Australian Government
Australian Digital Health Agency

Representing Coding in CDA Documents Implementation Guidance

11 October 2011 v1.0
Approved for external use
Document ID: NEHTA-1097:2011

Australian Digital Health Agency ABN 84 425 496 912, Level 25, 175 Liverpool Street, Sydney, NSW 2000
Telephone 1300 901 001 or email help@digitalhealth.gov.au
www.digitalhealth.gov.au

Acknowledgements
The Australian Digital Health Agency is jointly funded by the Australian Government and all state and territory governments.

Document information

Key information

Owner Director, Interoperability Products

Contact for enquiries Australian Digital Health Agency Help Centre
Phone 1300 901 001
Email help@digitalhealth.gov.au

Product or document version history

Product or document version	Date	Release comments
1.0	11/10/2011	First Release
1.0	13 May 2025	The document presentation has been enhanced to align with current branding guidelines, however the content has not been changed.

Transition of terms

Certain terms used within the context of this document have changed. The table provides a clear comparison of the historical terms used in text and their current equivalents for your reference.

Historical term	Current term
National eHealth Transition Authority (NEHTA)	The Australian Digital Health Agency (ADHA)

Table of contents

1 Introduction

This document provides advice for implementers with respect to coding common clinical concepts in NEHTA CDA documents, including e-Referral, Discharge Summary, Event Summary, Shared Health Summary, and Specialist Letter. This document describes how to correctly encode a concept for maximum utility in the future.

Note that this document is only concerned with the correct way to represent concepts in a CDA document. NEHTA may additionally make extra rules concerning the particular codes that must be used, but these rules will apply in addition to this document.

The first part of this document describes how the coding data types work in principle. The second half of the document describes the common coding scenarios in a recipe-like format.

1.1 Warning/Disclaimer

In the future, NEHTA expects to publish other documents that provide guidance with relationship to coding practices. As these documents are published, updated versions of this document may be released that describe more sophisticated coding practices (particularly with regard to SNOMED CT-AU alignment).

The following points should be noted when reading this document;

- While this document describes general ways to represent codes and concepts in CDA document, and mappings from one code system to another, it does not endorse any particular code system, mappings or coding processes, even those mentioned here. Rather, this document describes how to represent coding of various concepts in CDA. It does not provide advice about how coding should be done at the clinical level, or how mapping should be done. In particular, users should use caution when mapping between coding systems (especially from classifications to terminologies) and also when evaluating background language-based coding systems.
- Note that the sections titled 'Code Systems & Versions' and 'OIDS for Coding' does not extend to any discussion of implementation challenges, some of which will be clinical safety significant, where a single codeset per value domain (e.g. Diagnosis, Substance etc) has not been mandated nationally.
- Maintaining mappings between large codesets is considered an interim state. The ongoing maintenance of such mappings carry a significant overhead and complexity. It should also be noted that there are inherent limitations inherent in relying on human readable message content as an alternative to machine readable codes.

1.2 Glossary

This document assumes that the reader is familiar with the NEHTA specifications, and the CDA specification. For convenience, this glossary covers common terms used in this document.

SCS	Structured Content Specification
SDT	Structured Document Template (now replaced with Structured Content Specification)
CD	HL7 data type for coded element
CE	HL7 data type for coded element (same as CD, but slightly different rules)
OID	Object Identifier (see ISO/IEC 9834-1)
UUID	Universal Unique Identifier (or “GUID”) – see ISO/IEC 11578:1996
SNOMED CT-AU	Structured Nomeclature for Medicine (https://www.snomed.org)
AMT	Australian Medicines Terminology (https://www.digitalhealth.gov.au/)
ICPC2+	International Classification of Primary Care (See http://www.fmrc.org.au/icpc2plus/structure.htm)
ICD-10-AM	International Classification of diseases - http://nccc.uow.edu.au/icd10am/ic d10am/index.html
DOCLE	Clinical Coding system. http://www.docle.com.au/
MIMS	Medication information – see http://www.mims.com.au
LOINC	Logical Observation Identifiers Names and Codes (see https://loinc.org/)

2 NEHTA Specifications and Coded Data Types

In the Structured Content Specifications (SCSs), there are two data types for elements that may be coded – “Coded Text” and “Codeable Text”. Both specify that the content may be represented by a code, but the rules around each differ. When they are used, the coded data types may have a “value set” assigned to them – this defines the list of codes that are supposed to be used for this data element.

In the NEHTA specifications for coded elements, there are 3 possibilities.

SCS Type	Coded Text	Codeable Text
Data Element has a value set	✓	✓
Data Element does not have a value set	✗	✓

In the SCSs, a data element may have a type of either Coded Text or Codeable Text. Also, an element may have a value set assigned. The difference between these is important.

Coded Text 

This is a reference to a concept – the intent is that this is a code picked directly from a list of possible codes. If the field is mandatory, then a code from the specified value set must be provided – it is not valid to just provide text. If there is no known code from the list of possible codes, then there is no proper value. Text may be provided in addition to the code. Because of this, Coded Text is only used when a value set is assigned, and generally for status or workflow options (Note for CDA centric readers: Coded Text = CV CNE)

Codeable Text 

This is a reference to a concept, where the concept might be represented as either a code or text. Text is a proper alternative, but a code should be provided. It is usually appropriate to provide text with the code as well. Codeable Text data types may have a fixed value set assigned to them, or they may be open (any code or text at all), though NEHTA intends to specify value sets for all codeable text elements eventually. (Note for CDA centric readers: Codeable Text = CD CWE)

Note that some existing NEHTA specifications use a Coded Text without assigning a value set. These will be fixed by issuing a technical correction, either by changing the field to Codeable Text or by choosing an appropriate value set.

2.1 CDA “CD” Data Type

Both the Coded Text and Codeable Text data types are represented using the CE/CD data types in CDA specifications. The CD and CE data types are closely related and share the following aspects:

Group	Attributes	Meaning
Code	code : string codeSystem : string codeSystemVersion : string	Identifies the code system and code defined by it
Display	displayName : string	One defined display representation for the code
Text	originalText : ST (element)	Provides the text that the user said/typed/chose when picking the code or in place of the code
Translations	Translation (element)	Recursive reference to more of the same type.

Notes about this type:

- None of the attributes have any length limits, nor does the originalText text content
- The translations have the same type as the containing class (CD or CE). See further discussion below
- CE is different to CD because it allows “post-coordination” see below under “Complex Coded Systems”
- The displayName is provided so that an end-system that doesn’t know the coding system can still display something useful to its users
- It can be difficult at times to determine what the original text is. Sometimes it has the same value as the displayName, but this does not make it redundant. See further discussion below.

In XML, this looks like:

```
<x nullFlavor=”[NF]” code=”[code]” codeSystem=”[oid]”
    displayName=”[display]”/>
  <originalText>[text]</originalText>
  <translation nullFlavor=”[NF]” code=”[code]” codeSystem=”[oid]”
    displayName=”[display]”/>
</x>
```

Note that throughout this document, “x” will be used as the element name for the coded element, as it may be either “code”, “value”, or something else though the various elements of the CDA document.

There is a relationship between the nullFlavor, code, codeSystem, and text. The interplay between the various aspects of the coded data types and the NEHTA specifications can be complex, particularly once translations come into play. The rest of this document works through the common scenarios for Australian vendors explaining how to represent them in a CDA coded data type.

2.1.1 nullFlavors

The CD data type includes an attribute called “nullFlavor”. This attribute is used to indicate why the coded value is unknown. It can have one of the following values:

Code	Name	definition
NI	No Information	The value is missing for some unknown reason Note that `<x nullFlavor="NI"/> is exactly the same as not including <x>` at all.
UNK	unknown	The value is not known.
ASKU	asked but unknown	Information was sought but not found (e.g., patient was asked but didn't know)
NAV	temporarily unavailable	Information is not available at this time but it is expected that it will be available later.
NASK	not asked	This information has not been sought (e.g., patient was not asked)

Note indenting – it denotes that there are relationships between the codes; e.g. “ASKU” is a special type of “UNK” – if something is ASKU, it is also UNK.

In addition to this table, the value of the nullFlavor can be “OTH” – this is a special value that means that the “concept” – the meaning – is known, but it is not a valid code in the context of the specified value set (see below). OTH should only be used as described below.

Sometimes coding systems include codes that overlap with the nullFlavors. For instance, this is the indigenous status from Meteor:

Code	displayName
1	Aboriginal but not Torres Strait Islander origin
2	Torres Strait Islander but not Aboriginal origin
3	Both Aboriginal and Torres Strait Islander origin
4	Neither Aboriginal nor Torres Strait Islander origin
9	Not stated/inadequately described

The “9” code overlaps with the nullFlavor values, though not clearly. As a rule of thumb, using a code is preferable to a nullFlavor if a suitable code exists.

2.1.2 Code Systems and Versions

The codeSystem is an attribute that identifies the coding system by providing an OID (2.16.840.1.113883.6.96) or a UUID (441D40AF-0A07-426C-96AA-00E9D4C4A713).

(UUIDs are also known as GUIDs). These are opaque identifiers that uniquely identify a coding system (opaque because when you look at them, you cannot determine what they mean).

The first use of the codeSystem attribute is to keep codes apart, so that the code “X” used by one system isn’t accidentally confused with the code “X” used by another system for another use. When vendors are assigning identifiers to internal code systems, this is the first thing to keep in mind – codes must never clash within a code system.

The second use of the codeSystem attribute is to recognise the coding system, so that the code can be interpreted correctly. For instance, 2.16.840.1.113883. 6.96 identifies the SNOMED CT terminology. HL7 maintains an OID registry at http://www.hl7.org/oid/index.cfm?ref=common where OIDs must be registered. So it is possible to resolve any OID in a CDA document using this registry. The second thing to keep in mind when assigning identifiers to internal coding systems is that if they are going to be used in CDA documents, they will need to be registered with a coherent useful description of the code system.

In addition to the codeSystem attribute, there is also a codeSystemVersion attribute which exists to handle changes in meaning of a code over time. In principle, the meaning of a code should never change over time – the definition of “X” should always mean the same thing.

However in practice, the meaning of codes may change over time – occasionally the definitions are revised to “clarify” meaning. Because of this, it is beneficial to supply a code system version if it is known, although the NEHTA specification examples do not generally provide a code system version, and no version is defined for the NEHTA defined code systems used in the CDA documents.

2.1.3 Picking Original Text

OriginalText has two uses:

When there is no right code to pick, the original text is the meaning of the concept

Even when there is a code, the code does not always capture all the details and/or nuances that the user had in mind. In this case, the originalText may be (should be) closer to the user’s meaning.

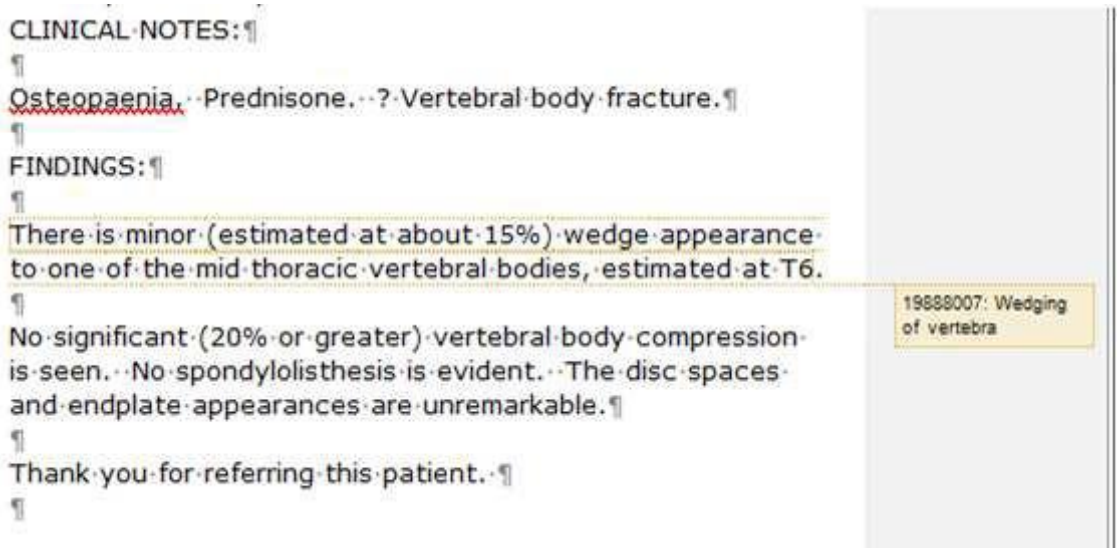
Some code systems do not have the ability to code data immediately and rely on “post coding”. In this case, the original text captures the concept as expressed by the data enterer prior to coding.

It can be difficult to determine what the correct original text is. Here is a guide to common scenarios:

Scenario	Original Text
User picks a code from a list of codes, displayed as the codes themselves (usually this only works with small lists of well known terms, particularly where the codes are meaningful)	None

Scenario	Original Text
User picks a code from a list of codes, displayed as text	Display text
User typed some text which was processed in the background	Text user typed
User typed some text which started a code look up	The text description of the code they picked
User typed some text which was processed into a suggested list of codes, and then the user typed more text to further narrow the suggested list	The choice of "original text" becomes a little arbitrary; in the case where the original text stands as part of a report (see image below), the first original text applies.
User chose a code from a list and typed more text to clarify further (see image below)	The display name for the code, with the clarifying text appended.

Text as part of a report:



Clarifying text:

In the case of the first example, it would be possible to assign several codes to the narrative; the different codes are different CodeableText values in the structured data. In the context of CDA, the text would be rendered narrative for the section containing the (in this case) radiology report. The data entries include the CD data types that correspond to the Codeable Text data elements. In these cases, the CD data types can refer to the content in the narrative directly instead of duplicating the text.

In practice, the CDA narrative content would look like this:

```
<text>
  <paragraph>CLINICAL NOTES:</paragraph>
  <paragraph>
    Osteopaenia. Prednisone. ?Vertebral body fracture.
  </paragraph>
  <paragraph>FINDINGS: </paragraph>
  <paragraph>
    <content id="e23">
      There is minor (estimated at about 15%) wedge appearance to
      one of the mid thoracic vertebral bodies, estimated at T6.
  </paragraph>
  <paragraph>
    No significant (20% or greater) vertebral body compression is
    seen. No spondylolisthesis is evident. The disc spaces and
    endplate appearances are unremarkable.
  </paragraph>
  <paragraph>Thank you for referring this patient.</paragraph>
</text>
```

A CD value referring to the text as shown above would be constructed like this:

```
<value code="19888007" codeSystem="2.16.840.1.113883.6.96"
    displayName="Wedging of vertebra">
  <originalText><reference value="#e23"/><originalText>
</value>
```

Conformant CDA implementations must always be able to resolve the original text by following a reference instead of expecting the text to be provided directly in the originalText.

2.1.4 displayName

The CD data type includes a displayName attribute, which is the text that is designated for use to represent the code/concept by the coding system. The following table summarises the source of the displayName for common code systems:

Code System	Source of displayName
SNOMED CT-AU	Preferred name in the Australian English Language reference set
AMT	Preferred Name (or, for v3, Preferred name in the Australian English Language reference set)
HL7 code systems and v2 tables	The Print name for the code
ICD-10-AM	Preferred Name

Code System	Source of displayName
ICPC2+	The ICPC2+ term for the code
MIMS	The display term provided by MIMS
DOCLE	Tbd

2.1.5 Translations

Translations exist to allow in place mappings between different code systems, for instance, if a sender is using ICPC2+, and a receiver expects SNOMED CT-AU. They also allow for a community of systems (including senders) to gracefully transition from one coding system to another. As this is the situation we find ourselves in – multiple systems using different coding systems, and all systems gradually migrating to SNOMED CT-AU, translations are important. Although the translation structure is recursive (CDs contain translations which are CDs, which contain translations…), the translations should not have originalText (there's only one "text" for the entire concept) and there should be no nested translations (there is an advanced use case for them, but it does not apply in Australia).

Generally, if the root code is a LOINC, SNOMED CT-AU or AMT code, there is no need for translations (there is one minor edge case described below where this occurs).

Translations should not be used with data elements that have a type "Coded Text".

2.1.6 Value Set vs Code System

As explained above, the data elements may have a value set assigned, which specifies the set of allowed values for the codes. Simple value sets specify a list of possible codes. However more complicated value sets are possible that control how complex coded expressions are used (see below).

In the context of NEHTA specifications, when value sets are assigned, they are always simple lists of codes taken from a single code system. Most of the NEHTA specified value sets – especially the clinical ones – are taken from SNOMED CT-AU or AMT. In these cases, the value sets are simple lists of possible codes defined as SNOMED CT-AU reference sets, and included in the SNOMED CT-AU and AMT releases. Note that there is a case where one value set includes codes from both SNOMED CT-AU and AMT, however these are actually the same coding system, with the same OID.

Because all the value sets are based on a single coding system, this document refers to an expected code system or an expected value set interchangeably. Because CDA doesn't directly reference the value set, but does reference the code system directly, the document focusses on expected code systems.

In the absence of an applicable value set, any reference below to an expected value set or code system does not apply.

2.1.7 Complex Coded Expressions

Most coding systems allow for various forms of combinatorial expressions. For instance, SNOMED CT-AU allows for post-coordination using a defined expression language:

<value code="128045006:{363698007=56459004}"
codeSystem="2.16.840.1.113883.6.42">

<originalText>Cellulitis of the foot</originalText>

</value>

Note that the SNOMED CT-AU expression syntax allows additional "display text" to be included in the expression (following each code) surrounded by pipes (|) as in: 123 | foot fracture |: 456 | laterality | = 789 | left |. This form should not be used – display name should go in the displayName attribute.

ICD-10 allows for dual coding, where one code clarifies the other. Here's an example:

<value code="J21.8 B95.6" codeSystem="2.16.840.1.113883.6.260">

<originalText>Staph aureus bronchiolitis</originalText>

</value>

DOCLE and others also make use of combinations of codes.

These expression based coding systems are a problem; while the need for such capability arises innately and obviously to clinical users, all aspects of their implementation are difficult, and support for them is not generally available within Australian clinical systems. At this time, NEHTA recommends against use of post- coordination in CDA documents (note that the reference sets are all enumerations of pre-existing codes, and SNOMED CT-AU expressions are never valid members of the reference sets).

The CD and the CE data type differ in that the CD data type allows for "qualifiers" – additional qualifiers that modify the meaning of the primary code. These qualifiers are intended to support representation of these complex coded expressions in CDA documents, but they are too complex[1]. Instead of using the CD qualifiers, implementers should use expressions in codes as shown in the examples above except as specifically described otherwise in NEHTA specifications. This document does not specify whether implementers should use or support such expressions in their systems. However implementers should be aware that for some coding systems (those listed below) the code and displayName[2] attributes may be quite long, and should not be truncated.

This table summarises the state of expression support in the relevant code systems:

Code System	Expressional Syntax	Maximum Length
SNOMED CT (and -AU)	The expression syntax defined by IHTSDO[3]	Potentially very long, possibly >255 chars, though this is unusual
DOCLE	TBD	

2.2 OIDs for coding systems

As described above, Code systems are identified by an OID or a UUID which uniquely identifies the coding system. Any coding systems used in NEHTA CDA documents that are identified by an OID must be registered in the HL7 Intl OID registry at http://www.hl7.org/oid/index.cfm?ref=common . OIDs must be used – and registered –

[1] Qualifiers are too complex because:
- The qualifier semantics do not match the semantics in MIMS, DOCLE, etc
- The use of expressional coding is a matter for value sets, not the CDA typing system

[2] There is contention over what is the appropriate displayName for a SNOMED CT-AU expression. Some algorithmic methods have been described, but these are not yet accepted. In the absence of an agreed method, the displayName should be left blank, and an originalText should always be populated with an expression.

[3] https://confluence.ihtsdotools.org/display/DOC

except in the case below where UUIDs should be used. UUIDs should not be registered at this time.

NEHTA will register the commonly used shared coding systems in Australia. Vendors that have their own coding Systems will need to assign and register the code systems themselves (it is recommended that vendors use their ACN based OID – see http://www.alvestrand.no/objectid/1.2.36.html - note vendors do not need to register their own OIDs at that site).

If the code system is not found there, consult with NEHTA. This table summarises the OIDs for common coding systems:

Coding System	OID	Notes
SNOMED CT	2.16.840.1.113883.6.96	Includes SNOMED CT-AU
AMT	2.16.840.1.113883.6.96[4]	To specify AMT specifically, use the following string for the codeSystemVersion
Loinc	2.16.840.1.113883.6.1	
ICD-10	2.16.840.1.113883.6.3	
ICD-10-AM	2.16.840.1.113883.6.135	
MIMS	1.2.36.1.2001.1005.11.1	This describes the codes as specified by the MIMS Integrated Data Solution
ICPC 2+	2.16.840.1.113883.6.140.1	
DOCLE	1.2.36.1.2001.1005.13	
PBS Code	tbd	
MBS Code	tbd	

- Using expressions in place of codes allows systems that are not aware of the expression syntax, or not equipped to handle it, to treat an expression as an opaque code (this has its limitations, as SNOMED CT expressions can create multiple different expressions for the same concept)
- For all these and other reasons, HL7 removed qualifiers in later data type releases
- The NEHTA internal process is based on the later data type releases.

Note that the CDA Implementation Guides contain many small terminologies which are documented in place where they are used. This table focuses on the main commonly used clinical coding systems

[4] The assigned OID for AMT versions 1 and 2 is 1.2.36.1.2001.1004.100, and this has been used throughout the NICTIS specifications until now. However with AMT version 3, this is OID is going to change to the SNOMED CT OID in order to foster better integration with SNOMED CT-AU (and because this is correct). Technical corrections will be issued to clarify this change

2.3 Common Scenarios for Coding

There are two possible approaches to coding, depending on whether the type of the data element in the SCS is “Coded Text” or “Codeable Text”.

For Coded Text, these are the possible scenarios:

1. The correct code is known
2. The correct code is not known

For Codeable Text, these scenarios apply:

3. The value (Coded or not) is not known at all
4. User picks code directly from the value set
5. User enters text
6. User picks a code provided by some other code system (e.g. MIMS, ICPC2+, ICD-10, DOCLE, etc).
7. User picks a code from another code system and then provides additional clarifying text
8. User chooses a code they have defined themselves
9. The CDA document is being prepared on an interface engine from a v2 CWE type, and it is not known which of processes #4 - #8 applied.

Note that in cases 5 through to 8, a code in the expected code system could be determined by either consulting a mapping table, or using some form of linguistic/statistical analysis. At the time of preparation of this document (Sept 2011), the generally available linguistic/statistical mapping processes are far from ready for production. This means that the primary reliance will be on mapping tables, which will gradually become available. This document describes how to code the scenarios above both with and without such mapping tables on the grounds that they will gradually become available, and the CDA documents will gradually migrate towards better coding.

2.3.1 Picking the scenario

The following checklist assists in determining the applicable scenario:

- Is the type of the data element Coded Text or Codeable Text (SCS)?
- What value set is assigned to the data element (CDA implementation Guide)?
- What value set and/or code system does the application use?

2.3.2 What happens when the user cannot find an appropriate code? #1: Coded Text; The correct code is known

Coded text is simple – either the correct code is known, or it is not. If the correct code is known, then it is used directly

```
<x code=”01” codeSystem=”1.2.36.1.2001.1001.101.104.16299”
   displayName=”None known”>
</x>
```

If desired, an originalText can be provided.

```
<x code="01" codeSystem="1.2.36.1.2001.1001.101.104.16299"
   displayName="None known">
  <originalText>There are no known medications</originalText>
</x>
```

It is not usually appropriate to provide an originalText for a Coded Text data element; the choice lists are usually small and infrastructural. In the specific case above, the original text would correspond to the caption/label on the radio button that the user checked to choose "None known", but this should not imply anything different to the meaning of the code.

2.3.3 #2: Coded Text; The correct code is not known

If the correct code is not known, then a nullFlavor is used:

```
<x nullFlavor="UNK" codeSystem="2.16.840.1.113883.3.879">
</x>
```

This says that the value of the indigenous status is unknown.

It may be appropriate to provide additional text if some additional information is known that cannot be coded correctly:

```
<x nullFlavor="UNK" codeSystem="2.16.840.1.113883.3.879">
  <originalText>Chinese Malay / Aboriginal</originalText>
</x>
```

Note that the value is still unknown. Most of the Coded Text value sets contain codes for unclear concepts such as these (1 or 9 in this case), and use of originalText in this context should always be reviewed.

2.3.4 #3: Codeable Text; The value (Coded or not) is not known at all

For data elements of type Codeable Text, if the correct value is not known at all, then a nullFlavor is used:

```
<x nullFlavor="NASK">
</x>
```

This indicates that the value of the data element is unknown because the patient was not asked. In some cases, it might not be known why the data element is missing. In these cases:

```
<x nullFlavor="NI">
</x>
```

This is equivalent to simply omitting "x" from the CDA document altogether (which is also valid).

Note that for CodeableText, you should not provide a nullFlavor and an originalText – if any text is known, then the concept is not null.

2.3.5 #4: Codeable Text; User picks code directly from the expected value set

In this case, the correct code system is being used. For example, if this is SNOMED CT-AU, and the user chose the code 263063009 (Fracture dislocation of joint), and there is no applicable value set, or the code is in the value set, the code would be represented as

```
<x code=”263063009” codeSystem=”2.16.840.1.113883.6.96”
   displayName=” Fracture dislocation of joint”>
 <originalText>Fracture dislocation of joint</originalText>
</x>
```

In the very unlikely case that the user picked the code “263063009” directly without seeing any display text, the code would be represented as:

```
<x code=”263063009” codeSystem=”2.16.840.1.113883.6.96”
   displayName=” Fracture dislocation of joint”>
</x>
```

This form of representation is more likely for coding systems other than SNOMED CT-AU (particularly smaller code systems where the codes are meaningful to humans). Here is a simple example:

```
<x code=”M” codeSystem=”oid for gender”
    displayName=” Male”>
</x>
```

2.3.6 #5: Codeable Text; User enters text

In this case, the user enters text – either the user application only has a text field for this value, or the user couldn’t find the code that said what they entered, so they entered text.

Continuing with the dislocation example, and assuming that the user has written “fracture/dislocation”, the text would be represented like this:

```
<x>
  <originalText>Fracture/dislocation</originalText>
</x>
```

If a code in the target coding system is later generated based on some linguistic/statistical process, it can be added as a translation

```
<x>
  <originalText>Fracture/dislocation</originalText>
  <translation code=”263063009”
   codeSystem=”2.16.840.1.113883.6.96”
   displayName=” Fracture dislocation of joint”/>
</x>
```

In advanced use cases, it may be useful to indicate that the user did look for a code before entering text. Note that whether this is known depends on the application work flow. The following example demonstrates the correct way to represent that the coding was not possible:

```
<x nullFlavor=”OTH” codeSystem=”2.16.840.1.113883.6.140.1”>
  <originalText>Fracture/dislocation</originalText>
</x>
```

(This example uses ICPC2+, which has the OID 2.16.840.1.113883.6.140.1, as the original coding system). If the text is later mapped to SNOMED CT-AU:

```
<x nullFlavor=”OTH” codeSystem=”2.16.840.1.113883.6.140.1”>
  <originalText>Fracture/dislocation</originalText>
  <translation code=”263063009”
   codeSystem=”2.16.840.1.113883.6.96”
   displayName=” Fracture dislocation of joint”/>
</x>
```

2.3.7 #6 Codeable Text; User picks a code provided by some other code system

Typical examples for these other coding systems are MIMS, ICPC2+, ICD-10, DOCLE, etc. As an example, assume that the user picks the ICPC2+ code “L76013” (Fracture), and the value set is “SNOMED CT-AU Problem/Diagnosis Reference Set”:

```
<x code=” L76013” codeSystem=”2.16.840.1.113883.6.140.1”
   displayName=” Fracture”>
  <originalText>Dislocation or fracture</originalText>
</x>
```

This assumes the user picked from a list that includes the text, not just the ICPC 2+ codes – in which case there would be no originalText. This representation also holds for the situation where the user typed the text first, and some additional process followed that led to picking the code.

If a code in the SNOMED CT-AU Problem/Diagnosis Reference Set is available from either mapping or a linguistic/statistical process, it is added as a translation:

```
<x code=”L76013” codeSystem=”2.16.840.1.113883.6.140.1”
   displayName=” Fracture: other”>
 <originalText>Fracture dislocation of joint</originalText>
 <translation code=”263063009” codeSystem=”2.16.840.1.113883.6.96”
  displayName=” Fracture dislocation of joint”/>
</x>
```

Note that even if the expected code is not available when the document is written, because the code/codeSystem that the user picked is correctly coded, when the mappings become available later (or the linguistic/statistical processes improve to become useable later), systems can convert to the desired code system.

In the case of MIMS, MIMS integrated provides the translations directly – in fact, they provide the CD form directly - and the CD should take this form:

```
<code code="57790101" codeSystem="1.2.36.1.2001.1005.11.1"
   codeSystemName="MIMS Standard Code set"
   codeSystemVersion="20110900"
   displayName="Clavulin Duo Forte Tablets 875 mg/125 mg [10]">
  <translation code="13301011000036101"
   codeSystem="2.16.840.1.113883.6.96"
   codeSystemName="Australian Medicines Terminology (AMT)"
   codeSystemVersion="2.25" displayName="Clavulin Duo Forte 875/125
     tablet: film-coated, 10 tablets"/>
</code>
```

(An originalText may also be provided where appropriate, though in this case it will usually be the same as the MIMS displayName).

There is an unusual variation to this case – where the user picks a SNOMED CT code, but it is not in the correct value set (the SNOMED CT-AU Problem/Diagnosis Reference Set in this case). However by the rules of Codeable Text, this is still a valid concept:

```
<x code="209393006" codeSystem="2.16.840.1.113883.6.96"
   displayName="Other open fracture dislocation"/>
  <originalText>Fracture dislocation of joint</originalText>
</x>
```

If this gets mapped into the right reference set later:

```
<x code="209393006" codeSystem="2.16.840.1.113883.6.96"
   displayName="Other open fracture dislocation"/>
  <originalText>Fracture dislocation of joint</originalText>
  <translation code="263063009"
   codeSystem="2.16.840.1.113883.6.96"
   displayName=" Fracture dislocation of joint"/>
</x>
```

(This at least could be done automatic ally based on the definitions in SNOMED CT-AU itself).

2.3.8 #7 Codeable Text; User picks a code from another code system and then provides additional clarifying text

In this case, the user picks some code from the coding system, and then provides some further clarifying/qualifying text. Here's an example UI (copied from above):

In this case, the "text" is the displayName of the code + the extra text. Usually a separator is used in the original text, so it looks like this:

> "Aneurysm;artery;cerebral – minimum deficit"

This modified original text swallows up all the other possibilities as the "text that the user intended", and the code would be represented like this:

```
<x code="K90001" codeSystem="2.16.840.1.113883.6.140.1"
   displayName="Aneurysm;artery;cerebral">
  <originalText> Aneurysm;artery;cerebral – minimum deficit
  </originalText>
</x>
```

If the code is mapped to the expected code set, then it would be represented like this:

```
<x code=”K90001” codeSystem=”2.16.840.1.113883.6.140.1”
 displayName=”Aneurysm;artery;cerebral”>
 <originalText> Aneurysm;artery;cerebral – minimum deficit
 </originalText>
 <translation code=”128608001”
  codeSystem=”2.16.840.1.113883.6.96”
  displayName=”Cerebral arterial aneurysm”/>
</x>
```

In this case, SNOMED CT-AU does not appear to have a more specific code for “Cerebral arterial aneurysm with minimum deficit”, but if such a code existed, and the tooling was capable of performing the mapping, it could also be used.

2.3.9 #8 Codeable Text; User chooses a self-defined code

In some clinical systems, when a user cannot find a code that represents their intent, they can simply define their own code that only they see and use. Note that this process has obvious dangers, and the various clinical systems exert different levels of control over the appropriateness of this action. These considerations are out of scope for this document.

As an example, assume that the user encountered the situation above (Aneurysm;artery; cerebral – minimum deficit), and instead of offering the ability to provide extra qualifying text, the system allows the user to create their own code. If , the user creates a code “AA1001”, which means “Cerebral arterial aneurysm with minimum deficit”, and the user picks this new code, in CDA, this would be represented as:

```
<x code=”AA1001” codeSystem=”441D40AF-0A07-426C-96AA-00E9D4C4A713”
   displayName=” Cerebral arterial aneurysm with minimum deficit”>
 <originalText>Minimal deficit Cerebral arterial aneurysm</originalText>
</x>
```

The code system here is a UUID that scopes the code AA1001 so that it could never be confused with any else’s “AA1001” code, should they use that particular code. In practice, the codeSystem could be an OID, but this would require some kind of external system to distribute unique identifiers to the installed base of the application; UUIDs are much easier in this case (and may be generated by some system API such as coCreateGUID on Windows). In these cases, systems must track and store the UUID so that it is consistently used for this purpose.

In the long term, it is possible that systems to gather and map these custom codes to national code systems will be put in place (this is not possible now, but people are already interested in the idea). For this reason, vendors should keep appropriate local records over the local codes so that this might be possible in the future.

2.3.10 #9 Codeable Text; CDA generated on an Interface Engine from HL7 v2

In the short term, many CDA documents will be generated by an interface engine on the perimeter of an organisation from existing exchanges. In practice, this means converting v2 messages to CDA documents, and in most cases, the user process around the coding will not be known.

In these cases, the CD data type is generated from a CWE or CNE data type. Ideally there would be consistency between the v2 and NEHTA specifications, so that CNE maps

to Coded Text and CWE maps to Codeable Text, but this is not always the case. In general, though, there is not much difference between the CWE and CNE data types, and what difference there is often misunderstood.

The following table shows a mapping between the v2 CWE data type, and the v3/CDA CD data type. This table is based on HL7 v2.4 and is indicative; actual usage of the CWE data type varies widely in Australia, and implementers should consult their message specifications and sources carefully.

CWE Component	CD attribute	Notes
1: identifier	x.code	
2: text	x.displayName	
3: coding system	x.codeSystem	Conversion from Name to OID required – see below
4: Alternate Identifier	x.translation.code	
5: Alternate Text	x.translation.displayName	
6: Alternate Coding System	x.translation.codeSystem	Conversion from Name to OID required – see below
7: coding system version	x.codeSystemVersion	
8: alternate coding system version	x.translation.codeSystemVersion	
9: original text	x.originalText	

Additional Notes:

1. If the applicable type is Coded Text, a nullFlavor is required. If there is no first component but a third or ninth component is provided, then the nullFlavor is “OTH” otherwise it’s something else (“UNK”)
2. If there is no identifier (component 1), and a text (component 2) is provided, then component 2 maps to original text instead of displayName
3. The mapping cannot be completed if both components 2 and 5 are populated and components 1 and 4 are not populated. It is also an error if component 9 is populated and either components 2 or 5 are populated without a matching identifier. (Neither of these conditions is illegal in v2, but they are nonsensical)
4. Components 7 and 8 should be brought through to x.codeSystemVersion and x.translation.codeSystemVersion untouched (see note above about codeSystem versions).
5. Generally components 1-3 map to the root code, and components 4-6 map to a translation. But it’s not always like this - implementers should check the previous cases and the v2 implementation guides carefully to help determine which way they go
6. In v2.4, the identifier lengths are restricted so as not to support expressions. Users might choose to ignore the length restrictions as they are somewhat nominal.
7. Components 4 and 6 are names. In CDA, the code system is an OID. The following table maps between the code system names commonly encountered in

Australia and their respective OIDs. For other coding systems, see the section about OIDs above.

Code	Description	OID
I10	ICD-10 *(assume Australian variant)*	2.16.840.1.113883.6.135
LN	Logical Observation Identifier Names and Codes (LOINC(r))	2.16.840.1.113883.6.1
SCT	SNOMED Clinical Terms	2.16.840.1.113883.6.96
SCT2	SNOMED Clinical Terms *alphanumeric codes*	codes will need to be translated to normal SNOMED CT-AU concept identifiers)
SNM	Systemized Nomenclature of Medicine (SNOMED) (1984)	2.16.840.1.113883.6.5
SNM3	SNOMED International (1993)	2.16.840.1.113883.6.51

The codes SNM and SNM3 are included to support old SNOMED codes still used for registry reporting in Australia. These are not “SNOMED CT” codes, and so have different OIDs. They could be translated to SNOMED CT-AU. SCT2 should not be used in Australia.

Case Summary (Italics = content that might be present depending on the case)

	Case	Null-Flavor	code	codeSystem	displayName	originalText	Translation code	Translation codeSystem	Translation displayName
Coded Text									
1	The correct code is known		01	1.2.36.1.2001.1001.101.104.16299	None known	*There are no known medications*			
2	The correct code is not known	UNK		2.16.840.1.113883.3.879		*Chinese Malay / Aboriginal*			
Codeable Text									
3	The value (Coded or not) is not known at all	NASK							
4	User picks code directly from the expected value set		263063009	2.16.840.1.113883.6.96	Fracture dislocation of joint	*Fracture dislocation of joint*			
	Additional Example		M	[oid for gender]	Male				
5	User enters text					Fracture/dislocation			
	If subsequently translated					Fracture/dislocation	263063009	2.16.840.1.113883.6.96	Fracture dislocation of joint
	If coding wasn't possible	OTH		2.16.840.1.113883.6.140.1		Fracture/dislocation			
	If coding wasn't possible and then subsequently translated	OTH		2.16.840.1.113883.6.140.1		Fracture/dislocation	263063009	2.16.840.1.113883.6.96	Fracture dislocation of joint
6	User picks a code provided by some other code system		L76013	2.16.840.1.113883.6.140.1	Fracture	Dislocation or fracture			
	If subsequently translated		L76013	2.16.840.1.113883.6.140.1	Fracture	Dislocation or fracture	263063009	2.16.840.1.113883.6.96	Fracture dislocation of joint
	Weird variation – mapping to right Snomed CT-AU code		209393006	2.16.840.1.113883.6.96	Other open fracture dislocation	Fracture dislocation of joint	263063009	2.16.840.1.113883.6.96	Fracture dislocation of joint
7	User picks a code from another code system and then provides additional clarifying text		K90001	2.16.840.1.113883.6.140.1	Aneurysm;artery;cerebral	Aneurysm;artery;cerebral – minimum deficit			
	If subsequently translated		K90001	2.16.840.1.113883.6.140.1	Aneurysm;artery;cerebral	Aneurysm;artery;cerebral – minimum deficit	128608001	2.16.840.1.113883.6.96	Cerebral arterial aneurysm
8	User chooses a self-defined code		AA1001	441D40AF-0A07-426C-96AA-00E9D4C4A713	Cerebral arterial aneurysm with minimum deficit	Minimal deficit Cerebral arterial aneurysm			

2.4 Advice for Receivers

When receiving codes, you can reverse engineer the table above to be sure about the exact circumstance that applied. However it is generally not required to do so. The following simple advice suffices for most uses:

- Displaying the concept to the user
    - If you get an originalText, display this to the user
    - Otherwise, if you get one, the displayName
    - Otherwise, if you can, look up the code
    - Otherwise, the code, if you get one
    - Otherwise the nullFlavor description in brackets
    - If you don't get anything then ("blank" or "—") or equivalent
    - It is sometimes useful to display the code in brackets if assigned (alerts the user that the concept is coded, if the work flow depends on the code)
- Storing the concept
    - Codes, displayNames, and originalText may be arbitrarily long. (>255 chars is possible)
    - They should never be truncated
    - Some unlimited type storage is appropriate. In practice this is challenging; in the end most implementations choose some variation of storing the entire document as a blob, indexing the parts of the document that are used for searching/matching, and marking in those indexes where content has been truncated.
- Making decisions based on the code
    - Check the root and the translations for the preferred code
    - It may not matter whether the code is an expression or not (need to consult documentation on terminology service/library)