nehta

Technical Specification

Template Package

Version 1.0 — 28 May 2012

Final

National E-Health Transition Authority

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

Document Information

Approver	PCEHR Design Authority
Owner	Head of PCEHR
Contact officer	PCEHR Design Team
Contributors	PCEHR - Standards, Foundation and Architecture (SFA) Team

Version History

Date	Version	Name	Comments/type of review
28 May 2012	1.0		First issue

Table of Contents

Preface

Purpose

The purpose of this document is to define the structure of the Template Package used by the Template Service to transport standardised specifications for clinical documents.

Intended Audience

This NEHTA-managed specification is intended for:

- Developers and implementers of PCEHR Conformant Repositories and other shared electronic health record (SEHR) systems (normative).
- Jurisdictional eHealth programs (informative).
- The Australian Health Informatics Standards development community (informative).
- Developers and implementers of software products which seek to interact with the PCEHR System or other SEHR systems (normative).
- Developers and implementers of software products that exchange clinical documents in with other systems utilising inter-operable formats (normative).

This is a technical document which makes use of the UML2.3 standard [UML2010].

This document assumes that the reader is familiar with:

- UML and service-oriented architecture concepts and patterns
- The PCEHR Concept of Operations [PCEHR_CON_OPS], September 2011 release.

Document Map

This document is not part of the PCEHR deliverable matrix, but should be read in conjunction with the Template Service Interface Logical Service Specification [LSS2011] and the Template Service Interface Technical Service Specification [TSS2011].

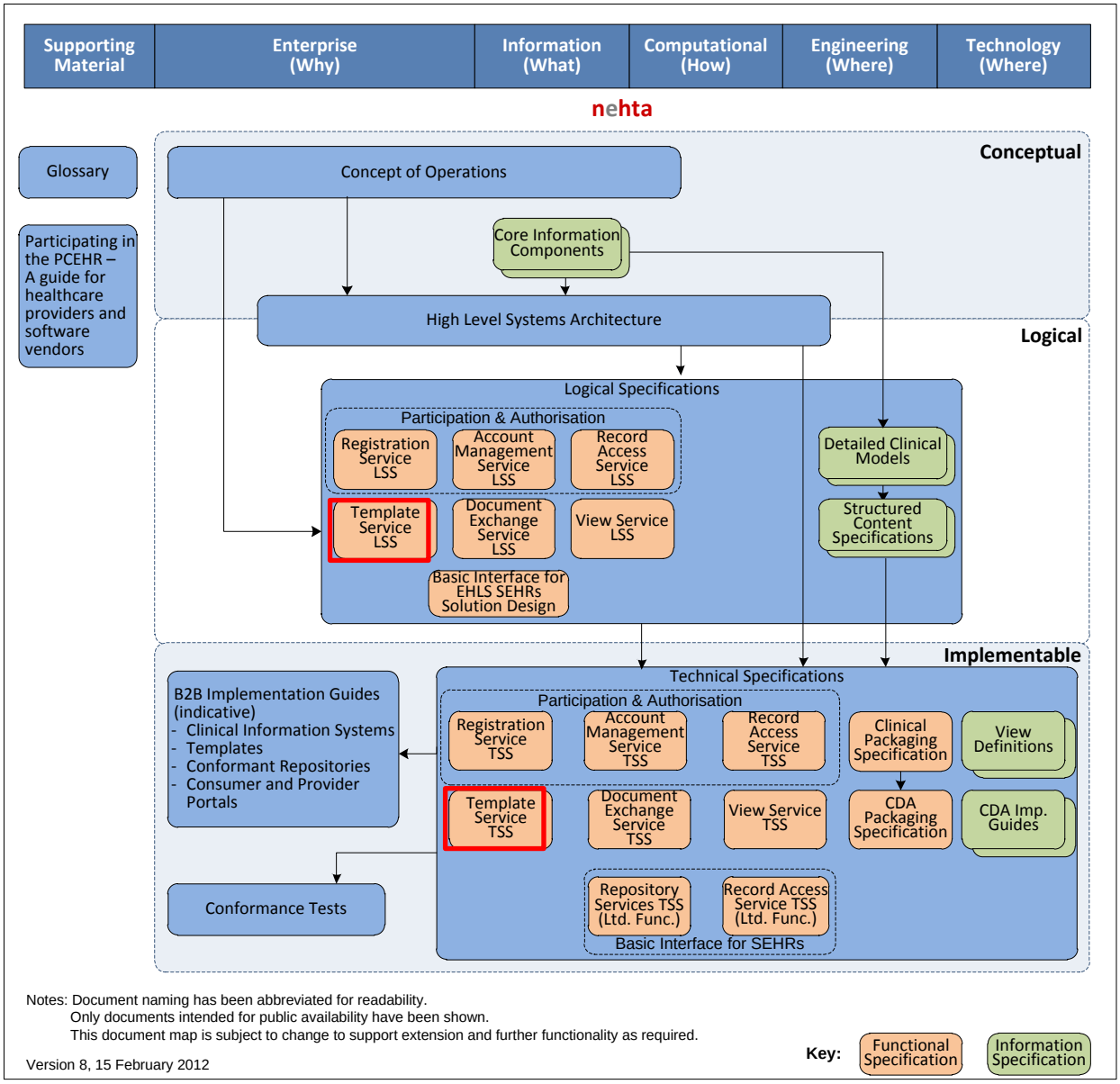

Figure 1 – Document map

Acronyms and Terminology

Please refer to Appendix B for definitions of the acronyms and terminology used in this document.

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

References

Please refer to Appendix C for details of the references used in this document.

1 Executive Summary

The Template Service enables implementers and systems to obtain standardised specifications for clinical documents to be exchanged within the Australian healthcare community. These specifications can contain both documents intended for human interpretation and machine-usable definitions intended for use in implementing interoperable systems. The Template Service handles all information related to a particular type of clinical document as a single template package.

This document defines the internal structure of template packages. The template package is the payload format used for exchanging information between the Template Service and its users. It is used by both its Template Portal component and its integration interface specified in the Template Service Interface Logical Service Specification [LSS2011] and the Template Service Interface Technical Service Specification [TSS2011].

2 Introduction

2.1 Context

The Template Service enables implementers and systems to obtain standardised specifications for clinical documents to be exchanged within the Australian healthcare community. The Template Service is responsible for managing and storing the data representations associated with all of the data formats stored within a personally controlled electronic health record (PCEHR), but can also store specifications for clinical documents not associated with a PCEHR.

A Template Package encapsulates the definition of a clinical document type and includes:

- a unique identifier
- data definitions
- data validation definition
- information about how to render a clinical document
- supporting material (such as implementation guides).

This information is packaged as a number of different files to meet the various needs, as well as metadata information about each file, and about the template as a whole.

2.2 Scope of Document

2.2.1 In Scope

This technical specification document defines the structure and contents of a Template Package, which is the mechanism for transporting a Template.

Each component within the Template Package, as well as the Template Package itself, is associated with metadata elements which provide information relevant to the provenance, handling and use of the component. These items are defined within this document.

2.2.2 Out of Scope

This document does not cover the requirements of tools or processes required to create or consume Template Packages.

The governance processes associated with the loading or management of templates are not within the scope of this document.

2.3 Conformance Points

This specification contains conformance points that identify normative requirements that are to be met by creators and users of Template Packages.

Any capability required to meet a conformance point SHALL be considered part of the requirements to be met under this specification.

Conformance points are identified in this document by the following notation:

TPKG-T 0	This is an example only. Conformance points SHALL be numbered and contain an identifier of '**TPKG-T**' which identifies them as being applicable to the Template Package technical specification.

3 Package Construction

3.1 Package Structure

The template itself consists of a number of files with different uses. They are transported as a ZIP file, which can be considered to have the folder structure shown in Figure 2. Conformance points relating to the logical package structure are defined in the Template Service Interface Logical Service Specification [LSS2011].

TPKG-T 1	All conformance points detailed in the Template Service Interface Logical Service Specification [LSS2011] SHALL apply to Template Packages.
TPKG-T 2	A template package SHALL NOT be encrypted or password protected.

The use of folders within the structure is intended to facilitate simpler navigation and use of the package by persons acquiring the package for non-automated use. The folder structure adds no value and conveys no meaning to automatic system users, as all the required information is conveyed via the metadata and manifest files.

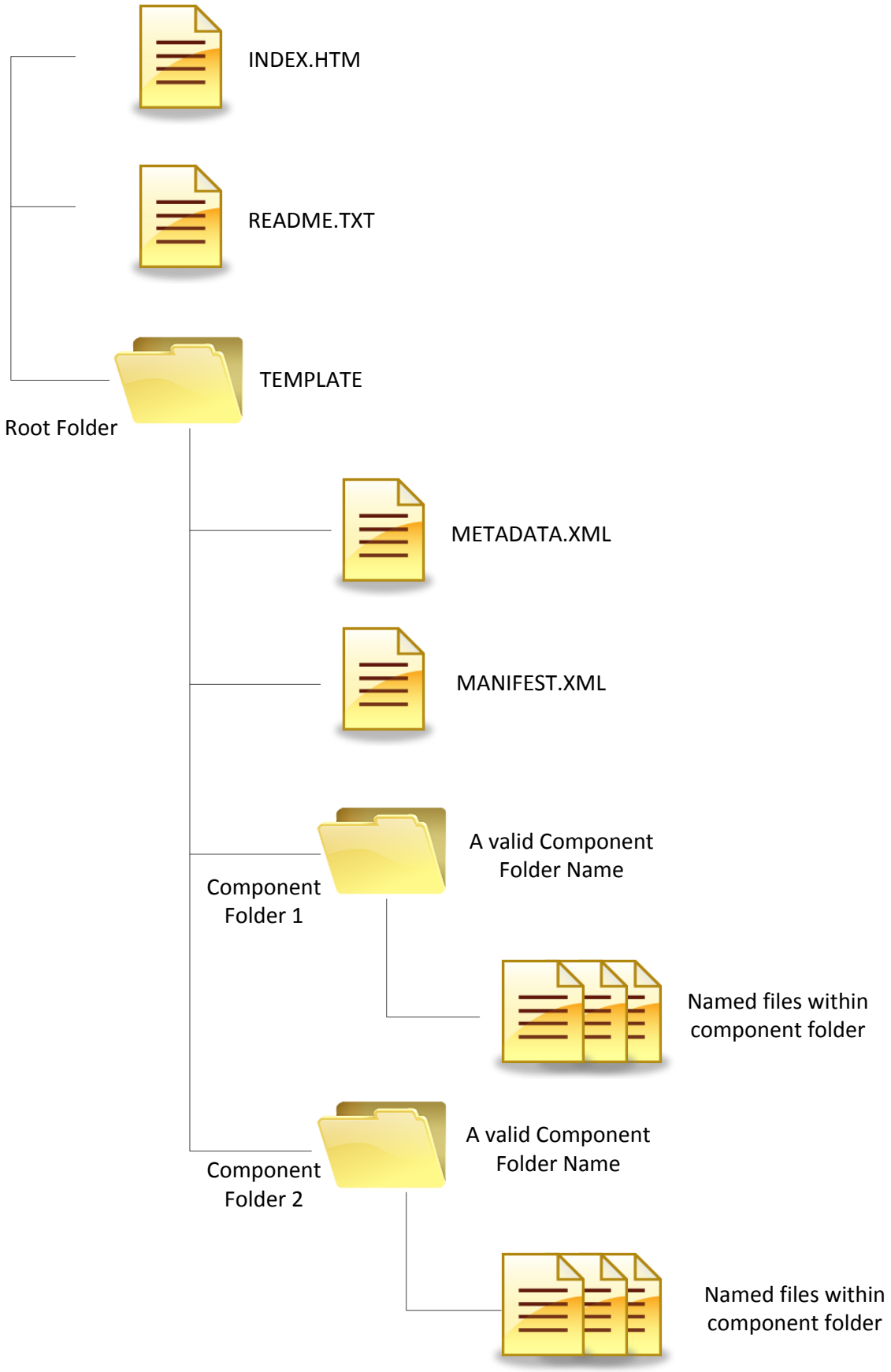

Figure 2 - Template Package Structure

Table 1 describes the files in the Template Package.

Table 1 – Files in the Template Package

File Name	Required	Description
INDEX.HTM	Optional	This file provides an HTML entry point into the package contents. If present, it can be used to provide a means to display or render selected components of the package.
README.TXT	Optional	This file contains informative data about the Template package. If present, it will contain information about the actual retrieval (such as search parameters, date of access etc.).
Root Folder – TEMPLATE	Mandatory	This folder defines the root of the Template Package contents. The name of this folder is fixed to TEMPLATE.
METADATA.XML	Mandatory	This file contains Template Package metadata. See [LSS2011] for logical content.
MANIFEST.XML	Mandatory	This file contains the definition for the template component files. It contains metadata for each file contained within the component type folders, including the file name and location. The [LSS2011] defines the structure.
Component type folders	Mandatory	Template components get bundled into component types to enable segregation of content and enhance usability when accessed via the Template Portal. There is no naming restriction on the component folder names, but the files contained within them must be explicitly referenced in the MANIFEST.XML file. Refer [LSS2011] for conformance point for component type folder naming information
Components	Mandatory	The actual files which make up the Template Package. Within each component folder, different format files may be stored. Each file in the component folder is referenced in MANIFEST.XML file.

Template package part construction details are specified in the Template Service Interface Technical Service Specification [TSS2011].

TPKG-T 3	All conformance points detailed in the Template Service Interface Technical Service Specification [TSS2011] SHALL apply to Template Packages.

The structure outlined above has two levels of folders, with the component type folders all being contained within the TEMPLATE folder.

TPKG-T 4	The *Component type folders* SHOULD NOT contain other folders. These *Component type folders* SHOULD contain component parts directly.

Informative Note

This structure enables all files to be processed to be referenced from the manifest file. Some content types require the use of multiple files to achieve the required outcome (such as multiple schema components). These files will all be stored in the same logical component type folder. The use of "sub-folders" would require the use of those folder names to maintain internal references within the components. This in turn would impose an obligation on the user of the package to maintain a folder structure when using the components.

3.2 Package References

TPKG-T 5	Template package components SHALL NOT rely on any external references. A Template package SHALL be self-contained.

Informative Note

Components within a template package should not require external content in order to be processed. Users of Template Packages may not have access to external components for many different reasons. The Template Package must be able to fulfil its purpose without the necessity for additional information.

TPKG-T 6	References within a component SHALL only be made to other components within the same Component type folder.
TPKG-T 7	References between components SHALL use relative reference names.

Informative Note

Use of absolute references to other components would imply that the references are to external components, which is not permitted. As the Template Package is intended to be transported and used in multiple environments, only relative references can be used.

TPKG-T 8	Relative references in components SHOULD NOT utilise ".” Or "..” notation. These references SHOULD spell out the relative folder and file names in full.

Informative Note

If the recommendation to not contain sub-folders is followed, then no component references will require folder names or paths within the reference, as all components will reside in the same folder. If it is necessary to use sub-folders, references should utilise the full names, as some users have difficulties processing the "..” notation. This limitation precludes some references from such sub-folders, and is one of the rationales for restricting the use of sub-folders.

4 Package Components

4.1 Definition Package Components

The Template package contains components which define a template or form. Multiple components may be required to achieve this and they may be of different types. The definition components can be classified into two main categories:

1. Formal components that make up the definition (from the perspective of actual use). Machine processable and dynamic components will fall into this category where they are normative, but static components may also be included (such as a static form intended only for printing and handwritten input).
2. Information, guidelines and other supporting documents which provide additional information about the definition. This may include formal specifications which define what has been rendered into other components to be used. Machine processable components that provide options or examples may be included in this category when they are non-normative.

TPKG-T 9	All formal components defining the template or form SHALL be placed in a single *Component type folder.*
TPKG-T 10	Formal components defining the template or form SHOULD be placed in a *Component type folder* named DEFN.
TPKG-T 11	All supporting information for the definition SHOULD be placed in a single *Component type folder.*
TPKG-T 12	Supporting information for the definition SHOULD be placed in a *Component Type folder* named DEFINFO.

Where non-normative machine-processable definition information is included within a template package, it may be included in its own Component type folder, separate from the DEFINFO folder. This will facilitate ease of use and differentiate it from other human-readable components.

The folder names used have no formal significance, and the type of the component (and hence its use and meaning within the template package) is defined by attributes within the manifest.xml file entry for that component.

TPKG-T 13	The `TemplateComponentType` of the manifest for a formal component defining the template or form SHALL be `Definition`.
TPKG-T 14	The `TemplateComponentType` of the manifest for supporting information defining the template or form SHALL be `Information-Definition`.
TPKG-T 15	The `TemplateComponentType` of the manifest for supporting information defining alternative non-normative representations of the template or form SHALL be `Definition-Alternative`.

Many definition types (such as XML schema) may be realised as multiple files, with common components referenced, by mechanisms such as "include" or "import". These files are part of the formal definition, but are not intended for processing on their own. Users of the template package will utilise those components when processing the actual definition.

TPKG-T 16	The `TemplateComponentType` of the manifest for an inclusion used by the template or form definition SHALL be `Definition-Inclusion`.

Informative Note

Included definition items may contain definitions for components not relevant to the template in question, due to being shared by multiple different templates. External

source control or publishing requirements may prevent the removal of those elements. Utilising an inclusion type ensures that only those items referenced from the actual definition will be considered. Inclusions could apply to either the normative `Definition` components, or `Definition-Alternative` components.

4.1.1 Definition Definition

A template or form is defined by the contents of the package. The use of the form, or the creation of a document from the template, need not use every component, as not all aspects may be required, but it must not contradict any aspect. Where alternative options exist, these are realised either as formal options within the definition or as separate template packages.

TPKG-T 17 All components of `TemplateComponentType Definition` SHALL comprise the definition. Any document or form instance derived from the template package SHALL NOT contradict any part of any `Definition`.

Informative Note

Some template packages may contain multiple definitions for alternative technology platforms. Any such definition may be utilised to realise a valid instance of the document or form. If validation of the resultant output against the template definition is being performed, any arbitrary definition suitable to the validator may be utilised, since the output must conform to all definitions.

Where information is included that is non-normative, such as sample schema information to be used as a base for customisation, this will not be a `Definition` component, but will be a `Definition-Alternative`.

4.1.2 Definition Standards

Template Packages are flexible in that they may be utilised to specify definitions in many different formats. These formats must be able to be reliably applied by any user in a manner which matches the intent of the creator, thus aiding interoperability.

The application of appropriate standards to the content of packages is strongly encouraged. It is only the need to maintain flexibility to accommodate the needs of different participants that prevents the Template Service imposing mandatory compliance with standards.

The governance process which controls the approval of templates for inclusion within the Template Service will contain policies to ensure appropriate compliance with standards.

Standards applicable to Template Definitions include:

- ISO/IEC 11179 Information technology — Metadata registries [ISO11179]
- W3C Recommendations : XML Schema [W3CXSD]

4.2 Validation Package Components

The Template Package contains components which can be used to validate an instance derived from a template or form. These validations do not necessarily check the structure of the instance, as that function may be covered by compliance with the definition. Validation components are used to further restrict the content of an instantiated document or form. Multiple components may be required to achieve this, and they may be of different types. The validation components can be separated into two categories:

1. Those that may be utilised to execute validations.
2. Information, guidelines and other supporting documents which provide additional information about the validation. This may include formal or

readable specifications which define what has been rendered into other components to be used.

TPKG-T 18 All executable validation components for the template or form SHALL be placed in a single *Component type folder.*

TPKG-T 19 Executable validation components for the template or form SHOULD be placed in a *Component type folder* named VALDN.

TPKG-T 20 All supporting information for validation SHALL be placed in a single *Component type folder.*

TPKG-T 21 Supporting information for validation SHOULD be placed in a *Component Type folder* named VALINFO.

The folder names used have no formal significance, and the type of the component (and hence its use and meaning within the template package) is defined by attributes within the manifest.xml file entry for that component.

TPKG-T 22 The `TemplateComponentType` of the manifest for an executable validation component for the template or form SHALL be `Validation`.

TPKG-T 23 The `TemplateComponentType` of the manifest for supporting information for validation components of the template or form SHALL be `Information-Validation`.

Many validation types (such as schematron) may be realised as multiple files, with common components referenced, by mechanisms such as "include" or "import". These files are part of the validation, but are not intended for processing on their own. Users of the template package will utilise those components when processing the actual validation.

TPKG-T 24 The `TemplateComponentType` of the manifest for an inclusion used by the template or form validation SHALL be `Validation-Inclusion`.

4.2.1 Validation Definition

The validation requirements of a template or form are defined by the contents of the package. In order for an instance derived from a template of form to be valid, all validation aspects must be fulfilled.

TPKG-T 25 All components of `TemplateComponentType` `Validation` SHALL comprise the validation. Any document or form instance derived from the template package SHALL satisfy every `Validation` component within the Template Package.

Different validation components may require application in different ways. Some validations may be conditional and may depend on the results of other validations. There is no restriction placed on the order in which different validation components are applied when validating a document or form. A validation component may have sequencing imposed internally, but cannot rely on any sequencing with respect to other validation components within the template package.

TPKG-T 26 The order of application of `Validation` components SHALL NOT have any effect on the outcome of the validation.

4.2.2 Validation Standards

Template Packages are flexible in that they may be utilised to specify validations in many different formats. These formats must be able to be reliably applied by any user in a manner which matches the intent of the creator, thus aiding interoperability.

The application of appropriate standards to the content of packages is strongly encouraged. It is only the need to maintain flexibility to accommodate the needs of different participants that prevents the Template Service imposing mandatory compliance with standards.

The governance process which controls the approval of templates for inclusion within the Template Service will contain policies to ensure appropriate compliance with standards.

Standards applicable to Template Validations include:

- ISO/IEC 19757-3 Information technology – Document Schema Definition Languages (DSDL) – Part 3: Rule Based validation – Schematron [ISO19757]

4.2.3 Validation Outcome

Validations contained within templates may be performed in different ways, and may produce intermediate or optional results, which could include suggestions, warnings, errors, or any other required outputs. Regardless of any other output produced, each validation component will have a formal result which may be expressed as a Boolean value.

TPKG-T 27	A `validation` component SHALL return a Boolean result

4.3 Information Package Components

The Template package may contain informative documentation related to the package as a whole, its intended, use, or any other relevant information types.

TPKG-T 28	General information components for the template or form SHOULD be placed in a *Component type folder* named INFO.

The folder names used have no formal significance, and the type of the component (and hence its use and meaning within the template package) is defined by attributes within the manifest.xml file entry for that component.

TPKG-T 29	The `TemplateComponentType` of the manifest for a general information component for the template or form SHALL be `Information`.

4.4 Transformation Package Components

The Template package may contain components intended to transform documents or forms conforming to the template package. These transformations may be undertaken for different types of reasons, including:

- Conversion to different physical formats such as:
  - Rendering for display on a screen
  - Production of a hard copy
  - Transformation to a different document format for exchange utilising other media (e.g. creation of a PDF file)
- Selection of discrete contents from a document, such as a summary section
- Changing or removing certain content, such as identifying information, to enable use for research or other secondary purposes.

TPKG-T 30	Tranformation components for the template or form SHOULD be placed in a *Component type folder* named TRANS.

The folder names used have no formal significance, and the type of the component (and hence its use and meaning within the template package) is defined by attributes within the manifest.xml file entry for that component.

TPKG-T 31	The `TemplateComponentType` of the manifest for a transformation component for the template or form for display within a web-browser SHALL be `Transform-Display`.
TPKG-T 32	The `TemplateComponentType` of the manifest for a transformation component for the template or form for changing format types SHALL be `Transform-Format`.

TPKG-T 33 The `TemplateComponentType` of the manifest for a transformation component for selectively choosing content from a document SHALL be `Transform-Select`.

TPKG-T 34 The `TemplateComponentType` of the manifest for a transformation component for removing identifying information SHALL be `Transform-Anonymous`.

4.4.1 Transformation Standards

Template Packages are flexible in that they may be utilised to specify transformations in many different formats. These formats must be able to be reliably applied by any user in a manner which matches the intent of the creator, thus aiding interoperability.

The application of appropriate standards to the content of packages is strongly encouraged. It is only the need to maintain flexibility to accommodate the needs of different participants that prevents the Template Service imposing mandatory compliance with standards.

The governance process which controls the approval of templates for inclusion within the Template Service will contain policies to ensure appropriate compliance with standards.

Standards applicable to Template Transformations include:

- W3C Recommendations : Cascading Style Sheets [W3CCSS]
- W3C Recommendations : XSL Transformations [W3CXSLT]

5 Package and Component Metadata

The XML schema for the metadata.xml and manifest.xml files are detailed in Appendix A.

5.1 Template Package Metadata

5.1.1 Metadata Structure

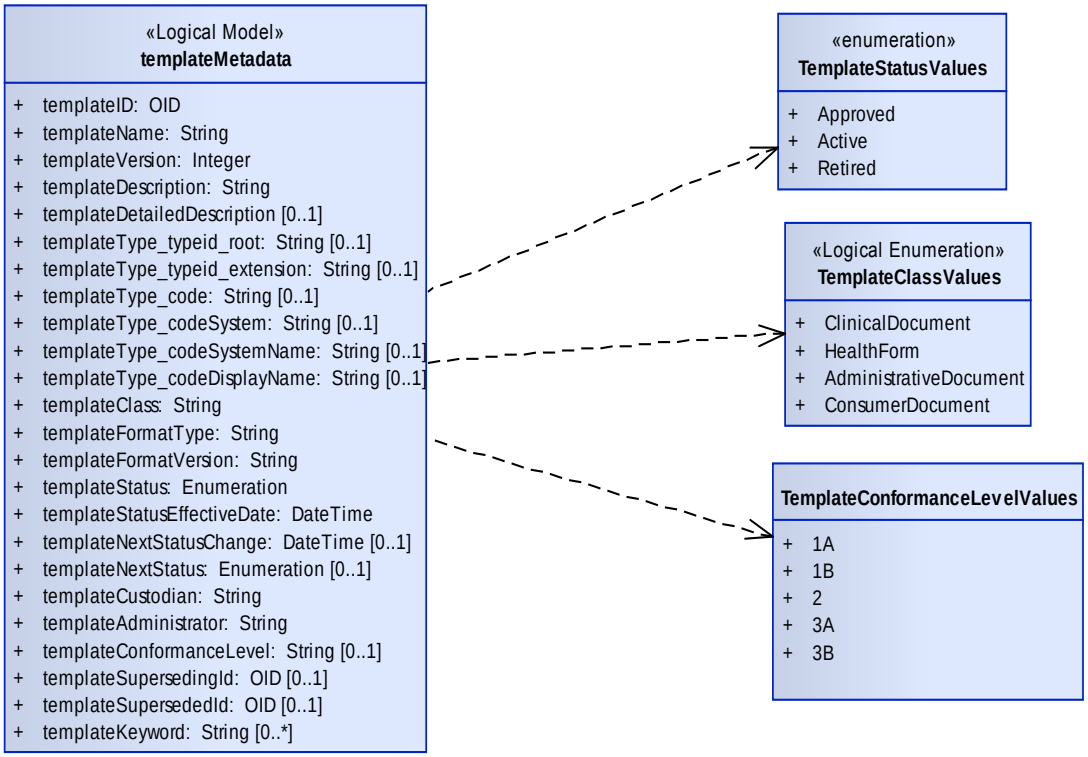

Figure 3 – Template Metadata Model

Table 2 – Template Package Metadata

templateMetadata			
Field	**Data Type**	**Description**	**Cardinality**
templateID	OID	The unique identifier for a template.	1
templateName	String	A short name for the template. Not guaranteed to be unique, but shared for different versions of a template. It is not required that all templates with the same name be related in any manner.	1
templateVersion	Integer	The version number of the template.	1
templateDescription	String	A short, easily rendered description of the template.	1
templateDetailedDescription	String	The detailed text description of the template.	0..1

templateMetadata			
Field	**Data Type**	**Description**	**Cardinality**
templateType_typeid_root	String	A part of the definition of the CDA document type applicable to a CDA document generated from the template. Only relevant to templates defining a CDA document.	0..1
templateType_typeid_extension	String	A part of the definition of the CDA document type applicable to a CDA document generated from the template. Only relevant to templates defining a CDA document.	0..1
templateType_code	String	A part of the definition of the CDA document type applicable to a CDA document generated from the template. Only relevant to templates defining a CDA document.	0..1
templateType_codeSystem	String	A part of the definition of the CDA document type applicable to a CDA document generated from the template. Only relevant to templates defining a CDA document.	0..1
templateType_codeSystemName	String	A part of the definition of the CDA document type applicable to a CDA document generated from the template. Only relevant to templates defining a CDA document.	0..1
templateType_codeDisplayName	String	A part of the definition of the CDA document type applicable to a CDA document generated from the template. Only relevant to templates defining a CDA document.	0..1
templateClass	String	A broad grouping which defines the sort of document that would be generated from the template.	1
templateFormatType	String	The super type of format the template adheres to, e.g. CDA, RTF, HL7 V2.	1
templateFormatVersion	String	The version of the TemplateFormatType.	1
templateStatus	String	The current status of the template.	1
templateStatusEffectiveDate	DateTime	The date the TemplateStatus took effect.	1
templateNextStatusChange	DateTime	The date that the TemplateStatus expires and the TemplateNextStatus will become the new status.	0..1
templateNextStatus	String	The status the template will have when the current status expires. Has the same permitted values as the TemplateStatus.	0..1
templateCustodian	String	The identifier of the template custodian that can be resolved from a person registry.	1

templateMetadata			
Field	**Data Type**	**Description**	**Cardinality**
templateAdministrator	String	The identifier of the package administrator that can be resolved from a person registry.	1
templateConformanceLevel	String	A CDA conformance level deemed to be met by documents which validate according to the template definition. Only applies to templates which define a CDA document.	0..1
templateSupersedingId	OID	The TemplateID of the template that replaced this one. Identifies the next later version of the template.	0..1
templateSupersededId	OID	The TemplateID of the template that this template is replacing. Identifies the immediate prior version of the template.	0..1
templateKeyword	String	Any searchable keyword that may identify the template. Multiple occurrences are permitted.	0..*

TPKG-T 35 The value for the "name" metadata item SHALL correspond to a literal present in the "Field" column of Table 2 above.

TPKG-T 36 The `templateID` SHALL be an OID that is unique across the Template Service.

TPKG-T 37 The `templateName` SHALL NOT contain leading or trailing spaces. It SHALL NOT be a null or zero length string.

TPKG-T 38 The `templateVersion` SHALL NOT be set to a value of 0.

TPKG-T 39 The `templateVersion` SHALL contain a value greater than the value of the `templateVersion` contained in the Template Package that has a `templateId` value that corresponds to the value of the `templateSupersededId` if it is present.

TPKG-T 40 The `templateDescription` SHALL NOT contain leading or trailing spaces. It SHALL NOT be a null or zero length string.

TPKG-T 41 If present, the `templateDetailedDescription` SHALL NOT contain leading or trailing spaces. It SHALL NOT be a null or zero length string.

TPKG-T 42 If any of `templateType_typeid_root`, `templateType_typeid_extension`, `templateType_code`, `templateType_codeSystem`, `templateType_codeSystemName` or `templateType_codeDisplayName` is present, then all of them SHALL be present.

TPKG-T 43 If `templateType_typeid_root`, `templateType_typeid_extension`, `templateType_code`, `templateType_codeSystem`, `templateType_codeSystemName` or `templateType_codeDisplayName` are present then the values contained SHALL be valid for populating the corresponding HL7 Clinical Document Architecture Release 2 (CDA) `ClinicalDocument/typeId` and `ClinicalDocument/code` fields.

TPKG-T 44 The `templateClass` SHOULD contain one of `ClinicalDocument`, `HealthForm`, `AdministrativeDocument` or `ConsumerDocument`.

TPKG-T 45 The `templateFormatType` SHALL NOT contain leading or trailing spaces. It SHALL NOT be a null or zero length string.

TPKG-T 46 The `templateFormatVersion` SHALL NOT contain leading or trailing spaces. It SHALL NOT be a null or zero length string.

TPKG-T 47 The `templateStatus` SHALL contain one of `Approved`, `Active` or `Retired`.

TPKG-T 48 The `templateStatusEffectiveDate` SHALL NOT be in the future.

TPKG-T 49 If present, the `templateNextStatus` SHALL contain one of `Approved`, `Active` or `Retired`.

TPKG-T 50 If present, the `templateNextStatus` SHALL NOT contain the same value as the `templateStatus`.

TPKG-T 51 If the `templateNextStatus` is present, the t`emplateNextStatusChange` SHALL be present.

TPKG-T 52 If the `templateNextStatusChange` SHALL NOT be present unless the t`emplateNextStatus` is also present.

TPKG-T 53 The t`emplateNextStatusChange` SHALL be in the future.

TPKG-T 54 The `templateCustodian` SHALL NOT contain leading or trailing spaces. It SHALL NOT be a null or zero length string.

TPKG-T 55 The `templateAdministrator` SHALL NOT contain leading or trailing spaces. It SHALL NOT be a null or zero length string.

TPKG-T 56 The `templateAdministrator` SHALL contain a representation of the user identification of the Template Portal user responsible for loading the template package.

TPKG-T 57 If present, the `templateConformanceLevel` SHOULD contain one of `1A`, `1B`, 2, `3A` or `3B`.

TPKG-T 58 If present, the `templateSupersedingId` SHALL be an OID that corresponds to the value of the `templateID` of another template housed in the Template Service.

TPKG-T 59 If present, the `templateSupersededId` SHALL be an OID that corresponds to the value of the `templateID` of another template housed in the Template Service.

TPKG-T 60 If both the `templateSupersededId` and the `templateSupersedingId` are present, they SHALL NOT have the same value.

TPKG-T 61 If present, the `templateKeyword` SHALL NOT contain leading or trailing spaces. It SHALL NOT be a null or zero length string.

5.2 Template Package Manifest

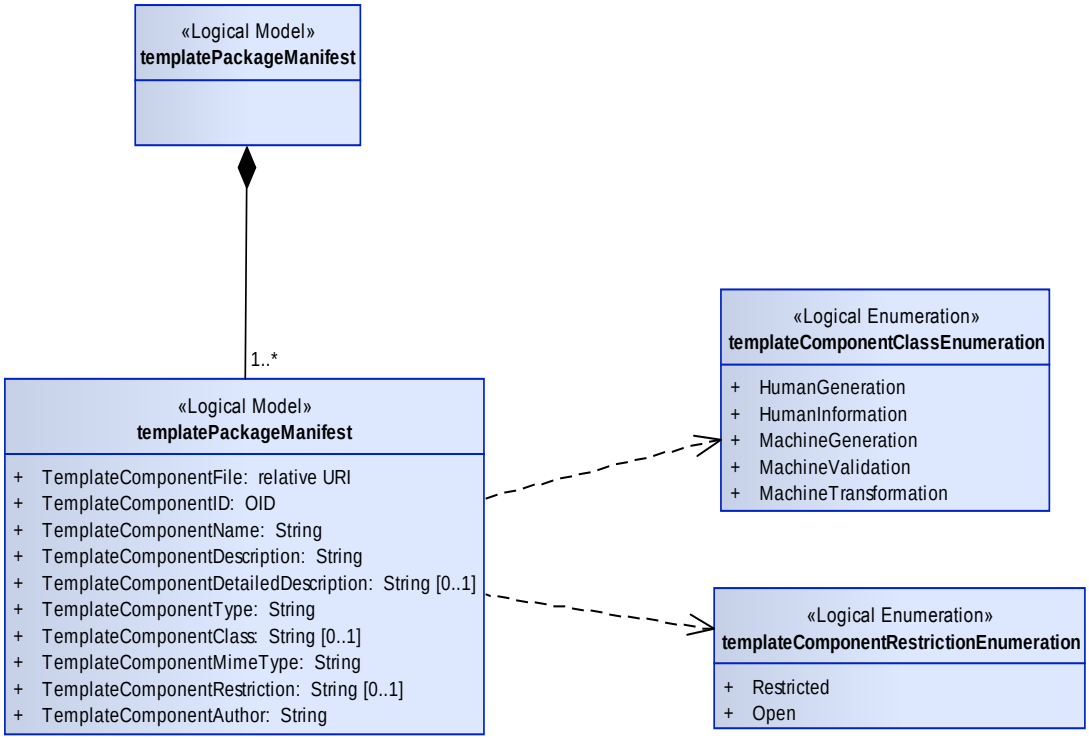

Figure 4 – Template Package Manifest Model

Table 3 – Template Package Manifest

templatePackageManifest			
Field	**Data Type**	**Description**	**Cardinality**
TemplateComponent File	relative URI	The relative reference to the file within the structure that this entry describes.	1
TemplateComponent ID	OID	The unique identifier for the template component.	1
TemplateComponentName	String	A short name for the template component. Not guaranteed to be unique.	1
TemplateComponentDescription	String	A short, easily rendered description of the template component.	1
TemplateComponentDetailedDescription	String	The detailed text description of the template component.	0..1
TemplateComponent Type	String	The logical use of this template. Needed to distinguish, especially for forms, where mimeType of RTF may be a form, or may just be an instruction.	1
TemplateComponentClass	String	Grouping of like components to enable selection of subsets of template components.	0..1
TemplateComponentMimeType	String	The mimeType of this template component.	1

templatePackageManifest			
TemplateComponentRestriction	String	Classification of component to enable selection for restricted or public access. Not all template or form components may have the same access levels.	0..1
TemplateComponentAuthor	String	The main author of the template component.	1

TPKG-T 62 The `TemplateComponentFile` SHALL be a relative URI that specifies a logical file within the Template Package ZIP. The URI SHALL be relative to the location of the manifest.xml file within the ZIP.

TPKG-T 63 The `TemplateComponentID` SHALL be an OID. The `TemplateComponentID` SHOULD be unique within this template package, but if it is not, then any two components with the same `TemplateComponentID` SHALL have the same content.

TPKG-T 64 If the same component file is packaged into different template packages, the value of the `TemplateComponentID` SHOULD be the same in each package.

TPKG-T 65 The `TemplateComponentName` SHALL NOT contain leading or trailing spaces. It SHALL NOT be a null or zero length string.

TPKG-T 66 The `TemplateComponentDescription` SHALL NOT contain leading or trailing spaces. It SHALL NOT be a null or zero length string.

TPKG-T 67 If present, the `TemplateComponentDetailedDescription` SHALL NOT contain leading or trailing spaces. It SHALL NOT be a null or zero length string.

TPKG-T 68 The `TemplateComponentType` SHOULD contain one of `Definition`, `Definition-Information`, `Definition-Inclusion`, `Definition-Alternative`, `Validation`, `Validation-Inclusion`, `Validation-Information`, `Validation-Report`, `Transform-Display`, `Transform-Format`, `Transform-Content`, `Transform-Anonymous` or `Information`.

TPKG-T 69 If present, the `TemplateComponentClass` SHALL contain one of `HumanGeneration`, `HumanInformation`, `HumanValidation`, `MachineGeneration`, `MachineValidation`, or `MachineTransformation`.

TPKG-T 70 The `TemplateComponentMimeType` SHALL contain a representation of a valid Mime Type as defined by [RFC2045] and [RFC2046].

TPKG-T 71 If present, the `TemplateComponentRestriction` SHOULD contain one of `Restricted` or `Open`.

TPKG-T 72 The `TemplateComponentAuthor` SHALL NOT contain leading or trailing spaces. It SHALL NOT be a null or zero length string.

Appendix A Schema

A.1 template-package-metadata.xsd

```
<?xml version="1.0" encoding="utf-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://ns.electronichealth.net.au/tplt/xsd/package/PackageMetadata/1.0"
xmlns:tmd="http://ns.electronichealth.net.au/tplt/xsd/package/PackageMetadata/1.0" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <element name="templatePackageMetadata">
    <complexType>
      <sequence>
        <element name="TemplateID" type="string" minOccurs="1" maxOccurs="1"></element>
        <element name="TemplateName" type="string" minOccurs="1" maxOccurs="1"></element>
        <element name="TemplateVersion" type="integer" minOccurs="1" maxOccurs="1"></element>
        <element name="TemplateDescription" type="string" minOccurs="1" maxOccurs="1"></element>
        <element name="TemplateDetailedDescription" type="string" minOccurs="0" maxOccurs="1"></element>
        <element name="TemplateTypeTypeIdRoot" type="string" minOccurs="0" maxOccurs="1"></element>
        <element name="TemplateTypeTypeIdExtension" type="string" minOccurs="0" maxOccurs="1"></element>
        <element name="TemplateTypeCode" type="string" minOccurs="0" maxOccurs="1"></element>
        <element name="TemplateTypeCodeSystem" type="string" minOccurs="0" maxOccurs="1"></element>
        <element name="TemplateTypeCodeSystemName" type="string" minOccurs="0" maxOccurs="1"></element>
        <element name="TemplateTypeCodeDisplayName" type="string" minOccurs="0" maxOccurs="1"></element>
        <element name="TemplateClass" type="string" minOccurs="1" maxOccurs="1"></element>
        <element name="TemplateFormatType" type="string" minOccurs="1" maxOccurs="1"></element>
        <element name="TemplateFormatVersion" type="string" minOccurs="1" maxOccurs="1"></element>
        <element name="TemplateStatus" minOccurs="1" maxOccurs="1">
          <simpleType>
            <restriction base="string">
              <enumeration value="Approved" />
              <enumeration value="Active" />
              <enumeration value="Retired" />
            </restriction>
          </simpleType>
        </element>
        <element name="TemplateStatusEffectiveDate" type="dateTime" minOccurs="1" maxOccurs="1"></element>
        <element name="TemplateNextStatusChange" type="dateTime" minOccurs="0" maxOccurs="1"></element>
        <element name="TemplateNextStatus" minOccurs="0" maxOccurs="1">
          <simpleType>
            <restriction base="string">
              <enumeration value="Approved" />
              <enumeration value="Active" />
              <enumeration value="Retired" />
```

```
            </restriction>
          </simpleType>
        </element>
        <element name="TemplateCustodian" type="string" minOccurs="1"
maxOccurs="1"></element>
        <element name="TemplateAdministrator" type="string" minOccurs="1"
maxOccurs="1"></element>
        <element name="TemplateConformanceLevel" type="string" minOccurs="0"
maxOccurs="1"></element>
        <element name="TemplateSupersedingId" type="string" minOccurs="0"
maxOccurs="1"></element>
        <element name="TemplateSupersededId" type="string" minOccurs="0"
maxOccurs="1"></element>
        <element name="TemplateKeyword" type="string" minOccurs="0"
maxOccurs="unbounded"></element>
      </sequence>
    </complexType>
  </element>
</schema>
```

A.2 template-package-manifest.xsd

```
<?xml version="1.0" encoding="utf-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://ns.electronichealth.net.au/tplt/xsd/package/PackageMan
ifest/1.0"
xmlns:tmf="http://ns.electronichealth.net.au/tplt/xsd/package/PackageManifest/
1.0" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <simpleType name="nonEmptyString">
    <restriction base="string">
      <minLength value="1" />
    </restriction>
  </simpleType>
  <element name="templatePackageManifest">
    <complexType>
      <sequence minOccurs="1" maxOccurs="unbounded">
        <element name="templateComponent">
          <complexType>
            <sequence>
              <element name="TemplateComponentFile" minOccurs="1"
maxOccurs="1" type="tmf:nonEmptyString" />
              <element name="TemplateComponentID" minOccurs="1" maxOccurs="1"
type="tmf:nonEmptyString" />
              <element name="TemplateComponentName" minOccurs="1"
maxOccurs="1" type="tmf:nonEmptyString" />
              <element name="TemplateComponentDescription" minOccurs="1"
maxOccurs="1" type="tmf:nonEmptyString" />
              <element name="TemplateComponentDetailedDescription"
minOccurs="0" maxOccurs="1" type="tmf:nonEmptyString" />
              <element name="TemplateComponentType" minOccurs="1"
maxOccurs="1" type="tmf:nonEmptyString" />
              <element name="TemplateComponentClass" minOccurs="0"
maxOccurs="1">
                <simpleType>
                  <restriction base="string">
                    <enumeration value="HumanGeneration" />
                    <enumeration value="HumanInformation" />
```

```
                    <enumeration value="HumanValidation" />
                    <enumeration value="MachineGeneration" />
                    <enumeration value="MachineValidation" />
                    <enumeration value="MachineTransformation" />
                  </restriction>
                </simpleType>
              </element>
              <element name="TemplateComponentMimeType" minOccurs="1"
maxOccurs="1" type="tmf:nonEmptyString" />
              <element name="TemplateComponentRestriction" minOccurs="0"
maxOccurs="1" type="tmf:nonEmptyString" />
              <element name="TemplateComponentAuthor" minOccurs="1"
maxOccurs="1" type="tmf:nonEmptyString" />
            </sequence>
          </complexType>
        </element>
      </sequence>
    </complexType>
  </element>
</schema>
```

Appendix B Acronyms and Terminology

B.1 Acronyms

Acronym	Explanation
OID	Object Identifier
PCEHR	Personally Controlled Electronic Health Record
WSDL	Web Service Definition Language
WSP	Web Service Profile – Commonly used to refer to the ATS-5820 Web Service Profile.
XSD	XML Schema Definition

B.2 Specialised Terminology

Term	Explanation
NASH certificate	A NASH certificate is a digital certificate that is compliant with the NASH certificate policies.
Service	A service encapsulates the collaboration which occurs between two or more parties to achieve a goal. Each participant in the service may offer multiple service interfaces.
Service interface	A service interface is a logical grouping of operations which be offered by a participant within the context of a service.
Service operation	A service operation is a specific function which supports communication between two participants.

Appendix C References

Tag	Name	Version Release Date
[ISO11179]	ISO/IEC 11179 – Information technology - Metadata registries, Parts 1 to 6 http://www.iso.org	Second Edition 15 September 2004
[ISO19757]	ISO/IEC 19757-3 Information technology – Document Schema Definition Languages (DSDL) – Part 3: Rule Based validation – Schematron http://www.iso.org	First Edition 01 September 2006
[LSS2011]	Template Service Interface Logical Service Specification	1.1 April 2011
[PCEHR_CON_OPS]	Concept of Operations: relating to a Personally Controlled Electronic Health Record System http://www.yourhealth.gov.au/internet/yourhealth/publishing.nsf/Content/PCEHR-document	September 2011 Release
[PK2007]	APPNOTE.TXT - .ZIP File Format Specification http://www.pkware.com/documents/casestudies/APPNOTE.TXT	Version 6.3.2 Release 28 September 2007
[RFC2045]	IETF, *RFC 2045: Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, N. Freed, Innosoft, N. Borenstein http://www.ietf.org/rfc/rfc2045.txt	November 1996
[RFC2046]	IETF, *RFC 2046: Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*, N. Freed, Innosoft, N. Borenstein http://www.ietf.org/rfc/rfc2046.txt	November 1996
[RFC2119]	IETF, *RFC 2119: Keywords for use in RFCs to Indicate Requirement Levels*, S. Bradner http://ietf.org/rfc/rfc2119.txt	March 1997
[TSS2011]	Template Service Interface Technical Service Specification	1.2 April 2011
[UML2010]	http://www.omg.org/spec/UML/2.3/	UML Version 2.3 Release May 2010
[W3CCSS]	W3C Recommendation Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) http://www.w3.org/TR/CSS21/	07 June 2011
[W3CXSD]	W3C Recommendation XML Schema http://www.w3.org/standards/techs/xmlschema#w3c_all	28 October 2004
[W3CXLST]	W3C Recommendation XSL Transformations (XSLT) Version 1.0 http://www.w3.org/TR/xslt	1.0 16 November 1999