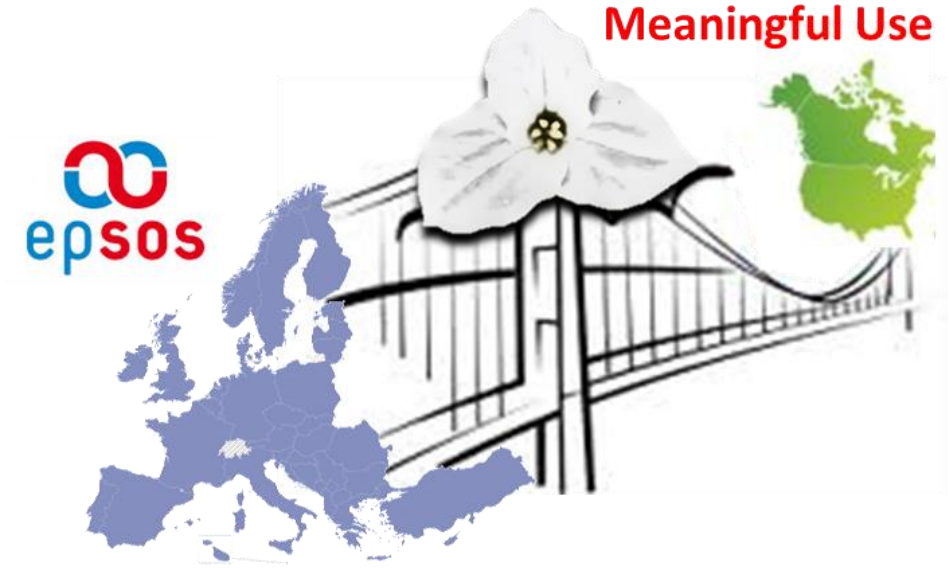
# Trillium Bridge

## Bridging Patient Summaries across the Atlantic




WP 3 – Assembling Interoperability Assets

Deliverable 3.2

# EU/US CTS-2 Infrastructure with selected Transcoding, Translation and Terminology Mappings for pilot use cases - Transformer

***Version: 1.3***

***Date of Issue: September 21, 2015***

## Document Information

| | |
|---|---|
| **Deliverable name (dc:title):** | **EU/US CTS-2 Infrastructure with selected Transcoding, Translation and Terminology Mappings for pilot use cases – Transformer** |
| **Deliverable No. (dc:identifier)** | **D3.2** |
| **Work Package** | **WP3: Assembling Interoperability Assets** |
| **Date of Issue (dc:date.issued):** | **March 2, 2015** |
| **Status** | **Final** |
| **Version (dc:relation.hasVersion)** | **1.3** |
| **Replace (dc:relation.replaces):** | **n/a** |
| **File Name** | **FP7-SA610756-D3.2-v1.3** |
| **Nature**[1] **(dc:type)** | **Other – Companion to the Transformer software package** |
| **Disseminiation Level**[2] **(dc:accessRights)** | **PU – Public** |

| | Name | Organization |
|---|---|---|
| **Responsibile (dc:publisher):** | **Catherine Chronaki** | **HL7 Foundation** |
| **Responsibile (dc:publisher):** | **Harold Solbrig** | **Mayo Clinic** |
| **Author (dc:contributor.creator):** | **Kevin Peterson** | **Mayo Clinic** |
| **Autore (dc: contributor.creator):** | **Giorgio Cangioli** | **HL7 Foundation** |
| **Contributor (dc:contributor):** | | |
| **Contributor (dc:contributor):** | | |
| **Contributor (dc:contributor):** | | |
| **Contributor (dc:contributor):** | | |
| **Contributor (dc:contributor):** | | |
| **Contributor (dc:contributor):** | | |
| **Contributor (dc:contributor):** | | |
| **Contributor (dc:contributor):** | | |
| **Contributor (dc:contributor):** | | |

[1] Please indicate the nature of the deliverable using one of the following codes:**R** = Report, **P** = Prototype, **D** = Demonstrator, **O** = Other

[2] Please indicate the dissemination level using one of the following codes: **PU** = Public **PP** = Restricted to other programme participants (including the Commission Services). **RE** = Restricted to a group specified by the consortium (including the Commission Services). **CO** = Confidential, only for members of the consortium (including the Commission Services).

## Document History

| Date | Vers. | Author | Change | Status |
|---|---|---|---|---|
| Feb 12, 2015 | 0.1 | Catherine Chronaki | Table of Contents | Draft |
| Feb 25, 2015 | 0.2 | Catherine Chronaki/Harold Solbrig | Methodology | Draft |
| Feb 27, 2015 | 0.4 | Catherine Chronaki/Harold Solbrig | Intro, Architecture, Demonstration, Limitations, Coclusions, Glossary | Draft |
| March 1, 2015 | 1.0 | Catherine Chronaki | Editorial Review | Final Draft |
| March 2, 2015 | 1.1 | Giorgio Cangioli | Review of Final Draft | Final Draft for External Review |
| March 2, 2015 | 1.2 | Harold Solbrig | Review of Final Draft | Final Draft for external review |
| September 21, 2015 | 1.3 | Catherine Chronaki | Corrections suggested by reviewers | Final |

# Table of Contents

# 1 Executive Summary

This document serves as a companion to the software package of the Transformer, which serves as a proof of concept for the transformation of patient summaries conforming to the European Patient Summary Guideline (EU PS) that has been adopted for the cross-border exchange of Patient Summaries among Member States in the European Union and HL7 C-CDA CCD (CCD) specification that has been adopted for Blue Button+ in the United States.

The development of the Transformer software package has evolved out of the work of Trillium Bridge Deliverables *D3.1* and *D3.*1 *addendum* that documented most of the transformations and transcoding necessary when converting a European patient summary to a United States one and vise versa. The Transformer also uses the STS2 service developed by Phast (based on the CTS2 standard service) which offers the value sets and best effort mappings between the value sets used in epSOS and the ones used in CCD for clinically equivalent sections. All the code systems, value sets and maps are available (login required) at http://extension.phast.fr/STS_UI. The mappings are delivered to the transformer by the standard web service interface of CTS2. If a mapping does not exist the event is logged both at the CTS2 and in the transformed document.

The Transformer is structured as a table driven XSLT 2.0 library. The actual transformations are based on XPATH expressions in a mapping table that is defined in the transformation schema (TBXform.xsd). The Transformer software has been tested with the a small number patient summary examples that were derived from selected patient stories developed in WP2 and presented in D2.1/D2.2 and used in demonstrations of Trillium Bridge in Europe and the United States.

The aim of the transformer software is to support the demonstration of patient summary exchange as part of the overall Trillium Bridge demonstrations at HIMSS, the eHealthWeek, and elsewhere to collect evidence on the limitations of transforming patient summaries that are fit for the purpose of use in the transatlantic setting and to inform the Trillium Bridge feasibility study. Furthermore, it is the intent of the Trillium Bridge team that the Transformer software is delivered in open source to serve as a shared interoperability asset that is further improved and elaborated upon.

The most recent version of the transformer code is available on github: https://github.com/trillium-bridge/trillium-bridge-transformer

"**Disclaimer**: It should be noted that the transformer has been tested against a limited number of samples and supersedes D3.1. As a result, only part of the transformations listed in D3.1 have been implemented/tested in the transformer. Moreover the implementation choices in the transformer do not always follow the design suggestions of D3.1. The transformer is a live open source distribution. For the latest version please check the GitHub at http://informatics.mayo.edu/trillium-bridge" .

# 2 Introduction

The work presented in Deliverables *D3.1* and 3.1 addendum, provides a list of the syntactic and value set transformations that are necessary to transform syntactically and semantically a patient summary according to the epSOS Patient Summary (ePSOS PS) that EU Patient Summary Guideline is referring to[3] to a clinical patient summary conforming to the HL7 CCDA CCD specification and vice versa, covering the sections that are clinically in correspondance.

The rest of the document is structured as follows: Section 3 presents the objectives of the Transformer software and this accompanying document. Section 4 provides a glossary of common terms. Section 5 describes the methodological approach. Section 6 describes the architecture of the transformer and the key functions, for those interested in extending it. Section 7 provides a copy of the user guide, which will in the future be updated online as a companion to the code of the transformer. Section 8 demonstrates the functionality of the Transformer web site allowing the transformation of patient summary documents, optionally using Bing for automatic text translation[4]. Section 9 presents the limitations of this work and proposes directions for future extensions and improvements.Finally, section 10 presents our conclusions.

[3] Actually, the transformation is to the epSOS Patient Summary (PS) Implementation Guide (IG), which is the portion of the patient summary that has implemented to date.

[4] The free text translation is out of scope of this project, this as been provided as reference implementation for future evolution.

# 3 Objectives

The objectives of this document is to accompany the transformer software release serving as a layman's introduction to:

- Demonstrate the feasibility of the transformation for limited number of representative patient summary samples to identify limitations and reflect on the feasibility of the overall effort
- Set up a framework for progressive development and refinement of the transformer function by the open source community
- Draw Recommentations about tooling and infrastructure to allow and to scale across a broad variety of clinical document templates and associated value sets looking forward to further development and progressive elaboration.

# 4 Glossary

**CTS2**: Comon Terminology Services 2 (CTS2) is a joint HL7/Object Management Group standard that specifies a model and API for the discovery, access, distribution of federated terminological resources on the internet. The standard can be found at http://www.omg.org/spec/cts2/and implementation iinformation at: http://informatics.mayo.edu/cts2/

**GitHub**: GitHub is a web-based softwer repository hosting service, which offers distributed revision control and source code management (SCM) functionality. GitHub provides a web-based graphical interface and desktop as well as mobile integration. It also provides access control and several collaboration features such as wikis, task management, bug tracking and feature requests for every project. It is based and extends Git, a command-line tool with similar functionality. Information about GitHub can be found at https://github.com/and the Trillium Bridge Transformer GitHub project at: https://github.com/trillium-bridge/trillium-bridge-transformer

**STS2 Service**: A service developed by Phast, which offers the value sets used in the epSOS patient summary and CCD as well as best effort mappings between value sets. Information about PHAST can be found at http://www.phast.fr/index.php, and the STS CTS2 services at http://extension.phast.fr/STS_UI.

**XPATH:** XPath, the XML Path Language, is a query language for selecting nodes from an XML document. XPath is used to navigate through elements and attributes in an XML document. XPath is a major element in W3C's XSLT standard. XPath is also used in XQuery and XPointer. See http://www.w3.org/TR/xpath20/for further information.

**XSLT:** XSLT is a language for transforming XML documents into XHTML documents or to other XML documents. See also: http://www.w3.org/TR/xslt20/

# 5 Methodology

The Trillium Transformer builds on the work presented in *D3.1 section 6-8*, but goes beyond to offer a general table-driven solution starting from the rules include in *D3.1/D3.*1 addendum. The Trillium Bridge Transformer software adresses transformation and transcoding of elements that have HL7 CE or CD data type[5] and which elements have clinical equivalence on both sides, i.e. sections can be identified on both sides with the same semantic meaning. Data elements that are mapped directly with fixed XML fragments, values, or translations are also convered. For those cases, and particularly in the case of translation the solution presented is not by any means complete. It is just a proof of concept for future extensions to be built upon.

The transformer calls the CTS2 server and performs a transcoding if necessary. The header and clinical sections in the two specifications were compared and the coded data elements that were found in correspondence were identified in *D3.1 section 7*, which listed in detail the necessary transformations for each section with respect to syntax and value sets.

For each of the proposed transformations, a set of rules and one or more table entries were created, if a change was indeed needed. When the data element is to be transformed the *templateID* usually changes. Even if the structure of the data element does not need to change, the *templateID* will change as appropriate from one of epSOS PS to CCD or vice-versa. The combination of syntax (structure of the the discrete data) and semantics (the value sets used for a particular data element) give four possibilities:

1. The structure is the same, and the value sets are identical: there is no transformation needed (except for replacing any *template IDs*). In this case, the transformer does copies the information verbatim.
2. The structure is the same, and the value sets have a mapping: there is no syntactic transformation needed (except for replacing any *template IDs*), but the data element will have a value obtained from the mapping. In this case, the transformer translate the *code* or *value* to the target system.
3. The structure is different, and the value sets are identical: there is a transformation needed, but the value of the data element stays the same. This category also includes the case where the implemented (XML) structure is used in a different way (e.g the same information is represented as observation.code in one template and as observation.value in the other as for example the case of allergies section). The transformer carries a variety of functions for moving, renaming and changing the content of the target sections
4. The structure is different, and the value sets have a mapping: a transformation is needed, and the data element will have a value obtained from the mapping.

The transformation rules are written only for those elements that can draw onto the mapping existing in the CTS server. However, provisions are made on how to handle exceptions such as those cases where data are not present, available or known. The translation from the language another European Member State to English and from English to the language of another Member State is part of the normal operation of the National Contact Point (NCP), and is typically not within the functional perimeter of the Transformer. However, the Transformer does offer the option of automatic translation through the Microsoft Bing translation service even though this strictly out of scope for Trillium Bridge.

Each section in D3.1/Section 7 contains a "Transform" part which lists the corresponding Xpaths for the data element. The transformation converts these XPaths into table entries to generate two unidirectional transformation epSOS PS->CCD and CCD->epSOS PS. There are potentially two forms of codes that may undergo mapping: the first involves elements that are explicitly identified as CD or CD-derived types; the second is a generic value, where the type is identified at runtime as being CD or a derivative.

[5]Elements of CE and CD data type are associated with a value set from which a value can be chosen for that particular data element with or without exceptions.

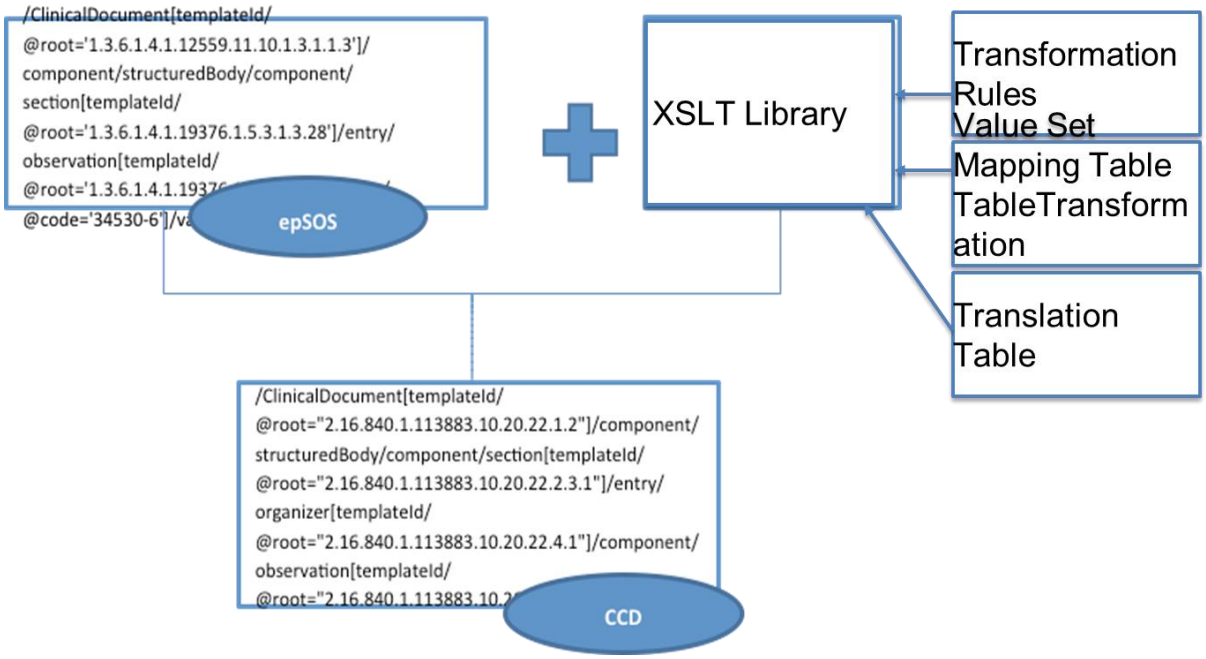



**Figure 1: Example of the transformation from the epSOS document fragment to CCD.**

As decided early on during the preparation of Deliverable *D3.1,* the root CD is replaced with the mapped equivalent and the original code is represented as a translation[6]. This decision highlights a basic distinction between *translation* and *transcoding/mapping*: a *translation* of a code in a given CD instance is not the same as a *mapping* from one CD instance to a different but related instance.

The second decision we had to make related to codes that have no maps. In that regard the decision was to formally indicate that no transformation is available. The event should be logged at the CTS2 infrastructure and be addressed operationally:

```
<value xsi:type="CD" displayName="Nausea" nullFlavor="NI">
    <translation code="R11.0" codeSystem="2.16.840.1.113883.6.3" displayName="Nausea"/>
</value>
```

Although the "UNC" null flavor would be the most applicable, this is not a part of the null flavor value set used in the 2005 CDA normative edition, for that reason the generic null flavor "No information" is used in this context..

The next section outlines the actual structure of the transformer XSLT library. All the code for the transformer is available at: https://github.com/trillium-bridge/trillium-bridge-transformer

## 5.1 Technical approach

The key elements of the technical approach are as follows:

1) Value Set Maps are loaded into PHAST server and exposed as CTS2 Maps
2) Correspondence of Paths described in Deliverable D3.1 are converted into entries in the transformation rules table (Note -- transformations that have no change are not included)
3) Function described in Deliverable *D3.1* are implemented in the XSLT Function Library, and are invoked from entries in the transformation rules.

[6] The ISO 21090 specification provides an additional field, codingRationale, that allows one to state whether a code or its translation is original, post-coded from free text, required, etc., but this attribute is not available in the model currently used.

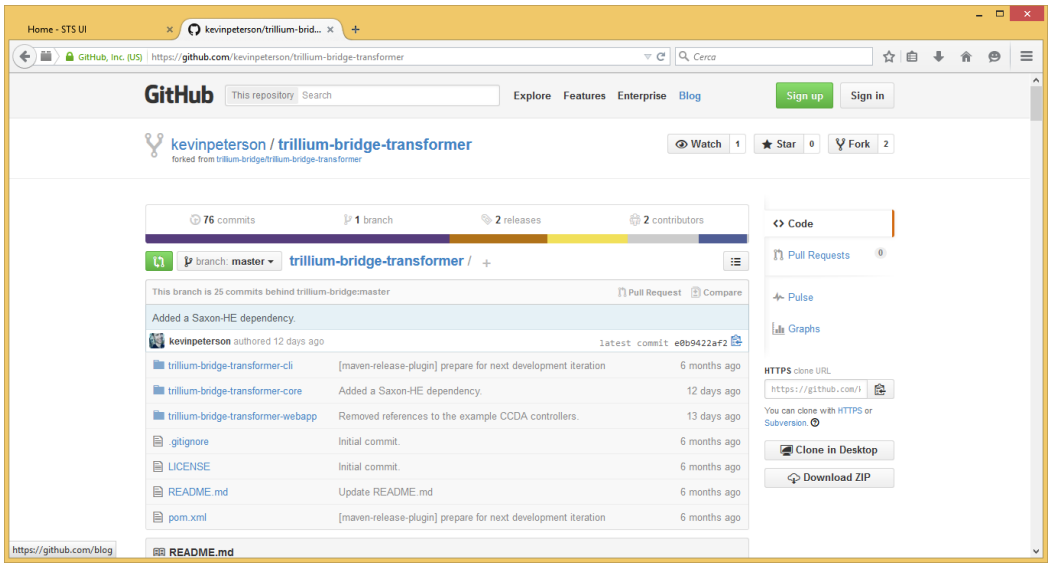



**Figure 2: The source code of the transformer is available on GitHub in open source.**

# 6 Architecture of the Transformer

## 6.1 Contexts, Tranformations, and Paths

### 6.1.1 Contexts

A context establishes a node that all internal transformations apply to. In the figure below the global context is that we are transforming from epSOS to CCD. This particular context applies to documents with a template id of 1.3.6.1.4.1.12559.11.10.1.3.1.1.3. The subsequent tansformations i.e. *translateTitle*, *changeTemplateRoots, etc.* apply to nodes within the document.

```
<context from="epSOS" to="CCD">
    <documentation>Entry point for the epSOS to CCD transformation (and visa-versa)</documentation>
    <root>/ClinicalDocument[templateId/@root="1.3.6.1.4.1.12559.11.10.1.3.1.1.3"]</root>
    <transform global="true">
        <documentation>All titles get location independent transformation</documentation>
        <path>/title</path>
        <transformation name="translateTitle"/>
    </transform>
    <transform>
        <documentation>The root template identifier needs to be changed from epSOS to CCD</documentatior
        <path>/templateId</path>
        <transformation name="changeTemplateRoots">
            <arg fromid="1.3.6.1.4.1.12559.11.10.1.3.1.1.3"/>
            <arg fromid="1.3.6.1.4.1.19376.1.5.3.1.1.1"/>
            <arg toid="2.16.840.1.113883.10.20.22.1.2"/>
            <arg toid="2.16.840.1.113883.10.20.22.1.1"/>
        </transformation>
    </transform>
    <transform>
```

Figure 3: The outer context of a document.

Contexts can be nested. The context root can be relative to an outer context, where the template id transformations apply to relative paths within the surrounding context.

```
<!-- ======================= allergies-adverse-reactions-alerts ======================= -->
<context>
    <root>/component/section[templateId/@root="2.16.840.1.113883.10.20.1.2"]</root>
    <transform>
        <path>/templateId</path>
        <transformation name="changeTemplateRoots">
            <arg fromid="2.16.840.1.113883.10.20.1.2"/>
            <arg fromid="1.3.6.1.4.1.19376.1.5.3.1.3.13"/>
            <arg toid="2.16.840.1.113883.10.20.22.2.6"/>
            <arg toid="2.16.840.1.113883.10.20.22.2.6.1"/>
        </transformation>
    </transform>
    <transform>
        <path>/text</path>
        <transformation name="translateText"/>
    </transform>
    <context>
        <root>/entry[@typeCode="DRIV"]/act[templateId/@root="2.16.840.1.113883.10.20.1.27"]</root>
        <transform>
            <path>/templateId</path>
            <transformation name="changeTemplateRoots">
```

Figure 4: Nested Inner Contexts.

## 6.2 Global Transformations

Global transformations apply anywhere within the parent context. The example applies *translateTitle* transformation to any title within the surrounding ("global") context.

```
<transform global="true">
    <documentation>All titles get location independent transformation</documentation>
    <path>//title</path>
    <transformation name="translateTitle"/>
</transform>
```

Figure 5: Example of a transformation applied everywhere within a context.

## 6.3 XPath expressions in use

For the purpose of the transformer, nodes with template identifiers are represented as: ***nodeName*[templateId/@root="*templateid*"]/.... ,** where *templateId* references the first template identifier within the node[7]. As an example, the XML fragment:

```
<!-- Allergies Section (entries required)  -->
<component>
    <section>
        <!-- Allergies section with entries optional templateId -->
        <templateId root="2.16.840.1.113883.10.20.22.2.6"/>
        <!-- Allergies section with entries required templateId -->
        <templateId root="2.16.840.1.113883.10.20.22.2.6.1"/>
        <code code="48765-2" codeSystem="2.16.840.1.113883.6.1"/>
```

could be referenced using the path:

**/component/section[@templateId/@root="2.16.840.1.113883.10.20.22.2.6"]**

Nodes with typeCodes are represented as ***nodeName* [@typeCode="..."]/....** As an example the fragment:

```
<entry typeCode="DRIV">
    <!-- Allergy Problem Act -->
```

could be referenced with the path:

**/entry[@typeCode="DRIV"]**

Nodes with classCodes are represented as ***nodeName*[@classCode="..."]/....** As an example, the playingEntity element below:

```
<!-- This particpant represents the causative agent -->
<participant typeCode="CSM">
    <participantRole classCode="MANU">
        <playingEntity classCode="MMAT">
            <code code="314422" displayName="ALLERGENIC EXT
```

would be referenced with:

**/participant[@typeCode="CSM"]/participantRole[@classCode="MANU"]/playingEntity[@classCode="MMAT"]**

[7] We realize that this is not a final solution, as there is nothing that asserts that template identifiers have to appear in a particular order within a CCDA document. Future implementations will address this issue by allowing any on the template identifiers within an element to reference the element.

A node that has both a *templateId* and a *typeCode* or a *classCode* is represented with just the *templateId.* As an example the observation element below:

```
<entryRelationship typeCode="SUBJ" inversionInd="true">
    <!--  Allergy status observation  -->
    <observation classCode="OBS" moodCode="EVN">
        <templateId root="2.16.840.1.113883.10.20.22.4.28"/>
        <code code="33999-4" codeSystem="2.16.840.1.113883.6.
```

would be referenced by the path:

**/entryRelationship[@typeCode="SUBJ"]/observation[templateId/@root="2.16.840.1.113883.10.20.22.4.28"]**

All other node paths are just ***nodeName***.

The transformation tool has an option that produces the absolute and relative paths for each node in a ducment that can be used for transformation creation and debugging. As an example, the last element above would produce:

```
                        <!--  Allergy status observation  -->
                        <!--  FULL PATH:
/ClinicalDocument[templateId/@root="2.16.840.1.113883.10.20.22.1.1"]/component/structuredBody/component/section[templateId
/@root="2.16.840.1.113883.10.20.22.2.6"]/entry[@typeCode="DRIV"]/act[templateId/@root="2.16.840.1.113883.10.20.22.4.30"]/e
ntryRelationship[@typeCode="SUBJ"]/observation[templateId/@root="2.16.840.1.113883.10.20.22.4.7"]/entryRelationship[@typeC
ode="SUBJ"]/observation[templateId/@root="2.16.840.1.113883.10.20.22.4.28"]
     LOCAL PATH: /entryRelationship[@typeCode="SUBJ"]/observation[templateId/@root="2.16.840.1.113883.10.20.22.4.28"]  -->
                        <observation classCode="OBS" moodCode="EVN">
```

## 6.4 Transformer functions

The transformation functions are documented in *TBXform.xsd*, which is present in the schema directory of the Transformer software distribution. Here they are presented to give an idea of the architecture of the transformer and its potential for extension.

### 6.4.1 *changeTemplateRoots* — remove and/or add template identifiers

This function does two things:

- remove all *fromid* entries
- insert all *toid* entries

```
<transform>
    <documentation>The root template identifier needs to be changed from epSOS to CCD</documentation>
    <path>/templateId</path>
    <transformation name="changeTemplateRoots">
        <arg fromid="1.3.6.1.4.1.12559.11.10.1.3.1.1.3"/>
        <arg fromid="1.3.6.1.4.1.19376.1.5.3.1.1.1"/>
        <arg toid="2.16.840.1.113883.10.20.22.1.2"/>
        <arg toid="2.16.840.1.113883.10.20.22.1.1"/>
    </transformation>
</transform>
```

Figure 6: The original *template ids* are replaced with new ones.

The above rule would transform the input:

```
<ClinicalDocument>
    <typeId extension="POCD_HD000040" root="2.16.840.1.113883.1.3"/>
    <templateId root="1.3.6.1.4.1.12559.11.10.1.3.1.1.3"/>
    <templateId root="1.3.6.1.4.1.19376.1.5.3.1.1.1"/>
    <id extension="TB1" root="2.16.840.1.113883.19.5.99999.1"/>
    <code code="60591-5" displayName="Patient Summary" codeSystem="2.16.8
```

to:

```
<ClinicalDocument>
    <typeId extension="POCD_HD000040" root="2.16.840.1.113883.1.3"/>
    <templateId root="2.16.840.1.113883.10.20.22.1.2"/>
    <templateId root="2.16.840.1.113883.10.20.22.1.1"/>
    <id extension="TB1.1" root="2.16.840.1.113883.19.5.99999.1"/>
    <code codeSystem="2.16.840.1.113883.6.1"
```

### 6.4.2 *newid* — generate a new document identifier

This function creates a new *document id* for the transformed patient summary by:

- Suffix or otherwise transform a document identifier to render it unique
- Current suffix is append a ".1" to extension.

The current default choice was the ".1", although other are also available.

```
<transform>
    <documentation>All documents need a new identifier </documentation>
    <path>/id</path>
    <transformation name="newid"/>
</transform>
```

Figure 7: Sample of a *newid* function in the transformation table.

The transformation aboove would the document identifier:

```
<id extension="TB1" root="2.16.840.1.113883.19.5.99999.1"/>
```

to:

```
<id extension="TB1.1" root="2.16.840.1.113883.19.5.99999.1"/>
```

### 6.4.3 *translateTitle* — translate a section title

At the moment, both *translateTitle* and *translateText* work with text, the mapping text of a title is based on *from* and *to* languages XML documents. Fixed translations live in tables in the translation directory with the name *translation/{from}to{to}.xml* document. Only the first two characters of the language code is used. As an example, translations from the language, "*it-IT*" to to "*en*" are in the table named "*ittoen.xml*" in the translations directory.

```
<context from="epSOS" to="CCD">
    <documentation>Entry point for the epSOS to CCD transformation (and visa-versa)</documentation>
    <root>/ClinicalDocument[templateId/@root="1.3.6.1.4.1.12559.11.10.1.3.1.1.3"]</root>
    <transform global="true">
        <documentation>All titles get location independent transformation</documentation>
        <path>/title</path>
        <transformation name="translateTitle"/>
    </transform>
```

Figure 8: Sample title transformation.

The translation above, when applied to CCDA document that is being transformed from Italian to English would use the following entry in the "ittoen.xml" table:

```
<?xml version="1.0" encoding="UTF-8"?>
<translations xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocati
    xmlns="http://trilliumbridge.org/xform" fromlanguage="it-IT" tolanguage="en">
    <entry>
        <source>Profilo Sanitario Sintentico (Patient Summary)</source>
        <target>Patient Summary</target>
    </entry>
    <entry>
        <source>Allergie, Reazioni Avverse ed Allarmi</source>
        <target>Allergies, adverse reactions, alerts</target>
    </entry>
    <entry>
```

Figure 9: Translation Table *it-ITtoen.xml*

to transform the input:

```
</code>
<title>Profilo Sanitario Sintentico (Patient Summary)</title>
<effectiveTime value="20140501"/>
```

into:

```
<title>Patient Summary</title>
<effectiveTime value="20140501"/>
```

In the future, titles may be associated with language and template ids, with tables named appropriate. Also note that, if the Microsoft Bing translation option is enabled, titles that aren't in the translation table will be translated using the translation service instead.

### 6.4.4 *replaceCode* — remove and/or add a code node

ReplaceCode removes or adds existing codes as shown in the example below. This is used when you wish specifically to insert / remove explicitly a code as directed in a template.

```
<transform>
    <path>/code</path>
    <transformation name="replaceCode">
        <arg>
            <fromcode>
                <code nullFlavor="NA" xmlns="urn:hl7-org:v3"/>
            </fromcode>
            <tocode>
                <code code="48765-2" codeSystem="2.16.840.1.113883.6.1" codeSystemName="LOINC"
                    displayName="Allergies, adverse reactions, alerts" xmlns="urn:hl7-org:v3"/>
            </tocode>
        </arg>
    </transformation>
</transform>
```

Figure 10: *ReplaceCode* replaces unavailable code with a specific loinc code.

The above example, when applied to:

```
<act classCode="ACT" moodCode="EVN">
    <templateId root="2.16.840.1.113883.10.20.1.27"/>
    <templateId root="1.3.6.1.4.1.19376.1.5.3.1.4.5.1"/>
    <templateId root="1.3.6.1.4.1.19376.1.5.3.1.4.5.3"/>
    <id extension="allergy.1" root="2.16.470.1.100.1.1.1000.990.1"/>
    <code nullFlavor="NA"/>
    <statusCode code="active"/>
```

will produce:

```
<act classCode="ACT" moodCode="EVN">
    <templateId root="2.16.840.1.113883.10.20.22.4.30"/>
    <id extension="allergy.1" root="2.16.470.1.100.1.1.1000.990.1"/>
    <code code="48765-2"
          codeSystem="2.16.840.1.113883.6.1"
          codeSystemName="LOINC"
          displayName="Allergies, adverse reactions, alerts"/>
```

### 6.4.5 *replaceValue* — remove and/or add a value node

*ReplaceValue* offers the same functionality as ReplaceCode for Values.

```
<transform>
    <path>/value</path>
    <transformation name="replaceValue">
        <arg>
            <fromValue>
                <value xsi:type="CD" xmlns="urn:hl7-org:v3">
                    <originalText>
                        <reference value="#allergy.1"/>
                    </originalText>
                </value>
            </fromValue>
            <toValue>
                <value xsi:type="CD" code="416098002" displayName="Drug allergy" codeSystem="2.16.840.1.113883.6.96"
                    codeSystemName="SNOMED CT" xmlns="urn:hl7-org:v3"/>
            </toValue>
        </arg>
    </transformation>
</transform>
```

Figure 11: Example of replacing an allergy reference code with a specific SNOMED CT code.

The above transformation, when applied to:

```
</effectiveTime>
<value xsi:type="CD">
    <originalText>
        <reference value="#allergy.1"/>
    </originalText>
</value>
<participant typeCode="CSM">
```

produces:

```
</effectiveTime>
<value xsi:type="CD"
       code="416098002"
       displayName="Drug allergy"
       codeSystem="2.16.840.1.113883.6.96"
       codeSystemName="SNOMED CT"/>
<participant typeCode="CSM">
```

### 6.4.6 *translateText* — translate a text section (stub)

This is a hook for extending the transformer in the future to support translation with a flag. For the sake of demonstration, we have used Bing translation as an option to the online interface.

```
<transform>
    <path>/text</path>
    <transformation name="translateText"/>
</transform>
```

Figure 12: *TranslateText* function.

This function goes through the XML elements finds strings in XML values and attempts to take from *language1* to *language2*. First it tries the Translation table and if an exact match is found replaces the text. If it does not find translation then it optionally calls BING. It does not support code 'embedding'.

```
<text>
   <paragraph>Original Text</paragraph>
   <br/>
   <text>
                        <table border="1">
                              <tbody>
                                    <tr>
                                          <th>Descrizione del Problema</th>
                                          <th>Data Insorgenza</th>
                                          <th colspan="2">Diagnosi</th>
                                    </tr>
                                    <tr ID="problem.1">
                                          <td/>
                                          <td>2008</td>
                                          <td>I10</td>
                                          <td ID="des.prob.1">Ipertensione Essenziale Non Specificata</td>
                                    </tr>
                              </tbody>
                        </table>
                  </text>
   <paragraph>Warning: this translation has been generated by a software component</paragraph>
   <br/>
                     <table border="1">
                           <tbody>
                                 <tr>
                                       <th>Active Problem</th>
                                       <th>Onset Date</th>
                                       <th colspan="2">tttttttt</th>
                                 </tr>
                                 <tr ID="problem.1">
                                       <td/>
                                       <td>2008</td>
                                       <td>t10</td>
                                       <td ID="des.prob.1">Systolic essential hypertension</td>
                                 </tr>
                           </tbody>
                     </table>
                  <br/>
</text>
```

**Figure 13: In this version of *translateText*, in case of failure the text is converted to "tttttt…"**

### 6.4.7 *addNode* — add node before or after a matching path

This function adds an XML fragment, i.e. set of XML elements before or after a matching path. As an example, the figure below adds a *results* component as the last element in the document. This is reflected by the argument *after*="1". If the argument was *before*="1" the node would be placed before the matching path, which in the example below is "*/component*". There is also the option of adding within or outside the matching path by including or excluding the argument "*outside*".

```
<root>/component/structuredBody</root>

<transform>
    <path>/component</path>
    <transformation name="addNode" outside="1">
        <arg after="1">
            <component xmlns="urn:hl7-org:v3">
                <section nullFlavor="NI">
                    <templateId root="2.16.840.1.113883.10.20.22.2.3"/>
                    <templateId root="2.16.840.1.113883.10.20.22.2.3.1"/>
                    <code code="30954-2" codeSystem="2.16.840.1.113883.6.1" codeSystemName="LOINC" displayName="RESULTS"/>
                    <title>Results</title>
                    <text>
                        <paragraph>Warning: this section has been generated by a software component</paragraph>
                        <list>
                            <item>No information</item>
                        </list>
                    </text>
                </section>
            </component>
        </arg>
    </transformation>
</transform>
```

**Figure 14: *addNode* adds an XML fragment after the last component in the document.**

*AddNode* also has access to the original document identifier, which allows a related document entry to be constructed. As an example the transformation below adds a related document node that references the extension and root of the original document.

```
<transform>
    <path>/component</path>
    <transformation name="addNode" outside="true">
        <arg before="true">
            <relatedDocument typeCode="XFRM" xmlns="urn:hl7-org:v3">
                <parentDocument>
                    <id extension="{id/@extension}" root="{id/@root}"/>
                </parentDocument>
            </relatedDocument>
        </arg>
    </transformation>
</transform>
```

Figure 15: The use of *addNode* adds a "*relateddocument*" node.

The above transformation transforms an input document containing:

```
        </serviceEvent>
    </documentationOf>
    <component>
        <structuredBody>
            <!--allergies-adverse-reactions-alerts-->
```

and produce:

```
        </serviceEvent>
    </documentationOf>
    <relatedDocument typeCode="XFRM">
        <parentDocument>
            <id extension="TB1" root="2.16.840.1.113883.19.5.99999.1"/>
        </parentDocument>
    </relatedDocument>
    <component>
        <structuredBody><!--allergies-adverse-reactions-alerts-->
```

### 6.4.8 *setStyleSheet* — set the document stylesheet

*setStyleSheet* changes the stylesheet processing instructions included as part of the document. So a different stylesheet is used for presenting the EU patient summary from one used to render the corresponding CCD, this function helps adopt a different stylesheet. The example below will set the stylesheet to "*CCD-en.xsl*" if it exists, otherwise "*CCD.xsl*" assuming the target language is English. However, one can complete omit the option of using a stylesheet when you invoke the transformation.

```
<context from="epSOS" to="CCD">
    <documentation>The stylesheet needs to be mapped from the appropriate type and langauge</documentation>
    <root/>
    <transform>
        <path>processing-instruction/xml-stylesheet</path>
        <transformation name="setStyleSheet">
            <arg value="{to}-{language}" default="{to}"/>
        </transformation>
    </transform>
</context>
```

Figure 16: Setting the stylesheet for the English language.

### 6.4.9 *mapLanguage* — map the document language code

MapLanguage changes the language of the document to the target language. The target language is passed as a parameter to the transformation itself.

```
<transform>
    <documentation>Translate the language</documentation>
    <path>/languageCode</path>
    <transformation name="mapLanguage"/>
</transform>
```

Figure 17: *mapLanguage* changes the language to the target language in the transform document.

As an example, an incoming document in Italian:

```
<title>Profilo Sanitario Sintentico (Patient Summary)</title>
<effectiveTime value="20140501"/>
<confidentialityCode code="D" codeSystem="2.16.840.1.113883.5.2
<languageCode code="it-IT"/>
<!--patient-data-->
```

would be transformed to:

```
<title>Patient Summary</title>
<effectiveTime value="20140501"/>
<confidentialityCode code="D" codeSystem="2.16.840.1.113883.5.25
<languageCode code="en-EN"/>
<!--patient-data-->
```

If the target language were "en-EN".

### 6.4.10 *mapValueSet* — transform a coded value using CTS2

*MapValueSet* maps a value or a coded node using an external CTS2 web service to retrieve the mapping. Each map can be used in a different service or local CTS2 XML document.

```
<transform>
    <path>/value</path>
    <transformation name="mapValueSet" map="ICD_10_SNOMED_CT_epSOSIllnesses_VS"/>
</transform>
```

Figure 18: This transfomation uses the ICD_10 to SNOMED_CT illness map.

The *valuesetmap* table identifies the URL to be used for the individual maps for specific sections. In the figure below, the *CCDtoEPSOSHeader* entry points to a local file with multiple maps, while *ATC_NDF_RT_epSOSActiveIngredient_VS* points to the PHAST CTS2 service. The reason for using a local file is to prefetch parts of the maps to address potential performance overheads. The table approach also allows subsequent implementations to use other/additional CTS2 services as well. The option "*entireMap*" reflects whether the URL provides all map entries or just the one for the supplied code. If a CTS service had support for *entireMap* the client would be able to prefetch the map and cache it locally.

FP7-SA610756-D3.1.xml | TBXform.xsl | ValueSetMaps.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<valuesetmap xmlns="http://trilliumbridge.org/xform"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://trilliumbridge.org/xform ../schema/TBXform.xsd">
    <entry name="CCDtoEPSOSHeader" entireMap="true">
        <uripattern>../cts2fileservice/map/CCDtoEPSOS/version/v0.1/resolution.xml</uripattern>
    </entry>
    <entry name="NUCC_ISCO_ProviderType_VS" entireMap="false">
        <uripattern>http://rd.phast.fr/REST/sts_rest_beta_2/0004/map/NUCC_ISCO_ProviderType_VS/resolution?entity={code}</uripattern>
    </entry>
    <entry name="ATC_NDF-RT_epSOSActiveIngredient_VS" entireMap="false">
        <uripattern>http://rd.phast.fr/REST/sts_rest_beta_2/0004/map/ATC_NDF-RT_epSOSActiveIngredient_VS/resolution?entity={code}</uripattern>
    </entry>
    <entry name="ICD_10_SNOMED_CT_epSOSIllnesses_VS" entireMap="false">
        <uripattern>http://rd.phast.fr/REST/sts_rest_beta_2/0004/map/ICD 10_SNOMED CT_epSOSIllnesses_VS/resolution?entity={code}</uripattern>
```

**Figure 19: Part of the master value set table used in the Trillium Bridge transformer.**

The ICD 10 to SNOMED CT Illness map would transform:

```
<value code="A30.9" displayName="Leprosy, unspecified"
    codeSystem="1.3.6.1.4.1.12559.11.10.1.3.1.44.2" codeSystemName="ICD-10" xsi:type="CD">
    <originalText>
        <reference value="#des.prob.1"/>
    </originalText>
</value>
```

to:

```
<code codeSystem="2.16.840.1.113883.6.96"
      codeSystemName="SNOMED CT"
      code="81004002"
      displayName="Leprosy (disorder)">
   <translation code="A30.9"
                displayName="Leprosy, unspecified"
                codeSystem="1.3.6.1.4.1.12559.11.10.1.3.1.44.2"
                codeSystemName="ICD-10"
                xsi:type="CD">
      <originalText>
         <reference value="#des.prob.1"/>
      </originalText>
   </translation>
</code>
```

The map would produce the following output:

```
<value code="I10" displayName="Essential (primary) hypertension"
    codeSystem="1.3.6.1.4.1.12559.11.10.1.3.1.44.2" codeSystemName="ICD-10" xsi:type="CD">
    <originalText>
```

to:

```
<code displayName="Essential (primary) hypertension" nullFlavor="NI">
   <translation code="I10"
                displayName="Essential (primary) hypertension"
                codeSystem="1.3.6.1.4.1.12559.11.10.1.3.1.44.2"
                codeSystemName="ICD-10"
                xsi:type="CD">
```

because there is no equivalent map for "I10" in ICD_10 in the target SNOMED CT value set.

### 6.4.11 *mapValueSetAndMove* — find a code in a relative path, map it and put it at the target

The *mapValueSetAndMove* function gets the code in a relative path identified by the *source* attribute, transforms values, and moves the revised to another part in the document.

```
<transform>
    <path>/manufacturedMaterial/code</path>
    <transformation name="replaceCode">
        <arg>
            <fromcode>
                <code nullFlavor="NA" xmlns="urn:hl7-org:v3"/>
            </fromcode>
        </arg>
    </transformation>
    <transformation name="mapValueSetAndMove" map="ATC_RxNorm_epSOSActiveIngredient_VS">
        <arg>
            <source>/manufacturedMaterial/epsos:ingredient[@classCode="ACTI"]/epsos:ingredient[@classCode="MMAT"]/epsos:code</source
        </arg>
    </transformation>
</transform>
```

**Figure 20: *MapValueSetAndMove* applies transrmation to the map codes and moves to a different part of the document.**

As an example of using this function consider the case below, where the *NA* code is removed (designated with the orange rectangle) in the figure below and is replaced by the translation of the epSOS ingredient code (as designated with the corresponding rectangle in Figure 22.

```
<consumable>
    <manufacturedProduct>
        <templateId root="2.16.840.1.113883.10.20.1.53"/>
        <manufacturedMaterial>
            <code nullFlavor="NA"/>
            <epsos:formCode code="10219000" codeSystem="1.3.6.1.4.1.12559.11.10.1.3.1.44.1" displayName="Tablet">
                <epsos:originalText>
                    <epsos:reference value="#form.med.1"/>
                </epsos:originalText>
            </epsos:formCode>
            <epsos:ingredient classCode="ACTI">
                <epsos:quantity>
                    <epsos:numerator value="100" unit="mg" xsi:type="epsos:PQ"/>
                    <epsos:denominator value="1" unit="1" xsi:type="epsos:PQ"/>
                </epsos:quantity>
                <epsos:ingredient classCode="MMAT" determinerCode="KIND">
                    <epsos:code code="C07AB02" displayName="metoprolol" codeSystem="2.16.840.1.113883.6.73" codeSystemName="Anatomical Therapeutic Chemical">
                        <epsos:originalText>
                            <epsos:reference value="#cod.med.1"/>
                        </epsos:originalText>
                    </epsos:code>
```

**Figure 21: Part of an EU patient summary in which the *mapValueSteAndMove* function is applied to *manufacturedMaterial*.**

```
<consumable>
   <manufacturedProduct>
      <templateId root="2.16.840.1.113883.10.20.22.4.23"/>
      <manufacturedMaterial>
         <code codeSystem="2.16.840.1.113883.6.88"
               codeSystemName="RxNorm"
               code="6918"
               displayName="Metoprolol">
            <translation code="C07AB02"
                         displayName="metoprolol"
                         codeSystem="2.16.840.1.113883.6.73"
                         codeSystemName="Anatomical Therapeutic Chemical">
               <epsos:originalText>
                  <epsos:reference value="#cod.med.1"/>
               </epsos:originalText>
            </translation>
         </code>
         <epsos:formCode code="10219000"
                         codeSystem="1.3.6.1.4.1.12559.11.10.1.3.1.44.1"
                         displayName="Tablet">
            <epsos:originalText>
               <epsos:reference value="#form.med.1"/>
            </epsos:originalText>
         </epsos:formCode>
         <epsos:ingredient classCode="ACTI">
            <epsos:quantity>
               <epsos:numerator value="100" unit="mg" xsi:type="epsos:PQ"/>
               <epsos:denominator value="1" unit="1" xsi:type="epsos:PQ"/>
            </epsos:quantity>
            <epsos:ingredient classCode="MMAT" determinerCode="KIND">
               <epsos:code code="C07AB02"
                           displayName="metoprolol"
                           codeSystem="2.16.840.1.113883.6.73"
                           codeSystemName="Anatomical Therapeutic Chemical">
                  <epsos:originalText>
```

**Figure 22: Result of applying the *MapValueSetAndMove* to the element *ManufacturedMaterial* in the previous figure.**

# 7 User Guide

The reference implementation of the transformer has been developed by Harold Solbrig and Kevin Peterson at Mayo and is available as open sorce at https://github.com/trillium-bridge/trillium-bridge-transformer

## 7.1 Overview

The Trillium Bridge support action extends the European Patient Summaries and Meaningful Use II, Transitions of Care in the United States to establish an interoperability bridge that will benefit EU and US citizens alike, advancing eHealth innovation and contributing to the triple win: quality care, sustainability and economic growth -- http://www.trilliumbridge.eu. The most updated version of the user guide will be at the TrilliumBridge GitHub.

The Trillium Bridge Transformer is a Java API, Command Line Interface, and Web-based Application to translate between epSOS Patient Summary documents and HL7 C-CDA Continuity of Care Documents (CCD).

The following sections provide a guidance on the usage of this package including the pre-requisites and the build/installation instructions.

## 7.2 Project Setup

1. Install Java SDK 7+
2. * Install Git
3. * Install Maven
4. * Clone the repository using a Git Clone https://github.com/kevinpeterson/trillium-bridge-transformer.git

(*) Not necessary unless bulding/compiling. See downloads for binary distributions.

## 7.3 Build/Compile

From the trillium-bridge-transformer directory, run mvn clean install

This will produce several artifacts, including a ZIP/TAR.GZ file will all necessary components. This artifacts will be located in the directory:

```
trillium-bridge-transformer/trillium-bridge-transformer-cli/target/dist
```

and are named:

```
trillium-bridge-transformer-cli-{version}-bin.zip
```

and

```
trillium-bridge-transformer-cli-{version}-bin.tar.gz
```

## 7.4 Download/Install

Download the latest ZIP or TAR.GZ binary distribution. For installation, extract the archive to the desired location on the filesystem.

## 7.5 Distribution Package

The `trillium-bridge-transformer-cli-{version}-bin.{suffix}` package will contain the following structure (see footnotes below for further information on the various parts § 7.5.1 Components):

```
├── bin
│   ├── ccda2epsos              <- (1)
│   ├── ccda2epsos.bat          <- (2)
│   ├── epsos2ccda              <- (3)
```

```
│   ├── epsos2ccda.bat          <- (4)
│   ├── tbt-webapp              <- (5)
│   └── tbt-webapp.bat          <- (6)
├── conf
│   ├── cts2fileservice         <- (7)
│   │   └── map
│   ├── nooptransform           <- (8)
│   ├── outputformats           <- (9)
│   │   ├── CDA.xsl
│   │   └── outputformats.json
│   ├── schema                  <- (10)
│   │   ├── CDA_R2_NormativeWebEdition2010  <- (11)
│   │   └── TBXform.xsd         <- (12)
│   ├── tbxform
│   │   ├── FP7-SA610756-D3.1.xml    <- (13)
│   │   └── ValueSetMaps.xml         <- (14)
│   ├── translation             <- (15)
│   │   ├── it-to-en.xml
│   │   └── en-to-es.xml
│   └── xslt
│       ├── CCD.xsl             <- (16)
│       ├── CCD-IT.xsl          <- (17)
│       ├── CTS2Access.xsl      <- (18)
│       ├── TBTransformations.xsl <- (19)
│       ├── TBXform.xsl         <- (20)
│       ├── ccda2epsos_options.json   <- (21)
│       ├── epsos2ccda.xsl            <- (22)
│       ├── epsos2ccda_options.json   <- (23)
│       ├── noop.xsl            <- (24)
│       └── xslt.properties     <- (25)
├── doc
│   └── README.txt
├── lib
│   └── *.jar <- Java jar dependencies
└── webapp
    ├── logs
    │   ├── error.log           <- (26)
    │   └── output.log          <- (27)
    └── trillium-bridge-transformer-webapp-{version}.war <- (28)
```

### 7.5.1 Components

#### 7.5.1.1 *Launch Scripts*

**(1)** `ccda2epsos` - the Unix launch file for the CCDA to epSOS transformation
**(2)** `ccda2epsos.bat` - the Windows launch file for the CCDA to epSOS transformation
**(3)** `ccda2epsos` - the Unix launch file for the epSOS to CCDA transformation
**(4)** `epsos2ccda.bat` - the Windows launch file for the epSOS to CCDA transformation

All above commands allow the following parameters:

```
(ccda2epsos | epsos2ccda) inputFile
 -f (-format, --format) [xml | html | pdf]  : The output format
 -h (-help, --help)                         : Print this message
 -v (-version, --version)                   : Print the application version
```

```
example: ccda2epsos -f html ../my/ccda.xml
```

**(5)** `tbt-webapp` - the Unix launch file for the web applcation
**(6)** `tbt-webapp.bat` - the Windows launch file for the web applcation

```
Usage: tbt-webapp(.bat) [--help|--version] [ server opts] [[ context opts] context ...]
Server opts:
 --version                          - display version and exit
 --log file                         - request log filename (with optional 'yyyy_mm_dd'
wildcard
 --out file                         - info/warn/debug log filename (with optional
'yyyy_mm_dd' wildcard
 --host name|ip                     - interface to listen on (default is all
interfaces)
 --port n                           - port to listen on (default 8080)
 --stop-port n                      - port to listen for stop command
 --stop-key n                       - security string for stop command (required if --
stop-port is present)
 [--jar file]*n                     - each tuple specifies an extra jar to be added to
the classloader
 [--lib dir]*n                      - each tuple specifies an extra directory of jars
to be added to the classloader
 [--classes dir]*n                  - each tuple specifies an extra directory of
classes to be added to the classloader
 --stats [unsecure|realm.properties] - enable stats gathering servlet context
 [--config file]*n                  - each tuple specifies the name of a jetty xml
config file to apply (in the order defined)
Context opts:
 [[--path /path] context]*n         - WAR file, web app dir or context xml file,
optionally with a context path

example: tbt-webapp --port 5150
```

By default, web app will bind to port 8080 and be available at http://localhost:8080/

#### 7.5.1.2 Local CTS2 Service

**(7)** `cts2fileservice` - the root directory for the local file-based CTS2 service. This is the place where you can put maps that aren't available externally as well as resources that need to be cached locally. The directory structure matches the CTS2 REST service interface.

#### 7.5.1.3 *Transformations*

**(8)** `nooptransform` - the implementaion of the no-op direct copy CCDA <-> epSOS transform. This will be deprecated and replaced with a live transform.
**(9)** `outputformats` - specification of output format XSLT transformations. See here for more inforamation on output format configuration

#### 7.5.1.4 *Schema*

**(10)** `schema` - directory that carries XML Schemas
**(11)** `CDA_R2_NormativeWebEdition2010` - CDA schema directory for resolving CDA document headers
**(12)** `TBXform.xsd` - the XML Schema that defines the transformation rules, value set maps and language transformation tables

#### 7.5.1.5 *Transformation Tables*

**(13)** `FP7-SA610756-D3.1.xml` Transformation rules. This is the primary table that controls the transform. The structure is defined by `TBXform.xsd`
**(14)** `ValueSetMaps.xml` Value Set Mapping table. This controls which file(s) or service(s) are used to resolve code and value maps. The structure is defined by `TBXform.xsd`
**(15)** `translations` - Translation tables. File names are in the form "{from-language}-to-{to-language}.xml". The structure is defined by `TBXform.xsd`

#### 7.5.1.6 *XSLT Files specification the main CCDA/epSOS XSLT transformations.*

**(16)** `CCD.xsl` - Default client-side html formatter for CCD data
**(17)** `CCD_IT.xsl` - Italian language html formatter for CCD data (example)
**(18)** `CTS2Access.xsl` - XSLT templates and functions for doing CTS2 based code and value transforms
**(19)** `TBTransformations.xsl` - XSLT function library for transformations
**(20)** `TBXform.xsl` - The main transformation engine that traverses an epSOS or CCD document and applies the rules in the transformation rules table
**(21)** `ccda2epsos_options.json` - Description of screen options (parameters) for the ccda to epsos transformation
**(22)** `epsos2ccda.xsl` -
**(23)** `epsos2ccda_options.json` - Description of screen options (parameters) for the epsos to ccda transformation
**(24)** `noop.xsl` -
**(25)** `xslt.properties` - Transformation configuration file. See here for more information on CCDA/epSOS XSLT configuration.

#### 7.5.1.7 *Web Application*

**(26)** `error.log` - the standard error log of the web application.
**(27)** `output.log` - the standard output log of the web application.
**(28)** `trillium-bridge-transformer-webapp-{version}.war` - the web application archive. This can be then deployed to an application server such as Tomcat, JBoss, etc.

## 7.6 Transformations phases

There are two different transformation phases. The first phase transforms CCDA XML to epSOS XML (or vice versa). The next phase (optional) takes that resulting transformed XML and is so required converts it to a desired output format (such as HTML).

### 7.6.1 Configuring the CCDA <-> epSOS Transformation

The `conf/xslt/xslt.properties` file is the configuration file used to configure the XSLTs used to execute the transformation, and had the following format:

```
xslt.epsos2ccda=TBXform.xsl
xslt.ccda2epsos=TBXform.xsl
```

This file should contain two entries as show -- one for each type of transformation. The value of the `xslt.epsos2ccda` and `xslt.ccda2epsos` properties should be the relative path to the XSLT used for conversion.

By default, the command line applications and the web application will introspect this file and utilize the specified XSLTs.

### 7.6.2 Configuring the Output Format Transformation

The `conf/outputformats/outputformats.json` file is the configuration file used to configure available output formats and the XSLTs used to implement them, and has the following format:

```
[
{
    "name": "CDA XSLT", // the name of the transformation
    "xslt": "CDA.xsl",  // the relative path to the XSLT
    "output": "HTML",   // the type of output (only HTML currently)
    "useFor": "BOTH"    // whether the tranform applies to 'CCDA', 'EPSOS', or 'BOTH'
},
{
//... more transforms
}
]
```

By default, the command line applications and the web application will introspect this file and utilize the specified XSLTs.

## 7.7 Web Application Deployment

The Trillium Bridge Transformer comes with a built-in Jetty server, which can be started from the `bin`directory.

Alternatively, the web application can be deployed to an existing web container. To do this, first ensure the Trillium Bridge Transformer HOME enviroment variable (`TBT_HOME`) is set. This will allow the web application to find the user-specified XSLT configuration files. `TBT_HOME` should be set to the root directory of the installation packge:

```
TBT_HOME
    ├── bin
    ├── conf
    ├── doc
    ├── lib
    └── webapp
```

Once `TBT_HOME` has been set, deploy the WAR file located in the webapp directory to the target web container.

## 7.8 Java API

To use the Java API directly, first add the Maven repository to your pom.xml file:

```
...
<repository>
  <id>informatics-releases</id>
  <url>http://informatics.mayo.edu/maven/content/repositories/releases</url>
</repository>
...
```

Then, add the Maven dependency:

```
<dependency>
    <artifactId>trillium-bridge-transformer-core</artifactId>
    <groupId>edu.mayo</groupId>
    <version>--version-here--</version>
</dependency>
```

To get started using the API, instantiate the transfomer:

```
TrilliumBridgeTransformer transformer = new XsltTrilliumBridgeTransformer();
```

The interface for the transformer is as follows:

```
/**
 * Transformation interface for converting XML files to and from CCDA and epSOS format.
 */
public interface TrilliumBridgeTransformer {

    /**
     * Valid output formats
     */
    public enum Format {XML, HTML, PDF}

    /**
     * Convert a CCDA XML document into epSOS format.
     *
     * @param ccdaStream the CCDA document
     * @param epsosStream the output stream
     * @param outputFormat the output format
     */
```

```
    public void ccdaToEpsos(InputStream ccdaStream, OutputStream epsosStream, Format
outputFormat);

    /**
     * Convert an epSOS XML document into CCDA format
     *
     * @param epsosStream the epSOS document
     * @param ccdaStream the output stream
     * @param outputFormat the output format
     */
    public void epsosToCcda(InputStream epsosStream, OutputStream ccdaStream, Format
outputFormat);

}
```

## 7.9 Testing

From the `trillium-bridge-transformer` directory, run `mvn clean test`.

## 7.10 Contributing changes

- Fork the repository
- Send a pull request

## 7.11 License

All artifacts are licensed under the Apache License.

.

# 8 Demonstration of the Transformer Web Application

## 8.1 Web page interface

The current graphical user interface of the Transformer appears in the figure below.

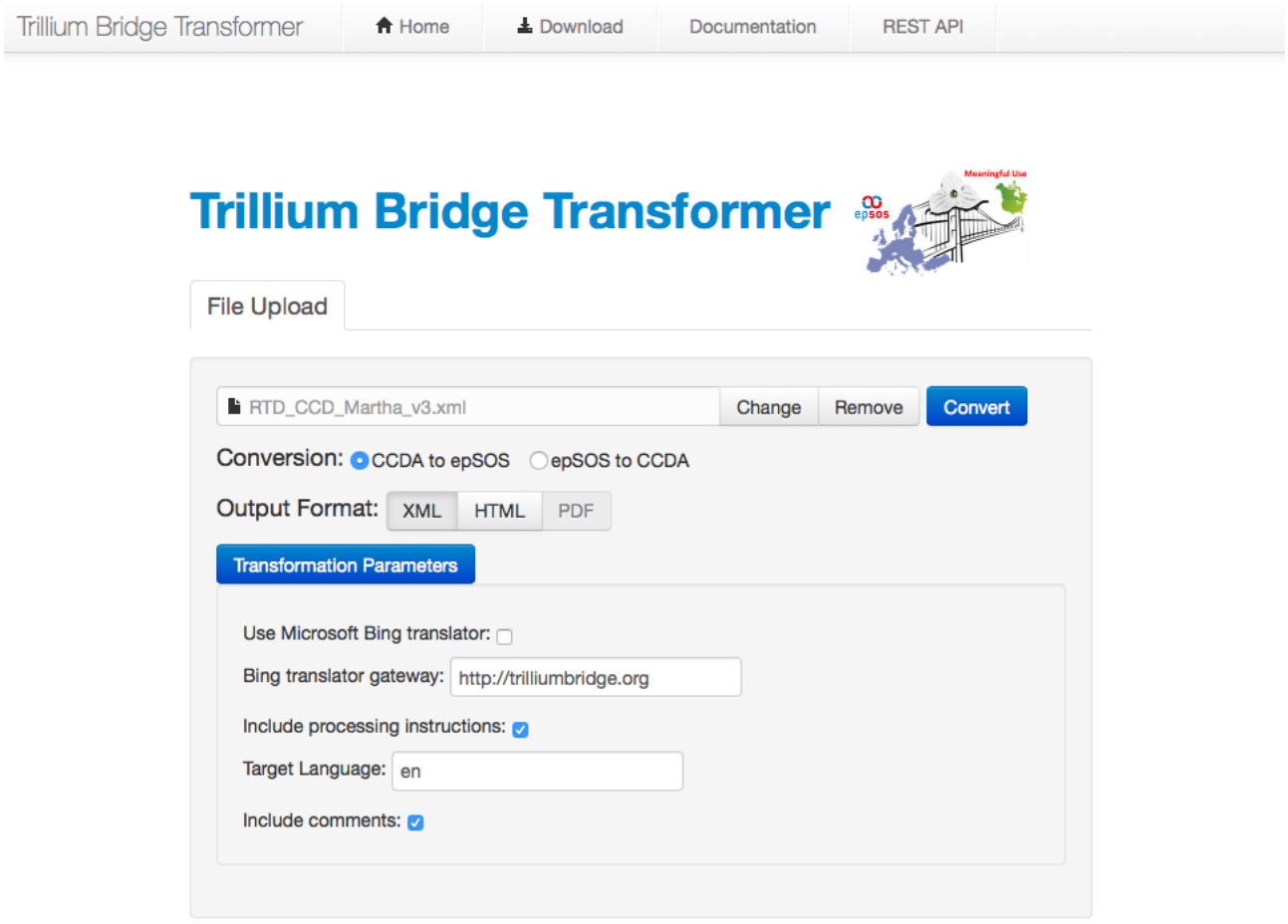
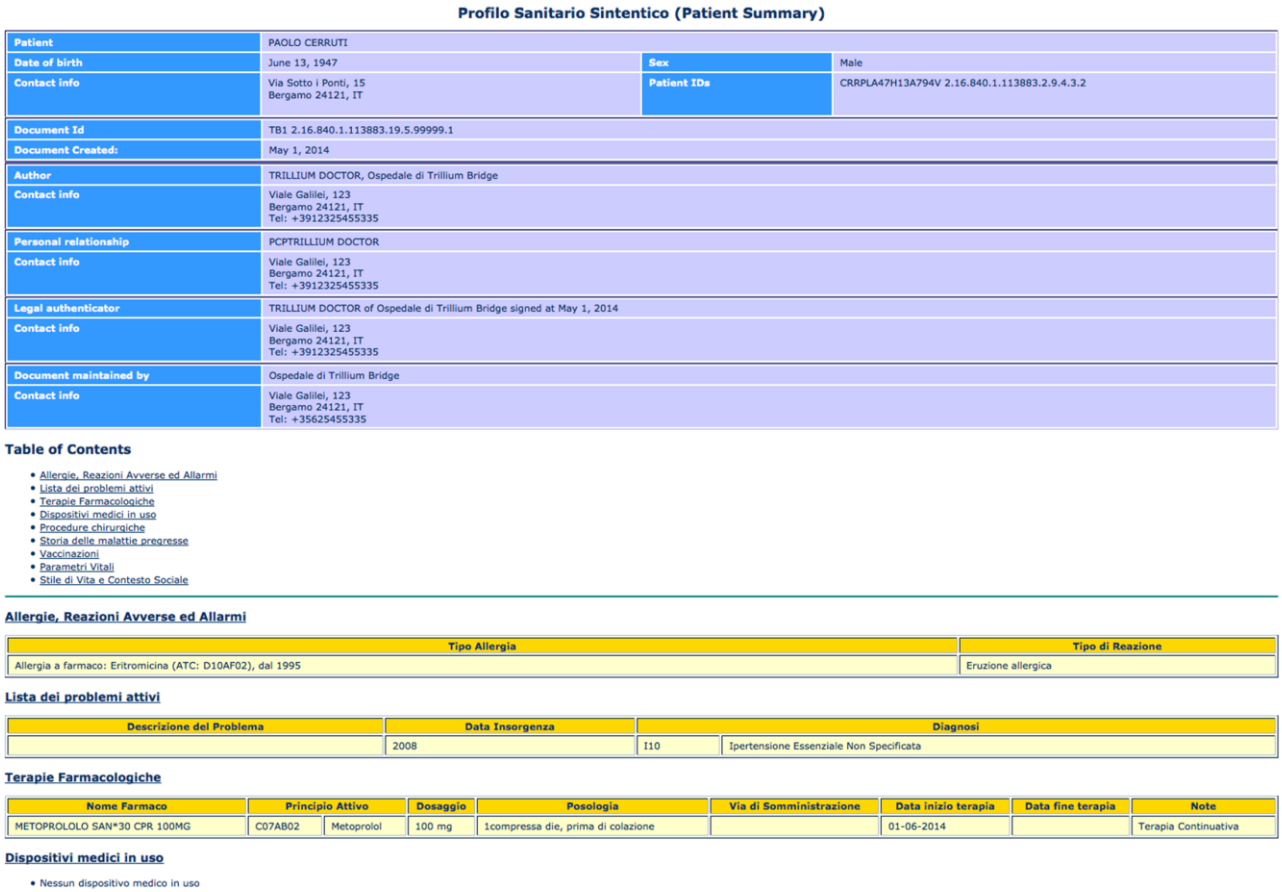



Figure 23: Transformer Engine web application interface



**Profilo Sanitario Sintentico (Patient Summary)**

| | | | |
|---|---|---|---|
| **Patient** | PAOLO CERRUTI | | |
| **Date of birth** | June 13, 1947 | **Sex** | Male |
| **Contact info** | Via Sotto i Ponti, 15<br>Bergamo 24121, IT | **Patient IDs** | CRRPLA47H13A794V 2.16.840.1.113883.2.9.4.3.2 |
| **Document Id** | TB1 2.16.840.1.113883.19.5.99999.1 | | |
| **Document Created:** | May 1, 2014 | | |
| **Author** | TRILLIUM DOCTOR, Ospedale di Trillium Bridge | | |
| **Contact info** | Viale Galilei, 123<br>Bergamo 24121, IT<br>Tel: +3912325455335 | | |
| **Personal relationship** | PCPTRILLIUM DOCTOR | | |
| **Contact info** | Viale Galilei, 123<br>Bergamo 24121, IT<br>Tel: +3912325455335 | | |
| **Legal authenticator** | TRILLIUM DOCTOR of Ospedale di Trillium Bridge signed at May 1, 2014 | | |
| **Contact info** | Viale Galilei, 123<br>Bergamo 24121, IT<br>Tel: +3912325455335 | | |
| **Document maintained by** | Ospedale di Trillium Bridge | | |
| **Contact info** | Viale Galilei, 123<br>Bergamo 24121, IT<br>Tel: +35625455335 | | |

**Table of Contents**

- Allergie, Reazioni Avverse ed Allarmi
- Lista dei problemi attivi
- Terapie Farmacologiche
- Dispositivi medici in uso
- Procedure chirurgiche
- Storia delle malattie pregresse
- Vaccinazioni
- Parametri Vitali
- Stile di Vita e Contesto Sociale

**Allergie, Reazioni Avverse ed Allarmi**

| Tipo Allergia | Tipo di Reazione |
|---|---|
| Allergia a farmaco: Eritromicina (ATC: D10AF02), dal 1995 | Eruzione allergica |

**Lista dei problemi attivi**

| Descrizione del Problema | Data Insorgenza | Diagnosi | |
|---|---|---|---|
| | 2008 | I10 | Ipertensione Essenziale Non Specificata |

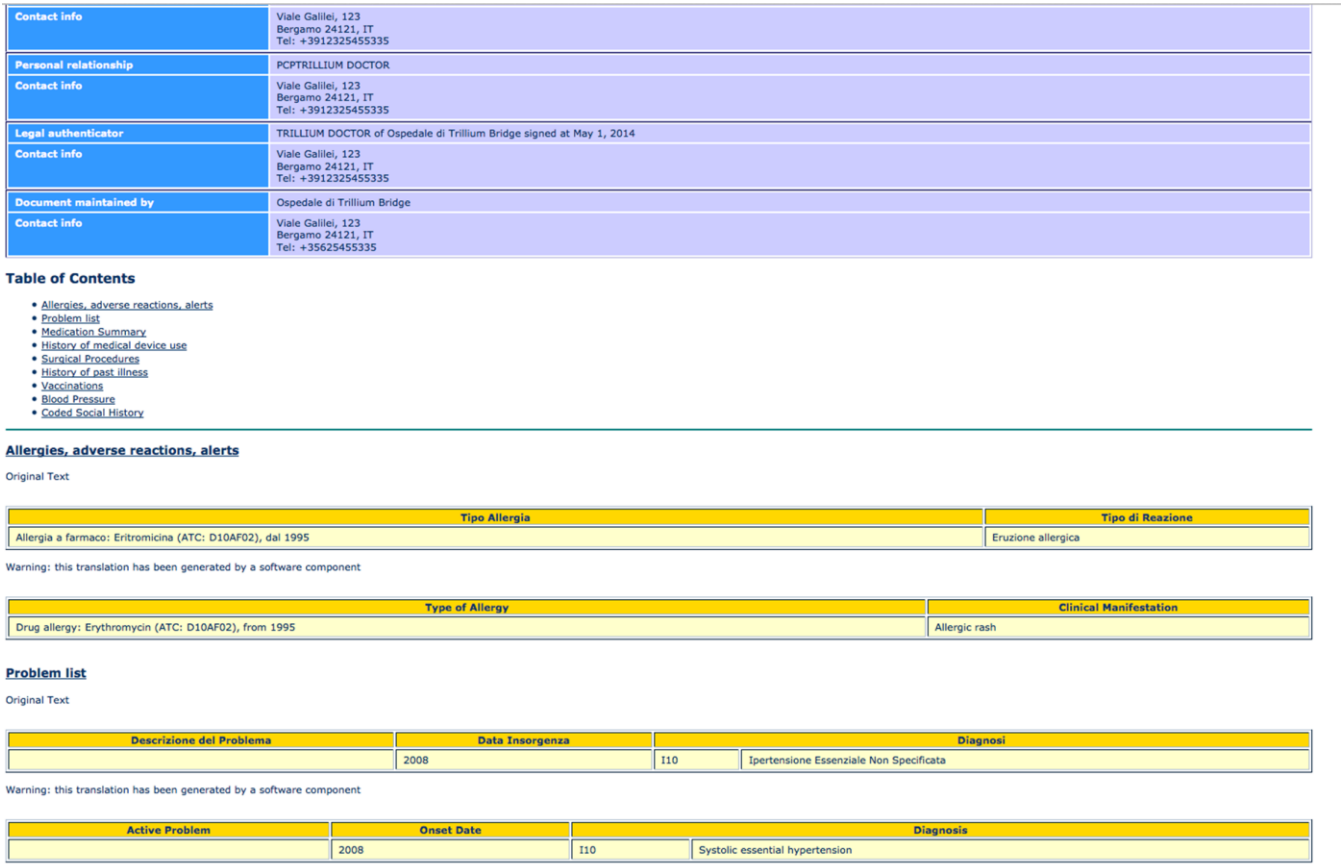
**Terapie Farmacologiche**

| Nome Farmaco | Principio Attivo | | Dosaggio | Posologia | Via di Somministrazione | Data inizio terapia | Data fine terapia | Note |
|---|---|---|---|---|---|---|---|---|
| METOPROLOLO SAN*30 CPR 100MG | C07AB02 | Metoprolol | 100 mg | 1compressa die, prima di colazione | | 01-06-2014 | | Terapia Continuativa |

**Dispositivi medici in uso**

- Nessun dispositivo medico in uso

**Figure 24: Paolo's Italian patient summary is used as input to the transformer.**



| | |
|---|---|
| **Contact info** | Viale Galilei, 123<br>Bergamo 24121, IT<br>Tel: +3912325455335 |
| **Personal relationship** | PCPTRILLIUM DOCTOR |
| **Contact info** | Viale Galilei, 123<br>Bergamo 24121, IT<br>Tel: +3912325455335 |
| **Legal authenticator** | TRILLIUM DOCTOR of Ospedale di Trillium Bridge signed at May 1, 2014 |
| **Contact info** | Viale Galilei, 123<br>Bergamo 24121, IT<br>Tel: +3912325455335 |
| **Document maintained by** | Ospedale di Trillium Bridge |
| **Contact info** | Viale Galilei, 123<br>Bergamo 24121, IT<br>Tel: +35625455335 |

**Table of Contents**

- Allergies, adverse reactions, alerts
- Problem list
- Medication Summary
- History of medical device use
- Surgical Procedures
- History of past illness
- Vaccinations
- Blood Pressure
- Coded Social History

**Allergies, adverse reactions, alerts**

Original Text

| Tipo Allergia | Tipo di Reazione |
|---|---|
| Allergia a farmaco: Eritromicina (ATC: D10AF02), dal 1995 | Eruzione allergica |

Warning: this translation has been generated by a software component

| Type of Allergy | Clinical Manifestation |
|---|---|
| Drug allergy: Erythromycin (ATC: D10AF02), from 1995 | Allergic rash |

**Problem list**

Original Text

| Descrizione del Problema | Data Insorgenza | Diagnosi | |
|---|---|---|---|
| | 2008 | I10 | Ipertensione Essenziale Non Specificata |

Warning: this translation has been generated by a software component

| Active Problem | Onset Date | Diagnosis | |
|---|---|---|---|
| | 2008 | I10 | Systolic essential hypertension |

**Figure 25: Transformation of Paolo's patient summary with the Bing Automatic Translation on, using BING for use in the US.**

| | |
|---|---|
| **Contact info** | Viale Galilei, 123<br>Bergamo 24121, IT<br>Tel: +3912325455335 |
| **Legal authenticator** | TRILLIUM DOCTOR of Ospedale di Trillium Bridge signed at May 1, 2014 |
| **Contact info** | Viale Galilei, 123<br>Bergamo 24121, IT<br>Tel: +3912325455335 |
| **Document maintained by** | Ospedale di Trillium Bridge |
| **Contact info** | Viale Galilei, 123<br>Bergamo 24121, IT<br>Tel: +35625455335 |

**Table of Contents**

- Allergies, adverse reactions, alerts
- Problem list
- Medication Summary
- History of medical device use
- Surgical Procedures
- History of past illness
- Vaccinations
- Blood Pressure
- Coded Social History

**Allergies, adverse reactions, alerts**

Original Text

| Tipo Allergia | Tipo di Reazione |
|---|---|
| Allergia a farmaco: Eritromicina (ATC: D10AF02), dal 1995 | Eruzione allergica |

Warning: this translation has been generated by a software component

| Type of Allergy | Clinical Manifestation |
|---|---|
| Drug allergy: Erythromycin (ATC: D10AF02), dal 1995 | Allergic rash |

**Problem list**

Original Text

| Descrizione del Problema | Data Insorgenza | Diagnosi | |
|---|---|---|---|
| | 2008 | I10 | Ipertensione Essenziale Non Specificata |

Warning: this translation has been generated by a software component

| Active Problem | Onset Date | Diagnosi | |
|---|---|---|---|
| | 2008 | I10 | Systolic essential hypertension |

Figure 26: Transformed Paolo's Italian patient summary without automatic translation for use in the US.

**Health Summary**

| | | | |
|---|---|---|---|
| **Patient** | Martha Token | | |
| **Date of birth** | May 16, 1971 | **Sex** | Female |
| **Contact info** | Primary Home:<br>321 Prince St.<br>San Diego, CA 92111, US<br>Tel: (619)555-1212 | **Patient IDs** | 551-12-1234 2.16.840.1.113883.4.1 |
| **Document Id** | TT988.1 2.16.840.1.113883.19.5.99999.1 | | |
| **Document Created:** | September 15, 2013, 00:00 -0400 | | |
| **Care provision** | Check-up from September 8, 2013, 10:15 to September 8, 2013, 10:45 | | |
| **Performer (primary care provider)** | Dr. Henry Oncologist of Atrius Health | | |
| **Author** | Henry Oncologist | | |
| **Contact info** | 1002 Healthcare Drive<br>Portland, OR 99123, US<br>Tel: 555-555-1002 | | |
| **Personal relationship** | PCP | | |
| **Contact info** | | | |
| **Legal authenticator** | Henry Oncologist of signed at February 27, 2013, 13:00:00 +0500 | | |
| **Contact info** | 1002 Healthcare Drive<br>Portland, OR 99123, US<br>Tel: 555-555-1002 | | |
| **Document maintained by** | Atrius Health | | |
| **Contact info** | Work Place:<br>1002 Healthcare Drive<br>Portland, OR 99123, US<br>Tel: 555-555-1002 | | |

**Table of Contents**

- Allergies, Adverse Reactions, Alerts
- Medications
- Problems
- Results
- Plan of Care

**Allergies, Adverse Reactions, Alerts**

| Type | Substance | Overall Severity | Reaction | Reaction Severity | Status |
|---|---|---|---|---|---|
| Drug allergy | ALLERGENIC EXTRACT, PENICILLIN | Moderate to Severe | Nausea | Moderate to Severe | Active |

**Medications**

| Medication | Directions | Start Date | Status |
|---|---|---|---|
| Anastrozole | 1 mg once daily | 20130103 | Active |

**Problems**

1. Breast Cancer Stage II Status: Active

Figure 27: Transformed Martha's patient summary for use in the EU.

# 9 Limitations and Future Extensions

The functionality of the transformer software developed as part of the Trillium Bridge project is limited. Its intent is to serve only as a proof of concept. The Trillium Bridge team hopes that the open source community will adopt and extend its functionality.

In this spirit, several limitations have been identified and listed below:

(1) The transformer has ony been tested with a limited set of patient summary samples.

(2) The transformation engine currently depends on the order of the *templateIds* in the patient summary. As an example, consider the following order of *templateIds*

```
<entry typeCode="DRIV">
        <act classCode="ACT" moodCode="EVN">
          <templateId root="2.16.840.1.113883.10.20.1.27"/>
          <templateId root="1.3.6.1.4.1.19376.1.5.3.1.4.5.1"/>
          <templateId root="1.3.6.1.4.1.19376.1.5.3.1.4.5.2"/>
```

This is not the same as:

```
<entry typeCode="DRIV">
        <act classCode="ACT" moodCode="EVN">
          <templateId root="1.3.6.1.4.1.19376.1.5.3.1.4.5.1"/>
          <templateId root="2.16.840.1.113883.10.20.1.27"/>
          <templateId root="1.3.6.1.4.1.19376.1.5.3.1.4.5.2"/>
```

This issue can be readily corrected.

(3) The transformation code has not been optimized for performance. There is a number of improvements could increase its speed and reliability. Moreover, the code has not been tested in a high load environment and the authors make no guarantee about thread safety, memory leaks, etc.

(4) The HTML tables are transformed literally and that may not readily correspond to the optimal structure for the transformed patient summary document.

(5) The transformer software code has been tested against a limited number of patient examples and has not been validated against all of the transformations specified in *D3.1*. It is likely that additional transformations will be required as more documents have been tested against the transformer functions. A possible action to fix it, besides testing additional samples is to go through *D3.1* and determine whether all transformations are supported.

(6) The only title and text field translations that will eventually be available are Italian, Portuguese, Spanish to English (and visa-versa).

(7) Title and text field translations are based on content rather than template identifier. We may want to change this in a subsequent release and create translation bound to template indentifiers.

(8) The transformation engine currently uses a free Bing Translator token that was acquired for the Trillium Bridge project. This is currently limited to 4 million characters of translation per month. We need to investigate alternative translation mechanisms and more scalable approaches for subsequent refind version of the transformer, but that goes beyond the scope of this project.

## 9.1 Future extensions

Extending the transformation rules requires expertise in XML, XPATH and, to a lesser degree, XSLT to maintain. The transformation rules have been designed in such a way that it should be possible to build a GUI on the front end that would allow viewing and authoring without an intimate understanding of the underlying technology.

# 10 Summary

This document serves as an introduction and guide to the transformer code available at GitHub https://github.com/trillium-bridge/trillium-bridge-transformer.

It is the hope of the Trillium Bridge team that developers around the world will use it with acknowledgement to the team that designed and developed it, extend it and make it useful for citizens of the world that would like to have the patient summary available and fit for the purpose of use in the transatlantic setting.