nehta

Clinical Package

Version 1.0 — 30 November 2011

Final

National E-Health Transition Authority Ltd
Level 25
56 Pitt Street
Sydney, NSW, 2000
Australia.
www.nehta.gov.au

Table of contents

Table of contents ..... **iii**

Document information ..... **v**
Release v
Change history v

1 Introduction **1**
1.1 Background 1
1.2 Purpose 1
1.3 Scope 1
1.4 References 2
1.4.1 Normative References 2
1.4.2 Informative References 2
1.5 Definitions, acronyms, abbreviations 3
1.5.1 Definitions 3
1.5.2 Acronyms and abbreviations 4
1.5.3 XML namespaces 4
1.5.4 Terminology 4
1.6 Overview 4

2 Clinical package **5**
2.1 Overview 5
2.2 Model 5
2.2.1 Class diagram 5
2.2.2 Concepts 6
2.2.3 Relationships 6
2.3 Model properties 6
2.3.1 Packages, root packages and referenced packages 6
2.3.2 Parts and byte streams 7
2.3.3 Members and identifiers 7
2.3.4 Distinguishers and distinguisher types 8

3 CP-ZIP **10**
3.1 Overview 10
3.1.1 Concepts 10
3.2 Representation 11
3.2.1 CP-ZIP 11
3.2.2 Root package index 12
3.2.3 Package index 12
3.2.4 Atomic item 15

Appendix A: XML Schemas **17**
A.1 Package index XML Schema 17

Appendix B: Clinical package profile **18**
B.1 Introduction 18
B.2 Contents 18
B.2.1 Logical model 18
B.2.2 Representation 18

Appendix C: Examples **19**
C.1 Example with parts 19
C.1.1 Introduction 19
C.1.2 Clinical package profile 19
C.1.3 Logical contents 19
C.1.4 CP-ZIP instance 19
C.2 Example with referenced package 21

C.2.1 Introduction 21
C.2.2 Clinical package profile 21
C.2.3 Logical contents 21
C.2.4 CP-ZIP instance 21
C.3 Example with parts and digital signatures 24
C.3.1 Introduction 24
C.3.2 Clinical package profile 24
C.3.3 Logical contents 24
C.3.4 CP-ZIP instance 24
C.4 Example with referenced packages and digital signatures 27
C.4.1 Introduction 27
C.4.2 Clinical package profile 27
C.4.3 Logical contents 27
C.4.4 CP-ZIP instance 28
C.4.5 Overview of digest values 32

Document information

Release

Status:	Final
Date:	2011-11-30
Author:	Hoylen Sue
Owner:	John McMillan

Change history

Version	Date	Comments
1.0	2011-11-30	First release

This page intentionally left blank.

1 Introduction

1.1 Background

Clinical information can comprise of data stored in multiple byte streams.

For example:

- A pathology report comprising of a CDA XML document and images that are attachments to that CDA XML document.
- A discharge summary comprising of a CDA XML document with an attachment that is a pathology report, where that pathology report comprises of another CDA XML document and images as its attachments.
- A prescription comprising of a CDA XML document, an attachment and a digital signature. The digital signature is a separate byte stream that signs the CDA XML document.
- A discharge summary comprising of a CDA XML document, an attachment image, an attachment prescription and a digital signature. The digital signature is a separate byte stream that signs the discharge summary CDA XML document. That prescription attachment is comprised of multiple byte streams.

These byte streams are typically thought of as files, but in this specification they will be referred to as byte streams because they do not necessarily have to be stored as separate files in a file system. For example, they could exist as data in a database or sent in a message.

Clinical information is made up of a set of members. As illustrated in the above examples, there are two types of members: members that are byte streams (e.g. the images) or other clinical information that has its own structure (e.g. the pathology report attachments). Some of these members are set apart from the other members (e.g. the CDA XML document can be distinguished so that clinical software can easily find it for processing).

1.2 Purpose

This specification defines a "clinical package" as a logical model of the data it contains. This model can be profiled to create data models for specific clinical data.

This specification also defines the "CP-ZIP" representation: a serialized syntax for representing instances of the *clinical package* logical model. This representation can be used to exchange instances of *clinical packages* between computer programs.

Specifications that use *clinical packages* can define *clinical package profiles* that constrain the logical model for *clinical packages* to a logical model that contains data that suits their requirements.

Specifications that need to exchange instances of clinical packages, or profiles of clinical packages, can use the CP-ZIP representation as the serialized representation.

1.3 Scope

This specification defines the logical model for *clinical packages* and a possible representation of that logical model.

It is out of scope for this specification to define how *clinical packages* are created or how they are processed.

The following are also out of scope:

- The format or contents of the byte streams.
- The members present in an instance of a *clinical package*.
- Types of distinguishers.
- The distinguishers present in an instance of a *clinical package*.

This additional information can be defined by *clinical package profiles*. Different clinical package profiles can be defined to satisfy different requirements, but it is outside the scope of this specification to define any *clinical package profiles*.

An example of a *clinical package profile* is the "CDA Package" profiles specified by [CDAP2011]. Those profiles define a *clinical package* that contains a mandatory root CDA XML document, optional packaged attachments and cryptographic signatures. The packaging of CDA XML documents is one possible use of *clinical packages*, but it can be used for packaging other types of information.

The CP-ZIP is one possible representation for *clinical packages*. Other representations are possible, but it is outside the scope of this specification to define them.

1.4 References

1.4.1 Normative References

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

[ANSI-X3.4-1968] American National Standards Institute, *Coded character set—7-bit American National Standard Code for Information Interchange*, ANSI X.3.4-1968.

[PK2004] PKWare Inc, *.ZIP File Format Specification*, version 6.2.0, 26 April 2004. http://www.pkware.com/documents/APPNOTE/APPNOTE_6.2.0.txt

[RFC3986] IETF, *RFC 3986: Uniform Resource Identifier (URI): Generic Syntax*, T. Berners-Lee, R. Fielding, L. Masinter, January 2005.

[XSD2004a] W3C, *XML Schema Part 1: Structures*, Second Edition, W3C Recommendation, 28 October 2004.

[XSD2004b] W3C, *XML Schema Part 2: Datatypes*, Second Edition, W3C Recommendation, 28 October 2004.

[XML2008] W3C, Extensible Markup Language (XML) 1.0, Fifth Edition, W3C Recommendation, 26 November 2008.

1.4.2 Informative References

[CDAP2011] NEHTA, CDA Package 1.0.

[RFC2119] IETF, *RFC 2119: Keywords for use in RFCs to Indicate Requirement Levels*, S. Bradner, March 1997, http://ietf.org/rfc/rfc2119.txt

[RFC4122] IETF, *A Universally Unique Identifier (UUID) URN Namespace*, July 2005, http://ietf.org/rfc/rfc4122

[RFC5234] IETF, *Augmented BNF for Syntax Specifications: ABNF*, January 2008, http://ietf.org/rfc/rfc5234

[XNS2009] W3C, *Namespaces in XML 1.0*, third edition, W3C Recommendation, 8 December 2009, http://www.w3.org/TR/2009/REC-xml-names-20091208/

1.5 Definitions, acronyms, abbreviations

1.5.1 Definitions

Atomic item	Physical concept of a *ZIP item* that contains the *byte stream*.
Byte stream	Sequence of octets associated contained in a *part*.
Clinical package	A logical model of data.
Clinical package profile	A specification that constrains the *clinical package* logical model for specific purposes, specifying the *parts*, *referenced package* and *distinguishers*.
CP-ZIP	Physical concept of a format that represents instances of a *logical clinical package*. A "Clinical Package ZIP" archive profiles the general ZIP archive.
Distinguisher	Logical concept that tell a *member* apart from the other *members* of a *package*.
Distinguisher type	A URI that associated with a *distinguisher* to indicate its type.
Identifier	A URI-reference (i.e. relative references are permitted) unique to the scope of a *package* which is used to identify a *member* (i.e. a *part* or a *referenced package)* contained in that *package*.
Logical clinical package	A synonym for *clinical package*.
Member	Logical concept of data that makes up a *package*. All *members* have an *identifier*. There are two types of *members*: *parts* and *referenced packages*.
Octet	An 8-bit byte.
Package	Logical concept that is made up of *parts*, *referenced packages* and *distinguishers*.
Package index	Physical concept of data stored in a *ZIP item* to represent a *package*.
Package prefix	Physical concept of a string that is used in the derivation of *ZIP item names* from values found in a *package index*.
Part	Logical concept that is made up of a *byte stream* and an *identifier*. Contained in *packages*.
Referenced package	Logical concept of a *package* that has an *identifier*. Contained in another *package*.
Root package	*Logical concept* of a *package* that is not contained in another *package*.
ZIP archive	Physical artefact: data conforming to the syntax of the ZIP format.

ZIP item	Physical artefact: a unit of data inside a *ZIP archive*.
ZIP item name	Physical artefact: a printable US-ASCII string associated with a *ZIP item* identifying it in a *ZIP archive*.

1.5.2 Acronyms and abbreviations

XML	Extensible Markup Language
URI	Uniform Resource Identifier
UUID	Universally Unique Identifier

1.5.3 XML namespaces

`pi`	http://ns.electronichealth.net.au/pkg/PackageIndex/1.0

1.5.4 Terminology

The keywords **SHALL**, **SHALL NOT**, **SHOULD**, and **SHOULD NOT** in this document are to be interpreted as described in IETF's RFC 2119 [RFC2119].

1.6 Overview

The *clinical package* logical model is specified in chapter 2.

The CP-ZIP representation is specified in chapter 3.

The *Package Index XML Schema* is specified in Appendix A.

A non-normative description of what a *clinical package profile* could contain is in Appendix B.

Examples of *clinical packages* are presented in Appendix C.

2 Clinical package

2.1 Overview

This chapter defines the *clinical package* as a logical model for data.

The term "logical clinical package" is used in this document as a synonym for "clinical package" to emphasise the fact that the *clinical package* is a logical model which can have multiple representations.

This logical model can have different representations. Different representations are useful for different purposes. For example: internal representations in program memory for processing, database representations for storage and serialized representations for exchange. Since those different formats represent the same logical data, transformations can convert between the different formats without any loss of data from the logical model. It is possible for systems to store *clinical packages* in a different representation from which it was received in, and to output it in a different representation from which it was stored.

A possible representation for *clinical package* is the *CP-ZIP* representation defined in chapter 3.

Informally, a *clinical package* is made up of a set of *members* that are either *parts* or *referenced packages*. *Parts* are analogous to files. *Referenced packages* are other related *clinical packages*—allowing the nesting of *packages* inside *packages*. Some *members* can be *distinguished,* to tell them apart from the other *members* in that instance of a *clinical package*.

2.2 Model

2.2.1 Class diagram

Figure 1 shows the data model of *logical clinical packages*.

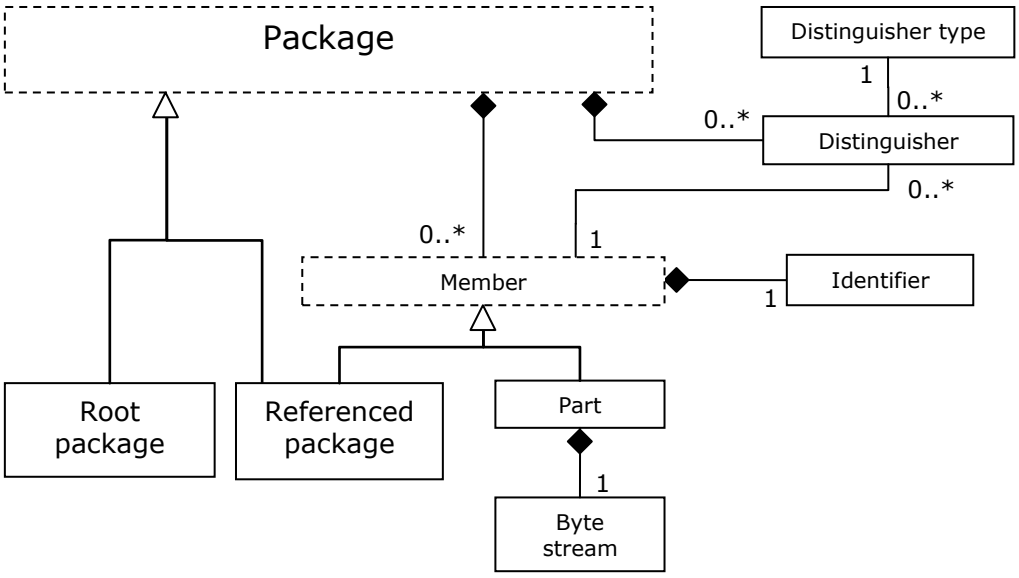

Figure 1 Logical clinical package class diagram

2.2.2 Concepts

The *logical clinical package* has these concepts:

- Package
- Root package
- Referenced package
- Member
- Identifier
- Part
- Byte stream
- Distinguisher
- Distinguisher type

2.2.3 Relationships

The *logical clinical package* has these relationships:

- A *package* has zero or more *members*.
- A *package* has zero or more *distinguishers*.
- A *member* has exactly one *identifier.*
- A *part* is a *member*.
- A *part* has exactly one *byte stream*.
- A *distinguisher* is associated with exactly one *distinguisher type*.
- A *distinguisher* is associated with exactly one *member*.
- A *root package* is a *package*.
- A *referenced package* is a *package* and is also a *member*.

2.3 Model properties

2.3.1 Packages, root packages and referenced packages

2.3.1.1 Introduction

A *package* comprises of *members* and some of those *members* might be distinguished.

There are two types of *packages*: *root packages* and *referenced packages*. A *root package* is a *package* that is not a member of another *package*, whereas a *referenced package* is a *package* that is a member of another *package*. Therefore, a *referenced package* has an *identifier*, but a *root package* does not.

The CP-ZIP representation in chapter 3 is defined as syntax for representing a single *root package*. *Referenced packages* only appear as members of *another package*.

2.3.1.2 Conformance points

2.3.1.2.1 Root package

PKG 1 A *root package* shall conform to all the conformance points for a *package* (section 2.3.1.2.3).

2.3.1.2.2 Referenced package

PKG 2 A *referenced package* shall conform to all the conformance points for a *package* (section 2.3.1.2.3).

PKG 3 A *referenced package* shall conform to the conformance points for a *member* (section 2.3.3.3.1).

2.3.1.2.3 Package

PKG 4 A *package* shall contain zero or more *members* that conform to the conformance points for *members* (section 2.3.3.3.1).

PKG 5 A *package* shall contain zero or more *distinguishers* that conforms to the conformance points for *distinguishers* (section 2.3.4.3).

2.3.2 Parts and byte streams

2.3.2.1 Introduction

Parts represent a *byte stream* that is associated with an *identifier*, since a *part* is one type of *member*.

2.3.2.2 Conformance points

PKG 6 A *part* shall conform to the conformance points for a *member* (section 2.3.3.3.1).

PKG 7 A *part* shall contain exactly one *byte stream* that is an ordered sequence of zero or more octets.

2.3.3 Members and identifiers

2.3.3.1 Introduction

There are two types of *members*: *parts* and *referenced packages*.

Members are contained in a *package* and are uniquely identified within that *package* by its *identifier*.

2.3.3.2 Definition

Two *identifiers* are treated as strings and they are identical if and only if the strings are identical, that is, they are the same sequence of characters.

2.3.3.3 Conformance points

2.3.3.3.1 Members

PKG 8 A *member* shall be associated with exactly one *identifier* that conforms to the conformance points for *identifiers* (section 2.3.3.3.2).

2.3.3.3.2 Identifiers

PKG 9 *Identifiers* shall be a non-empty value that conform to the `URI-reference` production from "Uniform Resource Identifier (URI): Generic Syntax" [RFC3986].

PKG 10 *Identifiers* shall be unique to the scope of the *package* that the *member* it identifies is contained in.

2.3.3.4 Informative notes

2.3.3.4.1 URI-reference

The `URI-reference` production in [RFC3986] is defined as either a URI (sometimes referred to as an absolute URI) or a relative reference (sometimes incorrectly referred to as a relative URI).

As specified in [RFC3986] a relative reference can be an empty value (containing zero characters). The empty value is not useful for identifying more than one item, so the empty value has been explicitly prohibited as an *identifier* value.

2.3.3.4.2 Comparison of identifiers

Comparison of *identifiers* is case-sensitive and no %-escaping is done or undone.

This definition of URI comparison is consistent with the definition used by XML namespaces as defined in section 2.3 of [XNS2009]. It can be different from URI equivalence which is discussed in section 6 of [RFC3986].

2.3.3.4.3 Identifier semantics

From the *logical clinical package* point of view, *identifiers* are treated as opaque values. Other than associating them with a *member* and the operation of comparing two *identifiers* for equivalence, the *logical clinical package* does not associate any other meaning to their values.

These *identifiers* can be used inside the *byte stream* for them to reference *members* of the same *package*. Some representations (e.g. the CP-ZIP representation) use these *identifiers* to represent the *member* that is associated with a *distinguisher*.

2.3.3.4.4 Uniqueness of identifiers

Identifiers only need to be unique amongst the *identifiers* in the same *package* they are contained in—they do not need to be unique across different *packages*.

2.3.3.4.5 Use of UUID for identifiers

If the source for the *members* has not already allocated values for *identifiers*, it is best practice to generate UUID values for *identifiers*. *Identifiers* are only required to be unique within a *package*, but using a globally unique UUID value can have additional benefits beyond *clinical packages*.

It is also best practice for UUID based *identifiers* to not have additional information, since that information cannot be expressed in formats that only support the value of a single UUID. For example, if the *identifier* was "`bbd1d1f2-fdd4-43cc-8951-3b35cececd95`" it can be represented as the URN UUID "`urn:uuid:bbd1d1f2-fdd4-43cc-8951-3b35cececd95`" but if the *identifier* was "`bbd1d1f2-fdd4-43cc-8951-3b35cececd95.xml`" it cannot be represented as a URN UUID without loss of information. The URN UUID is defined in [RFC4122].

2.3.4 Distinguishers and distinguisher types

2.3.4.1 Introduction

Distinguishers are used to tell a *member* apart from the other members of that *package*.

They are typically used to help *package* processors find the *part* to start processing with. For example, there can be *distinguishers* to help find the root CDA XML document or to help find the digital signature. There can be multiple

distinguishers in a *package*: a *distinguisher type* can be associated with more than one *distinguisher* instance; a *member* can be associated with more than one *distinguisher* instance.

2.3.4.2 Definition

Two *distinguisher types* are treated as strings and they are identical if and only if the strings are identical, that is, they are the same sequence of characters.

2.3.4.3 Conformance points

PKG 11 A *distinguisher* shall be associated with exactly one *distinguisher type* that is a Uniform Resource Identifier (URI) as defined by [RFC3986].

PKG 12 A *distinguisher* shall be associated with exactly one *member* from the *package* that the *distinguisher* is contained in.

2.3.4.4 Informative notes

2.3.4.4.1 URI

The *distinguisher type* is a URI. Unlike *identifiers*, which are defined as the `URI-reference` production, a *relative reference* (sometimes incorrectly known as a "relative URI") is not permitted as a *distinguisher type*. *Distinguisher types* must always be absolute URIs.

2.3.4.4.2 Comparison of distinguisher types

Comparison of *distinguisher types* is case-sensitive and no %-escaping is done or undone.

This definition of URI comparison is consistent with the definition used by XML namespaces as defined in section 2.3 of [XNS2009]. It can be different from URI equivalence which is discussed in section 6 of [RFC3986].

2.3.4.4.3 Cardinality

A *package* can have multiple *distinguishers*, because different *byte streams* (or *referenced packages*) might be used for different purposes. For example, there might be a digital signature part that integrity checkers process and the clinical data part that clinical software processes.

The *logical clinical package* permits multiple *distinguishers* with the same *distinguisher type*, but it is expected that use of *distinguishers* in this way would be rare.

2.3.4.4.4 Dependability of values

Distinguishers are usually used as hints for *package* processors: to identify a *part* (or in rare cases a *referenced package*) to start processing. They are hints since they could be confirmed by examining the actual *byte streams*. For example, if a *distinguisher* distinguishes a part as the clinical data, examining the *byte stream* of that *part* could prove whether that hint was correct or not.

It is best practice to only use *distinguishers* for non-critical purposes. Processors need to treat *distinguishers* as processing hints, and only rely on data that is confirmed by examining the *byte stream*. This is because the integrity of *distinguishers* might not be determinable—data in *distinguishers* might not be digitally signed. If data integrity is important, store that data in a *byte stream*.

3 CP-ZIP

3.1 Overview

This chapter specifies the CP-ZIP representation, which is a serialized syntax for representing instances of the *logical clinical package* defined in chapter 2.

Informally, a CP-ZIP instance is a *ZIP archive* where each *byte stream* is stored as a *ZIP item* and there is an extra *package index zip item* corresponding to each *package*. The *package index* stores the mapping between *identifiers* and *ZIP items* in the *ZIP archive* as well as any *distinguishers* in its *package*.

3.1.1 Concepts

3.1.1.1 ZIP archive, ZIP item and ZIP item names

The CP-ZIP representation (short for "Clinical Package ZIP") is a *ZIP archive* format with the additional conventions specified in this chapter.

For the purposes of this specification, a *ZIP archive* is considered to be a format that represents an unordered set of *ZIP items*, with each *ZIP item* associated with a *ZIP item name* that is unique to the scope of the entire *ZIP archive*.

This specification does not rely on other features that the *ZIP archive* format can support. For example, the *ZIP items* are ordered in a *ZIP archive*, but this ordering is not significant to the CP-ZIP representation.

3.1.1.2 Printable US-ASCII equivalent value

These conformance points make reference to the "printable US-ASCII equivalent value" of various attribute values. The US-ASCII character set is defined by [ANSI-X3.4-1968]. Printable US-ASCII characters are those in the range of %x20 to %x7E, inclusive.

There is no equivalent printable US-ASCII value for Unicode strings that contain Unicode code points outside that range, so those Unicode strings cannot be used in those attributes. Attributes which are required to have a printable US-ASCII equivalent value can only contain Unicode code points from the range of printable US-ASCII characters (i.e. %x20 to %x7E).

Note: These conformance points are designed to ensure that all *ZIP item names* contain only printable US-ASCII characters. Newer ZIP formats have extensions that support Unicode *ZIP item names*, but the version of the *ZIP archive* format referenced by this specification does not support Unicode *ZIP item names* [PK2004].

3.1.1.3 Package prefix

A *package prefix* is an ordered sequence of zero or more printable US-ASCII characters. It is used to determine *ZIP item names* according to the conformance points in this chapter.

A *package prefix* is implicitly associated with every *package index*. It does not explicitly appear in the CP-ZIP representation, but is built up from data found in it. Namely, by concatenating together the values in the `base` attribute (section 3.2.3.2.3).

Package prefixes are typically chosen to mimic subdirectories under which all the contents of the *package* are located.

Note: *Package prefixes* do not exist in the *logical clinical package*. The *package prefix* is a feature of the CP-ZIP representation, and is not present in other representations. Therefore, their values are not visible to the *logical clinical package* and they do not have to be preserved when only the *logical clinical package* is stored.

3.2 Representation

3.2.1 CP-ZIP

3.2.1.1 Introduction

An instance of the CP-ZIP represents exactly one *root package*.

An instance of CP-ZIP does not represent any *referenced packages* independent of a *root package*, since in the model of a *logical clinical package* the *referenced packages* do not exist independently of a *root package*.

3.2.1.2 Conformance points

PKG 13 An instance of CP-ZIP shall be a single *ZIP archive* that conforms to the specification for the *ZIP archive* format in [PK2004].

PKG 14 An instance of the CP-ZIP shall not use features of a *ZIP archive* which are not portable, including (but not limited to) "Traditional PKWARE Encryption" and "Strong Encryption Specification" from the *ZIP archive* specification [PK2004].

PKG 15 An instance of CP-ZIP shall contain exactly one *root package index* that conforms to the conformance points for a *root package index* in section 3.2.2.2.

3.2.1.3 Informative notes

3.2.1.3.1 ZIP archive

ZIP archives are commonly used as a ZIP file that stores files and directories in a compressed form, but there is no requirement for an instance of CP-ZIP to be used in this manner.

A CP-ZIP instance can be stored as a file, but do not need to.

The *ZIP items* in the CP-ZIP instance can correspond to files and directories, but does not need to.

The *ZIP items* can be compressed, but does not need to. Whether compression is used or not, the logical *byte stream* is unaffected—compression is a feature of the representation, not of the logical model.

3.2.1.3.2 Other ZIP items in the ZIP archive

These conformance points do not prohibit the presence of other *ZIP items* in the *ZIP archive: ZIP items* which are not specified as being part of CP-ZIP instance.

Those other *ZIP items* are not a part of the *logical clinical package*. Therefore, a compliant program that claims to process or store *clinical packages* can ignore or not store those ZIP items which are not a part of the *logical clinical package*.

It is best practice to permit other *ZIP items* to appear in the *ZIP archive*. Processors of the *CP-ZIP* will need to expect the possible presence of those other *ZIP items* in the *ZIP archive*, but what it does with them is implementation dependent.

3.2.2 Root package index

3.2.2.1 Introduction

A *root package index* is a *package index* (section 3.2.2) that corresponds to the *root package* being represented by the CP-ZIP instance.

3.2.2.2 Conformance points

PKG 16 The *root package index* shall be stored in a *ZIP item* with the *ZIP item name,* whose value is the US-ASCII string "`META-INF/PKGINDEX.XML`".

PKG 17 The *root package index* shall conform to the conformance points for a *package index* in section 3.2.3.2.

PKG 18 The *root package index* shall be associated with a *package prefix* whose value is the zero-length US-ASCII string.

3.2.3 Package index

3.2.3.1 Introduction

A *package index* is used to represent a *package* (either the *root package* or any *referenced packages*) and is used to:

- Identify all the *ZIP items* corresponding to *parts* contained in the *package*;
- Identify all the *ZIP items* corresponding to the *package index* for all *referenced packages* contained in the *package*.
- Represent all the *distinguishers* contained in the *package*.

All *package indexes* are associated with a *package prefix*: a string value that is used to interpret some of the values in the *package index*.

3.2.3.2 Conformance points

3.2.3.2.1 Schema

PKG 19 A *package index* shall be an XML document that is schema valid against the *Package Index XML Schema* in Appendix A.1.

3.2.3.2.2 Parts

PKG 20 A *package index* shall have exactly one corresponding `pi:part` element for each *part* contained in the corresponding *package* and no other `pi:part` elements.

PKG 21 A `pi:part` element shall have an `id` attribute whose value is the corresponding *part's identifier*.

PKG 22 When the `item` attribute is present on the `pi:part` element, its value shall not be a zero-length string.

PKG 23 If a `pi:part` element does not have an `item` attribute, the *atomic item* for the *part* shall be stored in a *ZIP item* with the *ZIP item name* that is the concatenation of:

- the *package prefix,* and
- the printable US-ASCII equivalent value of the `id` attribute.

PKG 24 If a `pi:part` element has an `item` attribute, the *atomic item* for the *part* shall be stored in a *ZIP item* with the *ZIP item name* that is the concatenation of:

- the *package prefix,* and
- the printable US-ASCII equivalent value of the `item` attribute.

3.2.3.2.3 Referenced packages

PKG 25 A *package index* shall have exactly one `pi:package` element corresponding to each *referenced package* in the *package* and no other `pi:package` elements.

PKG 26 A `pi:package` element shall have an `id` attribute whose value is the *identifier* of the corresponding *referenced package*.

PKG 27 If the `item` attribute on the `pi:package` element is present, its value shall not be the zero-length string.

PKG 28 If a `pi:package` element does not have an `item` attribute, the *ZIP item* containing the *package index* shall have a *ZIP item name* that is the concatenation of:

- the current *package prefix*,
- the printable US-ASCII equivalent value of the `base` attribute, and
- the printable US-ASCII string "`META-INF/PKGINDEX.XML`".

PKG 29 If a `pi:package` element has an `item` attribute, the *ZIP item* containing the *package index* shall have a *ZIP item name* that is the concatenation of:

- the current *package prefix*,
- the printable US-ASCII equivalent value of the `base` attribute, and
- the printable US-ASCII equivalent value of the `item` attribute.

PKG 30 The *package prefix* associated with the *referenced package's package index* shall be the concatenation of:

- the current *package prefix;* and
- the printable US-ASCII equivalent value of the `base` attribute.

3.2.3.2.4 Distinguishers

PKG 31 A *package index* shall contain exactly one `pi:distinguisher` element corresponding to each *distinguisher* in the *package* and no other `pi:distinguisher` elements.

PKG 32 A `pi:distinguisher` element shall have the corresponding *distinguisher's distinguisher type* as the value of its `type` attribute.

PKG 33 A `pi:distinguisher` element shall have the *identifier* of the corresponding *distinguisher's member* as the value of its `member` attribute.

3.2.3.3 Informative notes

3.2.3.3.1 CP-ZIP feature

A *package index* (and a *root package index*) is an artefact of the *CP-ZIP* representation. It does not exist as a concept in the logical package. Therefore, other representations might not have any *package index*.

3.2.3.3.2 Item

The `item` attribute, on the `pi:part` and `pi:package` elements, allows the *ZIP item name* to be different from the default. For a *part*, the default results in a *ZIP item name* that ends in the *part's identifier*. For a *referenced package*, the default results in a *ZIP item name* that ends in the value "META-INF/PKGINDEX.XML".

The use of the `item` attribute is entirely optional.

CP-ZIP creators are encouraged to use the `item` attribute to help ensure that *ZIP item names* mimic file and directory names, since *identifier* values are not always suitable for interpretation as file or directory names.

3.2.3.3.3 ZIP item containing the package index

If the corresponding *package* is a *root package*, the *package index* will be stored in a *ZIP item* with the *ZIP item name* of "META-INF/PKGINDEX.XML" as specified in section 3.2.2.2.

If the corresponding *package* is a *referenced package*, the *package index* will be stored in a *ZIP item* whose *ZIP item name* is indicated by the *package index* for the *package* that the *referenced package* is contained in (either explicitly with the `item` attribute or implicitly by the absence of that attribute).

The `item` attribute allows the *ZIP item name* to be different from the *identifier*. It must be used when the *identifier* is not a printable US-ASCII string, but can be used in other situations too. For example, it is permitted for the `item` attribute to be always used (even when it does not have to be used).

It is best practice to choose `item` attribute values that mimics file names. This makes it easier for developers to understand and to debug the contents of the *ZIP archive*.

The `base` attribute allows *ZIP items* relating to the same *referenced package* to be grouped together. It is best practice to choose `base` values that mimic directory names (e.g. by using a value that has only one slash character at the end).

Although the `base` attribute is mandatory, it can contain a zero-length string.

Referenced *package index* example 1:

If a *package index* is associated with a *package prefix* of "abc/" and it contains the following element:

```
<pi:package base="def/" id="…"/>
```

The *referenced package's package index* is stored in a *ZIP item* with the *ZIP item name* of "abc/def/META-INF/PKGINDEX.XML" and it is associated with *package prefix* of "abc/def/". Note that the `id` attribute is not used in determining the value of that *ZIP item name* or its *package prefix*.

In these examples, the value of the *identifier* is not significant to point being illustrated and an ellipsis is used as a placeholder.

Referenced *package index* example 2:

If a *package index* is associated with a *package prefix* of "abc/" and it contains the following element:

```
<pi:package base="uvw/" item="xyz" id="…"/>
```

The *referenced package's package index* is stored in a *ZIP item* with the *ZIP item name* of "abc/uvw/xyz" and it is associated with the *package prefix* of "abc/uvw/". Note that the `id` attribute is not used in determining the value of that *ZIP item name* or that *package prefix*.

3.2.3.3.4 ZIP item containing atomic items

Atomic item example 1:

If a *package index* is associated with a *package prefix* of "abc/" and it contains the following element:

```
<pi:part id="foo"/>
```

The *ZIP item name* for the *atomic item* is "abc/foo".

Atomic item example 2:

If a *package index* is associated with a *package prefix* of "`abc/`" and it contains the following element:

```
<pi:part id="foo" item="bar"/>
```

The *ZIP item name* for the *atomic item* is "`abc/bar`".

Atomic item example 3:

The `item` attribute can be used to make the *ZIP item name* to be not based on the *identifier*. For example, if a *package index* is associated with a *package prefix* of "`abc/`" and it contains the following element:

```
<pi:part id="identifierWithWeirdChars" item="part0001.dat"/>
```

The *ZIP item name* for the *atomic item* is "`abc/part0001.dat`".

3.2.3.3.5 Package prefix

CP-ZIP instance creators can use the *package prefix* to group together *ZIP item names* for the contents of the same *referenced package*. For example, to make *packages* appear to be stored in sub-directories.

There is no obligation to use a non-zero length *package prefix*. A *CP-ZIP* can use the zero length string as the *package prefix*. The *package prefix* for the *root package index* is always the zero length string. The *package prefix* remains the zero length string if the value of the `base` attribute is the zero length string.

Package prefixes can also be used to ensure that *ZIP item names* are unique, even though there could be a conflict with *identifiers*. This conflict can occur because *identifiers* are unique to the scope of a single *package*, but not necessarily unique across multiple *packages*. There are multiple *packages* in a CP-ZIP instance when there are *referenced packages*.

3.2.3.3.6 Distinguishers

The *distinguishers* are represented using the `pi:distinguisher` element. For example,

```
<pi:distinguisher type="http://ns.example.com/foo" member="bar"/>
```

3.2.3.3.7 Ordering

The order of multiple `pi:part` elements in the *package index* is not significant, since the there is no ordering of *parts* in the logical mode.

The order of multiple `pi:package` elements in the *package index* is not significant, since the there is no ordering of *referenced packages* in the logical mode.

The order of multiple `pi:distinguisher` elements in the *package index* is not significant, since the there is no ordering of *distinguishers* in the logical mode.

3.2.4 Atomic item

3.2.4.1 Introduction

Atomic items in CP-ZIP correspond to the *byte streams* in the logical model.

3.2.4.2 Conformance points

PKG 34 A *atomic item* shall be a *ZIP item* that contains the *byte stream* of the *part*.

3.2.4.3 Informative notes

The *atomic item* will be stored in a *ZIP item* whose *ZIP item name* is indicated by the *package index* for the *package* that the *part* is contained in.

Appendix A: XML Schemas

(Normative)

A.1 Package index XML Schema

```
<?xml version="1.0"?>

<xsd:schema
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:tns="http://ns.electronichealth.net.au/pkg/xsd/PackageIndex/1.0"
 targetNamespace="http://ns.electronichealth.net.au/pkg/xsd/PackageIndex/1.0"
 elementFormDefault="qualified">

  <xsd:element name="packageIndex" type="tns:PackageIndexType"/>

  <xsd:complexType name="PackageIndexType">
    <xsd:sequence>
      <xsd:element name="part" type="tns:PartType"
       minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="package" type="tns:ReferencedPackageType"
       minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="distinguisher" type="tns:DistinguisherType"
       minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="PartType">
    <xsd:attribute name="id" type="xsd:anyURI" use="required"/>
    <xsd:attribute name="item" type="xsd:string" use="optional"/>
  </xsd:complexType>

  <xsd:complexType name="ReferencedPackageType">
    <xsd:attribute name="id" type="xsd:anyURI" use="required"/>
    <xsd:attribute name="base" type="xsd:string" use="required"/>
    <xsd:attribute name="item" type="xsd:string" use="optional"/>
  </xsd:complexType>

  <xsd:complexType name="DistinguisherType">
    <xsd:attribute name="type" type="xsd:anyURI" use="required"/>
    <xsd:attribute name="member" type="xsd:anyURI" use="required"/>
  </xsd:complexType>

</xsd:schema>
```

Appendix B: Clinical package profile

(Informative)

B.1 Introduction

Clinical packages define a logical model, but do not define the data that is expected to populate that model. The definition of that data is called a *clinical package profile*.

Clinical package profiles specify the additional information needed to fully define a particular type of *clinical package*.

B.2 Contents

B.2.1 Logical model

A *clinical package profile* needs to define:

- Distinguishers:
  - Distinguisher type (a URI).
  - Cardinality.
- Parts:
  - Contents.
  - Cardinality.
- Referenced packages:
  - Contents (i.e. clinical package profile)
  - Cardinality.

Cardinality in the above list means how often the item can or must appear. For items that are not permitted to repeat, this can simply be whether the item is optional or mandatory. For items that can repeat, it needs to indicate how many times the item can be repeated.

Clinical package profiles typically also define the semantics of the data in the model.

B.2.2 Representation

The CP-ZIP representation is the default choice for representing a clinical package for interchange.

Some *clinical package profiles* might define alternative representations for the *clinical package*.

Appendix C: Examples

(Informative)

C.1 Example with parts

C.1.1 Introduction

This example shows a CP-ZIP instance of a *clinical package* that does not contain any *referenced packages*; it contains only *parts*.

It demonstrates that *package prefixes* and the `item` attributes do not have to be used unless the *CP-ZIP* creator needs to or wants to use them.

C.1.2 Clinical package profile

This example is an instance of eg1. The *clinical package profile* for eg1 is:

- An eg1 shall conform to a *logical clinical package*.
- Parts:
  - An eg1 shall contain exactly one *part* that is called the root CDA.
  - The contents of the root CDA shall be an XML document that conforms to the CDA specification.
  - If the root CDA has packaged attachments, they shall either be a *part* or a *referenced package* in the *logical clinical package*. That is, an eg1 shall contain zero or more *parts*, and zero or more *referenced packages* as determined by the contents of the root CDA.
- Distinguishers:
  - An eg1 shall have exactly one *distinguisher* with the *distinguisher type* of "`http://ns.example.com/eg1/root`".
  - The member associated with that *distinguisher* shall be the root CDA *part*.

C.1.3 Logical contents

This example is an instance of eg1 that contains:

- One CDA XML document; and
- One packaged attachment that is an image.

C.1.4 CP-ZIP instance

This example can be represented by a *ZIP archive* containing three *ZIP items*:

- `META-INF/PKGINDEX.XML` – the *root package index*
- `cda.xml` – *atomic item*
- `image001.jpg` – *atomic item*

These *ZIP items* are illustrated in Figure 2 as solid rectangles and the *package* as a dashed rectangle.

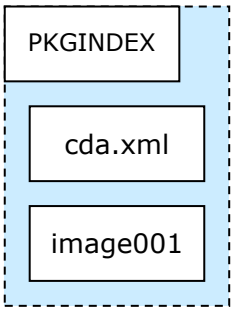

Figure 2 ZIP items in example 1

C.1.4.1 Root package index

This *root package index* is stored in a *ZIP item* with the *ZIP item name* of "META-INF/PKGINDEX.XML". This *ZIP item name* is fixed by the *clinical packages* specification. This *root package index* contains the following XML:

```
<?xml version="1.0"?>
<packageIndex
 xmlns="http://ns.electronichealth.net.au/pkg/xsd/PackageIndex/1.0">

  <part id="cda.xml"/>
  <part id="image001.jpg"/>
  <distinguisher type="http://ns.example.com/eg1/root" member="cda.xml"/>

</packageIndex>
```

C.1.4.2 CDA XML document

The *identifier* for this *part* is "cda.xml".

The *ZIP item name* for it is "cda.xml". This *ZIP item name* can be obtained by concatenating the *package prefix* associated with the containing *package index* (i.e. the zero length string, since it is the *root package index*) with the printable US-ASCII equivalent value of the id attribute (i.e. "cda.xml").

A processor expecting to process a CDA XML document can find this *part* by finding the *distinguisher* with the *distinguisher type* of http://eg.example.com/eg1/root.

Inside the CDA XML document, it can reference the image using the image's *identifier*. Note: this is referencing the *identifier* and not the *ZIP item name* (though they appear to be the same value, actually the *identifier* is logically a URI-reference and the *ZIP item name* is physically a printable US-ASCII string).

```
<?xml version="1.0"?>
<ClinicalDocument xmlns="urn:hl7-org:v3">
 …
 <value mediaType="image/jpeg"
  integrityCheckAlgorithm="SHA-256"
  integrityCheck="wGR+V5snZvnTm39zMkNZfFzV9QzjS44RaJj28SH+EkM=">
  <reference value="image001.jpg"/>
 </value>
 …
</ClinicalDocument>
```

C.1.4.3 Image

The *identifier* for this *part* is "image001.jpg".

The *ZIP item name* for it is "image001.jpg". This *ZIP item name* is the result of concatenating the *package prefix* associated with the containing *package index* (i.e. the zero length string, since it is the *root package index*) with the printable US-ASCII equivalent value of the id attribute (i.e. "image001.jpg" in this case).

C.2 Example with referenced package

C.2.1 Introduction

This example shows a CP-ZIP instance of a *clinical package* that contains a *referenced package*.

It also demonstrates the use of the `base` and `item` attributes.

C.2.2 Clinical package profile

This example is an instance of eg2. The *clinical package profile* for eg2 is:

- An eg2 shall conform to a *logical clinical package*.
- Parts:
  - An eg2 shall contain exactly one *part* that is called the root CDA.
  - The contents of the root CDA shall be an XML document that conforms to the CDA specification.
  - An eg2 shall not contain any *parts* other than the root CDA *part*.
  - If the root CDA has packaged attachments, they shall each be a referenced package. That is, an eg2 shall contain zero or more *referenced packages* as determined by the contents of the root CDA. These *referenced packages* will be called *reports*.
  - A *report* shall be a *clinical package* that conforms to an eg1.
- Distinguishers:
  - An eg2 shall have exactly one *distinguisher* with the *distinguisher type* of "`http://ns.example.com/eg2/root`".
  - The member associated with that *distinguisher* shall be the root CDA *part*.

C.2.3 Logical contents

This example is an instance of eg2 that contains:

- One CDA XML document; and
- One packaged attachment that itself contains:
  - One CDA XML document; and
  - One packaged attachment image.

The packaged attachment used is the *clinical package* from the previous example.

C.2.4 CP-ZIP instance

This example can be represented by a ZIP archive containing five *ZIP items*:

- `META-INF/PKGINDEX.XML` – the *root package index*
- `c3a8027b-440f-49b3-9f38-37f08376f926` – *atomic item*
- `a19605b5-6c76-4608-9046-86c417f1e43c/META-INF/PKGINDEX.XML`
- `a19605b5-6c76-4608-9046-86c417f1e43c/cda.xml` – *atomic item*
- `a19605b5-6c76-4608-9046-86c417f1e43c/image001.jpg` – *atomic item*

These *ZIP items* are illustrated in Figure 2 as solid rectangles and the *package* as a dashed rectangle.

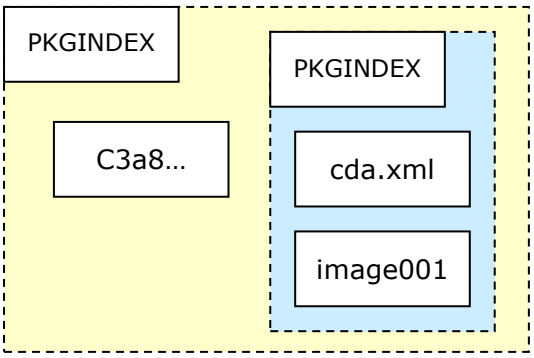

Figure 3 ZIP items in example 2

C.2.4.1 Root package index

This *root package index* is stored in a *ZIP item* with the *ZIP item name* of "`META-INF/PKGINDEX.XML`". This *ZIP item name* is fixed by the *clinical packages* specification. This *root package index* contains the following XML:

```
<?xml version="1.0"?>
<packageIndex
 xmlns="http://ns.electronichealth.net.au/pkg/xsd/PackageIndex/1.0">

  <part id="DischargeSummary" item="c3a8027b-440f-49b3-9f38-37f08376f926"/>

  <package id="PathologyReport" base="a19605b5-6c76-4608-9046-86c417f1e43c"/>

  <distinguisher type="http://ns.example.com/eg2/root"
                 member="DischargeSummary"/>
</packageIndex>
```

C.2.4.2 CDA XML document

The *identifier* for this *part* is "DischargeSummary".

The *ZIP item name* for it is "`c3a8027b-440f-49b3-9f38-37f08376f926`". This *ZIP item name* can be obtained by concatenating the *package prefix* associated with the containing *package index* (i.e. the zero length string, since it is the *root package index*) with the printable US-ASCII equivalent value of the `item` attribute (i.e. "`c3a8027b-440f-49b3-9f38-37f08376f926`").

In this example, the `item` attribute's value used is a randomly generated UUID. This value has no effect on the logical model of the *clinical package*, because the `item` attribute is only an artefact of the CP-ZIP representation.

A processor expecting to process a CDA XML document can identify this *part* by finding a *distinguisher* with the *distinguisher type* of `http://eg.example.com/eg2/root`.

Inside this CDA XML document, it can reference the *packaged attachment* using the *identifier* of the *referenced package* containing that *package attachment*.

```
<?xml version="1.0"?>
<ClinicalDocument xmlns="urn:hl7-org:v3">
 …
 <value mediaType="application/vnd.cda.package"
  integrityCheckAlgorithm="SHA-256"
  integrityCheck="XvOtfOutzzDDAS5tPdWhFsfkOROYqK/KxcxtVa8fHrw=">
  <reference value="PathologyReport"/>
 </value>
 …
</ClinicalDocument>
```

C.2.4.3 Attachment

The *identifier* of the *packaged attachment* is "PathologyReport".

The *packaged attachment* is a *referenced package* made up of a CDA XML document and an image file. For this example, it is made up of the same three *parts* from the previous example (Appendix C.1), but stored with different *ZIP item names*:

- `a19605b5-6c76-4608-9046-86c417f1e43c/META-INF/PKGINDEX.XML`
- `a19605b5-6c76-4608-9046-86c417f1e43c/cda.xml` – *atomic item*
- `a19605b5-6c76-4608-9046-86c417f1e43c/image001.jpg` – *atomic item*

In this example, the `base` attribute's value used is a randomly generated UUID. This value has no effect on the logical model of the *clinical package*; the `base` attribute is only an artefact of the CP-ZIP representation.

C.3 Example with parts and digital signatures

C.3.1 Introduction

This example shows a CP-ZIP instance of a *clinical package* that contains a digital signature.

C.3.2 Clinical package profile

This example is an instance of eg3. The *clinical package profile* for eg3 is:

- An eg3 shall conform to a logical clinical package.
- Parts:
  - An eg3 shall contain exactly one *part* that is called the root CDA.
  - The contents of the root CDA shall be an XML document that conforms to the CDA specification.
  - An eg3 shall contain exactly one part that is called the digital signature.
  - The contents of the digital signature shall be an XML document that conforms to (detailed syntax that is not relevant to this example) and shall only sign the root CDA.
  - If the root CDA has attachments, they shall each a *part* in the logical clinical package. That is, an eg3 shall contain zero or more *parts* (in addition to the root CDA and digital signature) as determined by the contents of the root CDA.
  - An eg3 shall not contain any *referenced packages*.
- Distinguishers:
  - An eg3 shall have exactly one *distinguisher* with the *distinguisher type* of "`http://ns.example.com/eg3/root`".
  - The member associated with that *distinguisher* shall be the root CDA *part*.

C.3.3 Logical contents

This example is an instance of eg3 that contains:

- One CDA XML document;
- One image attachment; and
- One digital signature.

C.3.4 CP-ZIP instance

This example can be represented by a *ZIP archive* containing four *ZIP items*:

- `META-INF/PKGINDEX.XML` – the *root package index*
- `1339429a-e8c6-4bfa-9ebd-599d9a7e5191`– *atomic item (CDA root)*
- `260d5e6e-a28f-4a0d-a86a-bed6a611f877`– *atomic item (image)*
- `d87e61d7-69ca-4ae8-a408-8eec513449ba` – *atomic item (signature)*

These *ZIP items* are illustrated in Figure 5 as solid rectangles and the *package* as a dashed rectangle.

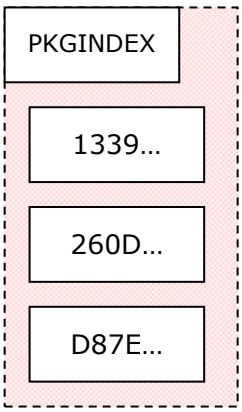

Figure 4 ZIP items in example 3

C.3.4.1 Package index

The *package index* for the first *referenced package* is stored in a *ZIP item* with the *ZIP item* name of "META-INF/PKGINDEX.XML". This *ZIP item name* is fixed by the *clinical packages* specification.

This *root package index* contains the following XML:

```
<?xml version="1.0" encoding="UTF-16"?>
<packageIndex
 xmlns="http://ns.electronichealth.net.au/pkg/xsd/PackageIndex/1.0">

  <part id="1339429a-e8c6-4bfa-9ebd-599d9a7e5191"/>
  <part id="image0001" item="260d5e6e-a28f-4a0d-a86a-bed6a611f877"/>
  <part id="DigitalSignature" item="d87e61d7-69ca-4ae8-a408-8eec513449ba"/>

  <distinguisher type="http://ns.example.com/eg3/root"
                 member="1339429a-e8c6-4bfa-9ebd-599d9a7e5191"/>
  <distinguisher type="http://ns.example.com/eg3/dsig"
                 member="DigitalSignature"/>

</packageIndex>
```

C.3.4.2 CDA XML document

The *identifier* for this *part* is "1339429a-e8c6-4bfa-9ebd-599d9a7e5191".

This CDA XML document is stored in a *ZIP item* with the *ZIP item name* of "1339429a-e8c6-4bfa-9ebd-599d9a7e5191". This *ZIP item name* can be obtained by concatenating the *package prefix* associated with the containing *package index* (i.e. the zero length string, since it is the *root package index*) with the value of the `item` attribute (i.e. "1339429a-e8c6-4bfa-9ebd-599d9a7e5191").

A processor expecting to process a CDA XML document can identify this *part* by finding the *distinguisher* with *distinguisher type* of `http://eg.example.com/eg3/root` the `pi:part` element with a `id` attribute with the same value.

Inside this CDA XML document, it references the image using its *identifier*.

```
<?xml version="1.0" encoding="UTF-8"?>
<ClinicalDocument xmlns="urn:hl7-org:v3">
 …
 <value mediaType="image/jpeg"
  integrityCheckAlgorithm="SHA-256"
  integrityCheck="wGR+V5snZvnTm39zMkNZfFzV9QzjS44RaJj28SH+EkM=">
  <reference value="image0001"/>
 </value>
 …
</ClinicalDocument>
```

C.3.4.3 Image

The *identifier* for this *part* is "`image0001`".

This image is stored in a *ZIP item* with the *ZIP item name* of "`260d5e6e-a28f-4a0d-a86a-bed6a611f877`". This *ZIP item name* can be obtained by concatenating the *package prefix* associated with the containing *package index* (i.e. the zero length string, since it is the *root package index*) with the value of the `item` attribute (i.e. "`260d5e6e-a28f-4a0d-a86a-bed6a611f877`").

C.3.4.4 Digital signature

The *identifier* for this *part* is "DigitalSignature".

This digital signature is stored in a *ZIP item* with the *ZIP item name* of "`d87e61d7-69ca-4ae8-a408-8eec513449ba`". This *ZIP item name* can be obtained by concatenating the *package prefix* associated with the containing *package index* (i.e. the zero length string, since it is the *root package index*) with the value of the `item` attribute (i.e. "`d87e61d7-69ca-4ae8-a408-8eec513449ba`" in this case).

A processor expecting to verify the digital signature of the first attachment *package* can identify this *part* by finding the *distinguisher* with *distinguisher type* of `http://eg.example.com/eg3/dsig` and finding the `pi:part` element with a `id` attribute that has the same value.

Inside this digital signature, it references the *parts* that it signs. This example assumes the hashed reference mechanism in CDA is used to ensure the integrity of the attachments referenced by the CDA XML document. Therefore, only the CDA XML document is digitally signed (i.e. the digital signature only references the *identifier* "1339429a-e8c6-4bfa-9ebd-599d9a7e5191").

```
<?xml version="1.0" encoding="UTF-16"?>

<sp:signedPayload
 xmlns:sp="http://ns.electronichealth.net.au/xsp/xsd/SignedPayload/2010"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
 <sp:signatures>
 …
 </sp:signatures>

 <sp:signedPayloadData id="x">
  <ds:Manifest>
   <ds:Reference URI="1339429a-e8c6-4bfa-9ebd-599d9a7e5191">
    <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <ds:DigestValue>D1qrG2xg+FOYRPb71NPUVwAWisM=</ds:DigestValue>
   </ds:Reference>
  </ds:Manifest>
 </sp:signedPayloadData>

</sp:signedPayload>
```

C.4 Example with referenced packages and digital signatures

C.4.1 Introduction

This example shows a CP-ZIP instance of a *clinical package* with a referenced package that contains digital signatures.

C.4.2 Clinical package profile

This example is an instance of eg4. The *clinical package profile* for eg4 is:

- An eg4 shall conform to a *logical clinical package*.
- Parts:
  - An eg4 shall contain exactly one *part* that is called the root CDA.
  - The contents of the root CDA shall be an XML document that conforms to the CDA specification.
  - An eg3 shall contain exactly one part that is called the digital signature.
  - The contents of the digital signature shall be an XML document that conforms to (detailed syntax that is not relevant to this example) and shall only sign the root CDA.
  - If the root CDA has attachments, they shall be a *part* or a *referenced package* in the *logical clinical package*. That is, an eg4 shall contain zero or more *parts*, and zero or more *referenced packages* (in addition to the root CDA and digital signature) as determined by the contents of the root CDA.
- Distinguishers:
  - An eg4 shall have exactly one *distinguisher* with the *distinguisher type* of "`http://ns.example.com/eg4/root`".
  - The member associated with that *distinguisher* shall be the root CDA *part*.

C.4.3 Logical contents

This example is an instance of eg4 that contains:

- One CDA XML document;
- One attachment image;
- One attachment that itself contains:
  - One CDA XML document;
  - One attachment image; and
  - One digital signature.
- One digital signature.

The attachment used is the *clinical package* from the previous example (Appendix C.3).

C.4.4 CP-ZIP instance

This example can be represented by a ZIP archive containing eight ZIP items:

- `META-INF/PKGINDEX.XML` – the *root package index*
- `cda80e3f-17d4-447e-a703-63aaf642ac36` – *atomic item* in *root*
- `389b2987-5ad5-4b9d-be0a-781112830777` – *atomic item* in *root*
- `d91c5799-ddfb-4aaf-af50-44d82b3ba809` – *atomic item* in *root*
- `attach01/a2380e3f-17d4-447e-a703-63aaf642ac36` – *package index*
- `attach01/1339429a-e8c6-4bfa-9ebd-599d9a7e5191` – *atomic item*
- `attach01/260d5e6e-a28f-4a0d-a86a-bed6a611f877` – *atomic item*
- `attach01/d87e61d7-69ca-4ae8-a408-8eec513449ba` – *atomic item*

These *ZIP items* are illustrated in Figure 5 as solid rectangles and the *packages* as dashed rectangles.

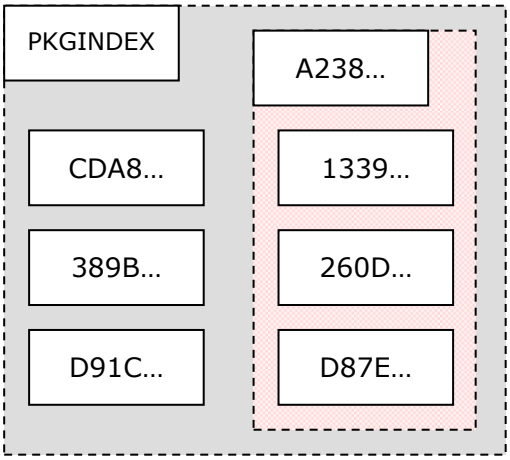

Figure 5 ZIP items in example 4

C.4.4.1 Root package: package index

The *root package index* is stored in a *ZIP item* with the *ZIP item name* of "`META-INF/PKGINDEX.XML`". This *ZIP item name* is fixed by the *clinical packages* specification.

In this *package index*, the *referenced package* has a `base` attribute that assigns a common prefix to *ZIP item names* for *ZIP items* belonging to that *referenced package*.

```
<?xml version="1.0" encoding="UTF-8"?>
<packageIndex
 xmlns="http://ns.electronichealth.net.au/pkg/xsd/PackageIndex/1.0">
  <part id="cda80e3f-17d4-447e-a703-63aaf642ac36"/>
  <part id="http://repository.example.com/24601/20111125/jvj"
        item="389b2987-5ad5-4b9d-be0a-781112830777"/>
  <part id="d91c5799-ddfb-4aaf-af50-44d82b3ba809"/>

  <package base="attach01/"
           id="pathologyReport0001"
           item="a2380e3f-17d4-447e-a703-63aaf642ac36"/>
  <distinguisher type="http://ns.example.com/eg4/root"
                 member="cda80e3f-17d4-447e-a703-63aaf642ac36"/>
  <distinguisher type="http://ns.example.com/eg4/dsig"
                 member="d91C144D9E6D43148D75C0DE26414E2E"/>

</packageIndex>
```

C.4.4.2 Root package: CDA XML document

The *identifier* for this *part* is "cda80e3f-17d4-447e-a703-63aaf642ac36".

This CDA XML document is stored in a *ZIP item* with the *ZIP item name* of "cda80e3f-17d4-447e-a703-63aaf642ac36". This *ZIP item name* can be obtained by concatenating the *package prefix* associated with the containing *package index* (i.e. the zero length string, since it is the *root package index*) with the value of the name attribute (i.e. "cda80e3f-17d4-447e-a703-63aaf642ac36" in this case).

Inside the CDA XML document, it references the image using the image's *identifier* and the *referenced packages* using its *identifier*.

In this example, the integrity check attributes allows the entire *package* to be digitally signed by signing just the CDA XML document. For the refer*enced packages*, the `reference` element contains the *identifier* and the digest is calculated on the digital signature *part* in the *referenced package*.

A processor expecting to process a CDA XML document can identify this part by finding the *distinguisher* whose *distinguisher type* is `http://ns.example.com/eg4/root` and finding the `pi:part` element with an `id` attribute that has the same value.

```
<?xml version="1.0" encoding="UTF-16"?>
<ClinicalDocument xmlns="urn:hl7-org:v3">
 …
 <value mediaType="image/jpeg"
  integrityCheckAlgorithm="SHA-256"
  integrityCheck="2jkFJ4ZGRUtLI4XyIr9FdHsT8Vdp5g2/L5fYnH3n0xI=">
  <reference value="http://repository.example.com/24601/20111125/jvj"/>
 </value>
 …
 <value mediaType="multipart/vnd.cda.package"
  integrityCheckAlgorithm="SHA-256"
  integrityCheck="wJnnbC1an7tckwyBUzyd60dMoG9i8H+/7jKSn54OdJk=">
  <reference value="pathologyReport0001"/>
 </value>
 …
</ClinicalDocument>
```

C.4.4.3 Root package: image

The *identifier* for this *part* is the URI "http://repository.example.com/24601/20111125/jvj".

The *root package's* image attachment is stored in a *ZIP item* with the *ZIP item name* of "389b2987-5ad5-4b9d-be0a-781112830777". This *ZIP item name* can be obtained by concatenating the *package prefix* associated with the containing *package index* (i.e. the zero length string, since it is the *root package index*) with the value of the `item` attribute (i.e. "389b2987-5ad5-4b9d-be0a-781112830777" in this case).

This is an example of when the `item` attribute is mandatory on a `pi:part` element, because the *identifier* does not have a printable US-ASCII equivalent value.

C.4.4.4 Root package: digital signature

The *root package's* digital signature is stored in a *ZIP item* with the *ZIP item name* of "d91c5799-ddfb-4aaf-af50-44d82b3ba809". This *ZIP item name* can be obtained by concatenating the *package prefix* associated with the containing *package index* (i.e. the zero length string since it is the *root package index*) with the value of the name attribute (i.e. "d91c5799-ddfb-4aaf-af50-44d82b3ba809" in this case).

A processor expecting to verify the digital signature of the *package* can identify this *part* by finding the *distinguisher* whose *distinguisher type* is

`http://ns.example.com/eg4/dsig` and finding the `pi:part` element with an `id` attribute that has the same value.

Inside the digital signature, it references the *parts* that it signs. This example assumes the hashed reference mechanism in CDA is used to ensure the integrity of the attachments referenced by the CDA XML document. Therefore, only the CDA XML document is digitally signed (i.e. the digital signature only references the *identifier* "cda80e3f-17d4-447e-a703-63aaf642ac36").

```
<?xml version="1.0" encoding="UTF-8"?>
<sp:signedPayload
 xmlns:sp="http://ns.electronichealth.net.au/xsp/xsd/SignedPayload/2010"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
 <sp:signatures>
 …
 </sp:signatures>
 <sp:signedPayloadData id="x">
  <ds:Manifest>
   <ds:Reference URI="cda80e3f-17d4-447e-a703-63aaf642ac36">
    <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <ds:DigestValue>nFWmKjsgB91Ex3+7MnSE/lqPEpE=</ds:DigestValue>
   </ds:Reference>
  </ds:Manifest>
 </sp:signedPayloadData>
</sp:signedPayload>
```

C.4.4.5 Referenced package: package index

The *package index* for the first *referenced package* is stored in a *ZIP item* with the *ZIP item* name of "`attach01/a2380e3f-17d4-447e-a703-63aaf642ac36`". This *ZIP item name* can be obtained by concatenating the *package prefix* associated with the containing *package index* (i.e. the zero length string, since it is the *root package index*) with the value of the `base` attribute (i.e. "`attach01/`" in this case) and the value of the `item` attribute (i.e. "`a2380e3f-17d4-447e-a703-63aaf642ac36`" in this case).

```
<?xml version="1.0" encoding="UTF-16"?>
<packageIndex
 xmlns="http://ns.electronichealth.net.au/pkg/xsd/PackageIndex/1.0">

  <part id="1339429a-e8c6-4bfa-9ebd-599d9a7e5191"/>
  <part id="260d5e6e-a28f-4a0d-a86a-bed6a611f877"/>
  <part id="d87e61d7-69ca-4ae8-a408-8eec513449ba">

  <distinguisher type="http://ns.example.com/eg3/root"
                 member="1339429a-e8c6-4bfa-9ebd-599d9a7e5191"/>
  <distinguisher type="http://ns.example.com/eg3/dsig"
                 member="d87e61d7-69ca-4ae8-a408-8eec513449ba"/>
</packageIndex>
```

C.4.4.6 Referenced package 1: CDA XML document

The *identifier* for this *part* is "1339429a-e8c6-4bfa-9ebd-599d9a7e5191".

This CDA XML document is stored in a *ZIP item* with the *ZIP item name* of "`attach01/1339429a-e8c6-4bfa-9ebd-599d9a7e5191`". This *ZIP item name* can be obtained by concatenating the *package prefix* associated with the containing *package index* (i.e. the string "`attach01/`") with the value of the `id` attribute (i.e. "`13397955DD63450E9C3E85B99E2104C9`").

A processor expecting to process a CDA XML document can identify this *part* by finding the *distinguisher* with *distinguisher type* `http://eg.example.com/eg3/root` and the `pi:part` element with an `id` attribute with the same value.

Inside this CDA XML document, it references the image using the image's *identifier*.

```
<?xml version="1.0" encoding="UTF-8"?>
<ClinicalDocument xmlns="urn:hl7-org:v3">
 …
 <value mediaType="image/jpeg"
  integrityCheckAlgorithm="SHA-256"
  integrityCheck="wGR+V5snZvnTm39zMkNZfFzV9QzjS44RaJj28SH+EkM=">
  <reference value="260d5e6e-a28f-4a0d-a86a-bed6a611f877"/>
 </value>
 …
</ClinicalDocument>
```

C.4.4.7 Referenced package 1: image

The *identifier* for this *part* is "260d5e6e-a28f-4a0d-a86a-bed6a611f877".

This image is stored in a *ZIP item* with the *ZIP item name* of "attach01/260d5e6e-a28f-4a0d-a86a-bed6a611f877". This *ZIP item name* can be obtained by concatenating the *package prefix* associated with the containing *package index* (i.e. the string "attach01/") with the value of the name attribute (i.e. "260d5e6e-a28f-4a0d-a86a-bed6a611f877").

C.4.4.8 Referenced package 1: digital signature

This digital signature is stored in a *ZIP item* with the *ZIP item name* of "attach01/d87e61d7-69ca-4ae8-a408-8eec513449ba". This *ZIP item name* can be obtained by concatenating the *package prefix* associated with the containing *package index* (i.e. the string "attach01/") with the value of the name attribute (i.e. "d87e61d7-69ca-4ae8-a408-8eec513449ba" in this case).

Inside this digital signature, it references the *parts* that it signs. This example assumes the hashed reference mechanism in CDA is used to ensure the integrity of the attachments referenced by the CDA XML document. Therefore, only the CDA XML document is digitally signed (i.e. the digital signature only references the *identifier* "1339429a-e8c6-4bfa-9ebd-599d9a7e5191").

A processor expecting to verify the digital signature of the first attachment *package* can identify this *part* by finding the *distinguisher* with *distinguisher type* of http://eg.example.com/eg3/dsig and finding the pi:part element with a name attribute that has the same value.

```
<?xml version="1.0" encoding="UTF-16"?>

<sp:signedPayload
 xmlns:sp="http://ns.electronichealth.net.au/xsp/xsd/SignedPayload/2010"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
 <sp:signatures>
 …
 </sp:signatures>

 <sp:signedPayloadData id="x">

  <ds:Manifest>
   <ds:Reference URI="1339429a-e8c6-4bfa-9ebd-599d9a7e5191">
    <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <ds:DigestValue>D1qrG2xg+FOYRPb71NPUVwAWisM=</ds:DigestValue>
   </ds:Reference>
  </ds:Manifest>

 </sp:signedPayloadData>

</sp:signedPayload>
```

C.4.5 Overview of digest values

An overview of the digest values used in this example is shown in Figure 6. In this figure, circles represent a digest value stored inside a *ZIP item* and the arrow points to the *ZIP item* that the digest is calculated over.

The integrity of the entire *clinical package* can be checked by first checking the digital signature in the digital signature part of the *root package* (i.e. the *ZIP item* named "`d91c5799-ddfb-4aaf-af50-44d82b3ba809`") and then checking references from it.

As per the specification, *package index ZIP items* are not digitally signed.

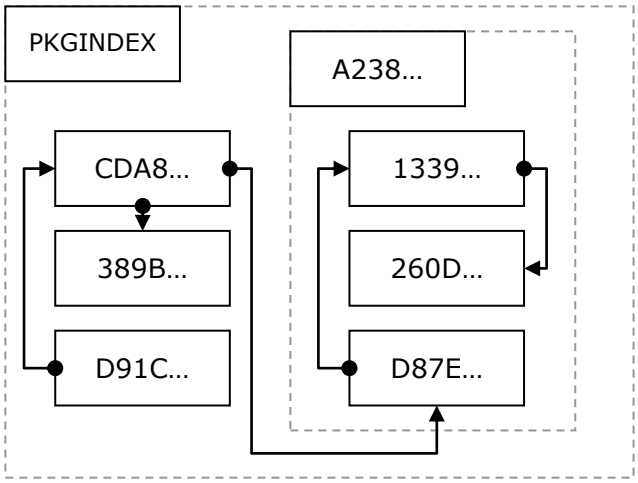

Figure 6 Overview of digest values in example 4