter the technical meaning of the conformance statement.

As examples, the template definitions of MDHT, IHE and Trifolia and ART-DECOR are shown and recommendations and best practice for template documentation are given.

### Meta data

A template documentation shall contain most of the meta data of the template, i.e.

- Name
- Version indication
- Id
- Open/closed
- Containments and Inclusions
- Description and Purpose
- Context
- Status

Furthermore the assignments of OIDs to template artifacts shall follow good OID assignment practice, for example that an OID should not be a node (structure for the management of OIDs) and a leaf (for an instance of an artifact) at the same time, see also ISO/DTS 13582: Health Informatics — Sharing of OID registry information [isoid].

### 3.94 Result Organizer (V2)

[organizer: templateId 2.16.840.1.113883.10.20.22.4.1.2 (open)]


**Table 369: Result Organizer (V2) Contexts**

Contained By:	Contains:
<a href="#">Health Concern Act (NEW)</a> (optional) <a href="#">Results Section (entries optional) (V2)</a> (optional) <a href="#">Results Section (entries required) (V2)</a> (required)	<a href="#">Author Participation (NEW)</a> <a href="#">Result Observation (V2)</a>

This template provides a mechanism for grouping result observations. It contains information applicable to all of the contained result observations. The Result Organizer code categorizes the contained results into one of several commonly accepted values (e.g., "Hematology", "Chemistry", "Nuclear Medicine").

If any Result Observation within the organizer has a statusCode of 'active', the Result Organizer must also have as statusCode of 'active'.

*Figure 7: C-CDA example for template meta data*

Allergy Problem Act					
Version	2013-01-31		Status	● In Review	
Version Label			Id	2.16.840.1.113883.10.20.22.4.30	
Name	AllergyProblemAct		Display Name	Allergy Problem Act	
Description, model, context and examples					
Description	This clinical statement act represents a concern relating to a patient's allergies or adverse events. A concern is a term used when referring to patient's problems that are related to one another. Observations of problems or other clinical statements captured at a point in time are wrapped in a Allergy Problem Act, or "Concern" act, which represents the ongoing process tracked over time. This outer Allergy Problem Act (representing the "Concern") can contain nested problem observations or other nested clinical statements relevant to the allergy concern.				
Context	Parent nodes of template element with id 2.16.840.1.113883.10.20.22.4.30				
Classification	cdaentrylevel				
Used by / Uses	Used by Template Id	As	Name		as of
	2.16.840.1.113883.10.20.22.2.6.1	Containment	AllergiesSectionentriesrequired Allergies Section (entries required)		2013-01-31
	2.16.840.1.113883.10.20.22.2.6	Containment	AllergiesSectionentriesoptional Allergies Section (entries optional)		2013-01-31
	Uses Template Id	As	Name		as of
	2.16.840.1.113883.10.20.22.4.7	Containment	Allergy-IntoleranceObservation Allergy - Intolerance Observation		2013-01-31

*Figure 8: ART-DECOR example for template meta data*

## Tabular View

A template shall contain a tabular view on the definitions, mentioning

- Hierarchical view on elements and attributes
- Cardinality and Conformance indications
- Data types
- Description or link to clinical concepts
- Vocabulary bindings, static or dynamic



- Containment / Inclusions, static or dynamic
- Labels / Conformance Numbers to be able to reference a statement, e.g. in a validation process
- Element level examples

**Table 370: Result Organizer (V2) Constraints Overview**

XPath	Card.	Verb	Data Type	CONF#	Fixed Value
organizer[templateId/@root = '2.16.840.1.113883.10.20.22.4.1.2']					
@classCode	1..1	SHALL		<a href="#">7121</a>	2.16.840.1.113883.5.6 (HL7ActClass)
@moodCode	1..1	SHALL		<a href="#">7122</a>	2.16.840.1.113883.5.1001 (ActMood) = EVN
templateId	1..1	SHALL		<a href="#">7126</a>	
@root	1..1	SHALL		<a href="#">9134</a>	2.16.840.1.113883.10.20.22.4.1.2
id	1..*	SHALL		<a href="#">7127</a>	
code	1..1	SHALL		<a href="#">7128</a>	
statusCode	1..1	SHALL		<a href="#">7123</a>	
@code	1..1	SHALL		<a href="#">14848</a>	2.16.840.1.113883.11.20.9.39 (Result Status)
effectiveTime	0..1	MAY		<a href="#">31865</a>	

*Figure 9: C-CDA example for tabular view on a template design body*

Item	DT	Card	Conf	Description	Label
↳ cda:act					AllergyProblemAct
↳ @classCode		1..1	F	ACT	conf-7469
↳ @moodCode		1..1	F	EVN	conf-7470
↳ cda:templateId		1..1	M		conf-7471
↳ @root		1..1	F	2.16.840.1.113883.10.20.22.4.30	conf-10489
↳ cda:id		1..*	M		conf-7472
↳ cda:code		1..1	M		conf-7477
	CONF			@code = 48765-2 @codeSystem = 2.16.840.1.113883.6.1	
↳ cda:statusCode		1..1	M		conf-7485
	CONF			Code from value set: 2.16.840.1.113883.11.20.9.19 (dynamic)	
↳ cda:effectiveTime		1..1	M		conf-7498
↳ cda:entryRelationship		1..*	M	contains : 2.16.840.1.113883.10.20.22.4.7 Allergy - Intolerance Observation (dynamic)	conf-7509
↳ @typeCode		1..1	F	SUBJ	conf-7915

*Figure 10: ART-DECOR example for tabular view on a template design body*

## Examples / Example Fragments

From implementation practice it is very obvious that templates should contain one or more XML instance example fragments. The optimum is to cover typical situations with the examples, e.g. normal, negating or unknown events. The examples should be correct or should be explicitly marked incorrect if they are in error for demonstration purposes.

### Early Hearing Screening Left Result example

```
<?xml version="1.0" encoding="UTF-8"?>
<observation xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="urn:hl7-org:v3" xsi:schemaLocation="urn:hl7-org:v3 CDA.xsd">
  <templateId root="1.3.6.1.4.1.19376.1.7.3.1.1.15.6.9"/>
  <id root="MDHT" extension="1552271554"/>
  <code code="54108-6" codeSystem="2.16.840.1.113883.6.1"
codeSystemName="LOINC" displayName="Newborn Hearing Screening Left Result"/>
  <effectiveTime>
    <low value="2013"/>
    <high value="2013"/>
  </effectiveTime>
  <value xsi:type="CD" code="22730789"/>
  <methodCode codeSystem="2.16.840.1.113883.6.1" codeSystemName="LOINC"/>
</observation>
```

Figure 11: MDHT example for an XML instance fragment example

## Conformance Conventions

During the last years various governance groups used different conventions to express conformance. Conformance verbs like “shall”, “should” and “may” etc. are in use as well as conformance statements like “mandatory”, “required” and “optional”. Expressions of “conditional” are even more different.

Table 317: Consolidated Conformance Verb Matrix

RFC 2119	HL7	IHE	HITS	Workgroup Consensus
<b>SHALL</b> Absolute requirement of the specification	<b>SHALL</b> Required/Mandatory	R (Required) Element must be present but can be NULL	R (Required) Data elements must always be sent. A NULL can be sent.	<b>SHALL</b> Element must be present but can be NULL  Where necessary to explicitly preclude nullFlavor (e.g. where you want to preclude nullFlavor on observation/value), can include something like "SHALL NOT include nullFlavor".  Where <b>SHALL</b> is applied to an attribute, it must be present and cannot be a NULL
<b>SHALL NOT</b> Absolute prohibition of the specification	<b>SHALL NOT</b> Not Required/Mandatory	-	-	<b>SHALL NOT</b> Absolute prohibition against inclusion

Figure 12: Comparison of different use of conformance verbs in Consolidated CDA [ccda1]

This specification aligns all aspects into a single view for the template definitions, please refer to section 2.10.1.4. A governance group may decide to offer alternative ways of presenting conformance.

## Verbose Constraint Statements

Some implementation guides offer a natural (verbose) conformance statement representation, along with the tabular view of instead.

This representation is not standardized (and also out-of-scope this specification). A governance groups shall offer clear guidance on how to interpret the constraint language. From practice this can be a difficult task and good explanation of use should be considered carefully.

### Early Hearing Screening Left Result

[Observation: templateId 1.3.6.1.4.1.19376.1.7.3.1.1.15.6.9]

1. **SHALL** contain exactly one [1..1] **code**/**@code**="54108-6" *Newborn Hearing Screening Left Result* (CodeSystem: 2.16.840.1.113883.6.1 LOINC)
2. **SHALL** contain exactly one [1..1] **methodCode**, where the **@code** **SHALL** be selected from ValueSet *NewbornHearingScreeningResultsMethodCodes* 1.3.6.1.4.1.19376.1.7.3.1.1.15.7.6 **DYNAMIC**
3. **SHALL** contain exactly one [1..1] **value** with data type CD, where the **@code** **SHALL** be selected from ValueSet *NewbornHearingScreeningResultValueCodes* 1.3.6.1.4.1.19376.1.7.3.1.1.15.7.5 **DYNAMIC**
4. **SHOULD** contain zero or one [0..1] **entryRelationship**
  - a. Contains **@typeCode**="RSON" *RSON*
  - b. Contains exactly one [1..1] *Early Hearing Screening Reason No Result* (templateId: 1.3.6.1.4.1.19376.1.7.3.1.1.15.6.5)

Figure 13: MDHT conformance statement representation example of a template design body

1. **SHALL** contain exactly one [1..1] **@classCode** (CodeSystem: HL7ActClass 2.16.840.1.113883.5.6 **STATIC**) (CONF:7121).
2. **SHALL** contain exactly one [1..1] **@moodCode**="EVN" Event (CodeSystem: ActMood 2.16.840.1.113883.5.1001 **STATIC**) (CONF:7122).
3. **SHALL** contain exactly one [1..1] **templateId** (CONF:7126) such that it
  - a. **SHALL** contain exactly one [1..1] **@root**="2.16.840.1.113883.10.20.22.4.1.2" (CONF:9134).
4. **SHALL** contain at least one [1..\*] **id** (CONF:7127).
5. **SHALL** contain exactly one [1..1] **code** (CONF:7128).
  - a. SHOULD be selected from LOINC (codeSystem 2.16.840.1.113883.6.1) or SNOMED CT (codeSystem 2.16.840.1.113883.6.96), and MAY be selected from CPT-4 (codeSystem 2.16.840.1.113883.6.12) (CONF:19218).
  - b. Laboratory results SHOULD be from LOINC (CodeSystem: 2.16.840.1.113883.6.1) or other constrained terminology named by the US Department of Health and Human Services Office of National Coordinator or other federal agency (CONF:19219).
6. **SHALL** contain exactly one [1..1] **statusCode** (CONF:7123).
  - a. This statusCode **SHALL** contain exactly one [1..1] **@code**, which **SHALL** be selected from ValueSet [Result Status](#) 2.16.840.1.113883.11.20.9.39 **STATIC** 2013-08-09 (CONF:14848).

*Figure 14: C-CDA R2 conformance statement representation example of a template design body*

## Embedding or Referencing other Artifacts

Vocabularies (mainly Value Sets) that are used in template definitions shall be made available (in computable format) to support optimal implementation. Please note that license restrictions of certain vocabularies shall be mentioned in the corresponding implementation guide (or template definition).

The same applies to templates that are included or contained in template definitions. They should be easily accessible (in computable format, i.e. the ITS described in this specification) through template repositories or by pointing to them in other publications.

**Table 405: Tobacco Use**

Value Set: Tobacco Use 2.16.840.1.113883.11.20.9.41		
Specific URL Pending		
Valueset Source: <a href="http://vts1.vetmed.vt.edu/">http://vts1.vetmed.vt.edu/</a>		
Code	Code System	Print Name
81703003	SNOMED CT	Chews tobacco
228494002	SNOMED CT	Snuff user
59978006	SNOMED CT	Cigar smoker
43381005	SNOMED CT	Passive smoker
228524006	SNOMED CT	Exposed to tobacco smoke at home
427189007	SNOMED CT	Maternal tobacco use
394871007	SNOMED CT	Thinking about stopping smoking
65568007	SNOMED CT	Cigarette smoker
160619003	SNOMED CT	Rolls own cigarettes
266927001	SNOMED CT	Pipe smoker
...		

Figure 15: C-CDA R2 Value Set Table to demonstrate / list allowed codes

## Revision History / Change Log

For implementers it is very important to see what changes have been applied to a new version of a formerly implemented template. For that purpose a Revision History or Change Log should be available, either documented by the template designer or generated automatically by comparison of the two template designs.

### 8.106 Reason for Referral Section (V2)

[Reason for Referral Section \(V2\) \(1.3.6.1.4.1.19376.1.5.3.1.3.1.2\)](#)

Change	Old	New
Name	Reason for Referral Section	Reason for Referral Section (V2)
Oid	1.3.6.1.4.1.19376.1.5.3.1.3.1	1.3.6.1.4.1.19376.1.5.3.1.3.1.2
Description	A Reason for Referral section records the reason the patient is being referred for a consultation by a provider. An optional Chief Complaint section may capture the patient's description of the reason for the consultation.	This section contains the reason(s) for a patient's referral by a provider to a consulting provider. An optional Chief Complaint section may capture the patient's description of the reason for the consultation.
CONF #: 30808 Added		MAY contain zero or more [0..*] entry (CONF:30808) such that it
CONF #: 30897 Added		SHALL contain exactly one [1..1] Patient Referral Act (NEW) (templateId:2.16.840.1.113883.10.20.22.4.140) (CONF:30897).
CONF #: 30867 Added		This code SHALL contain

Figure 16: C-CDA R2 Change Log Example

## Appendix D – References

Extensible Markup Language (XML) 1.0 (Fifth Edition),  
<http://www.w3c.org/TR/2008/REC-xml-20081126/>

HL7 Clinical Document Architecture (CDA Release 2).  
<http://www.hl7.org/implement/standards/cda.cfm>

HL7 Version 3 Interoperability Standards, Normative Edition 2010.  
<http://www.hl7.org/memonly/downloads/v3edition.cfm> - V32010

- [ccdar1] HL7 Implementation Guide for CDA® Release 2: IHE Health Story Consolidation, DSTU Release 1.1 (US Realm) Draft Standard for Trial Use, January 2013
- [ccdar2] HL7 Implementation Guide for CDA® Release 2: Consolidated CDA Templates for Clinical Notes (US Realm) Draft Standard for Trial Use Release 2, HL7 Draft Standard for Trial Use (DSTU) Ballot, September 2013
- [v2xml] XML Encoding Rules (HL7 Version 2: XML Encoding Syntax, Release 1)  
[http://www.hl7.org/implement/standards/product\\_brief.cfm?product\\_id=83](http://www.hl7.org/implement/standards/product_brief.cfm?product_id=83)
- [fhir] FHIR® – Fast Health Interoperable Resources, <http://hl7.org/fhir>
- [isosch] ISO/IEC 19757-3:2006 Information technology -- Document Schema Definition Language (DSDL) -- Part 3: Rule-based validation – schematron,  
<http://www.schematron.com>
- [isoid] ISO/DTS 13582:2012 Health Informatics — Sharing of OID registry information
- [mif] HL7 Version 3 Standard: Model Interchange Format, Release 1,  
[http://www.hl7.org/implement/standards/product\\_brief.cfm?product\\_id=101](http://www.hl7.org/implement/standards/product_brief.cfm?product_id=101)