Quick Guide to CAM Dictionaries

Building and using canonical XML components dictionaries for CAM

Author:
David RR Webber
Chair OASIS CAM TC
April, 2010

http://www.oasis-open.org/committees/cam
Agenda

• Why Dictionaries?
  • Making interoperability baked into the XML infrastructure
  • Linguistic and semantic alignment; neutral representations
  • International and national initiatives

• Dictionary concepts
  • What does the dictionary contain conceptually?
  • Understanding the dictionary XML / use with spreadsheets
  • Mapping across domains

• Generating Dictionaries
  • Harvesting from existing sources | Using modelling tools | Manually

• Catalog of existing dictionaries
  • Available dictionaries and domains

• Using Dictionaries / Tools
Why Dictionaries?

Making interoperability baked into the XML infrastructure
Linguistic and semantic alignment; neutral representations
International and National initiatives
Baking in Interoperability

• Using consistent component definitions dramatically improves interoperability and reuse
• Having formal design methods makes development faster, easier, predictable and repeatable
• Aligning local practice to industry domain dictionary can reduce complexity and reinforce best practices
• Dictionary definitions can be automatically evaluated for common mistakes and this reduces the opportunity for errors during design phase
• Generating software artifacts from neutral dictionary definitions ensures reliable information exchange results across user communities and their particular systems, platforms and tools
Neutral Content Model Representation

- Neutral representations allow business stakeholders to participate in dictionary development without technology barriers
- Concise neutral formats can be viewed as simple spreadsheets as they have no special syntax dependencies
- Based on open public standard specifications, semantic concepts and leading knowledge domain techniques
- Neutral representation prevents lock-in by vendor, syntax, tooling or platforms
- Maximizes flexibility and future proofing of dictionary definitions
Linguistic and Semantic Alignment

- Formal community domain naming and design rules provide consistency of definitions
- Consistency of definitions minimizes duplication and overlapping of dictionary components
- Dictionaries allow collaboration on component development to improve the overall results
- Formal component content detail drives alignment
- Design best practices ensure logical self-contained components that can be selected contextually
- Avoids explosion of complexity and excessive over definition (e.g. “kitchen-sink” schema)
International and National Initiatives

- Sampling of prior dictionary related work
  - ISO 11179
  - UN/CEFACT CCTS / ebXML
  - OASIS Registry / Repository specifications

- USA national initiatives
  - NIEM

- UK / EU examples
  - e-GIF
  - RIDE (Health interoperability) and iSURF interchanges

- Industry work
  - HL7 and NIST – Healthcare term registry
  - Geospatial use of ebXML registry
Introduction to dictionary concepts

What does the dictionary contain conceptually?
How does the XML organize the information?
How does this relate to Excel spreadsheets?
Mapping to dictionary and between domains
Follows Naming and Design Rule (NDR) principles and guidelines

Canonical Components Dictionary XML

Each compound component

Parent (ABIE) Item

Parent (ABIE) Item

Parent (ABIE) Item

Parent (ABIE) Item

Optional attributes of component

Child (BBIE) Item

Child (BBIE) Item

Child (BBIE) Item

Child (BBIE) Item

Attribute (ASBIE)

Attribute (ASBIE)

Attribute (ASBIE)

Attribute (ASBIE)

Each atomic component

ebXML CCTS terms (ABIE, BBIE, ASBIE)

Parent = Aggregate Business Information Entity
Child = Basic Business Information Entity
Attribute = Association Business Information Entity

* CCTS – Core Components Technical Specification
Example – Person Name

• Person Name (ABIE)
  • Verified Details? (ASBIE)
• First Name (BBIE)
• Middle Name (BBIE)
• Last Name (BBIE)
  • Previous Name? (ASBIE)
• Language Code (BBIE)

Language Code may exist independently of Person Name
Verified Details and Previous Name are flags that denote additional
information about the entity they are associated with

There are three component items aspects:
structure relationships; content rules; definitions
Conceptual Information View

**DOMAIN DATA COMPONENTS**

- **Structure**
- **Rules**
- **Definitions**

**CAM toolkit processing**

**CAM toolkit processing**

**Dictionary Components**

- **Items**
  - **Item (ABIE, BBIE, ASBIE)**
  - **Properties**
    - **Name**
    - **Unique ID**
    - **Component Type**
    - **Cardinality**
    - **Content Type**
    - **Content Mask**
    - **Children**
    - **Group**
    - **Structure Context**
    - **Where from**
    - **Definition**
    - **Rules**
    - **Language, Label, Notes**

* Required items in Blue
XML View of Dictionary Content

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ITEMSlookup xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  Dictionary generator, version 1.32
  Writing output to - file:C:/temp/NIEM/Trade-dictionary.xml
  Merging mode - true
  For namespace - *
  Source template - Trade
  CCTS - false
  Rules filter - false
  Generated on - 2010-04-11T13:17:55.509-04:00
</ITEMSlookup>
```

**Items**

- **Name**
- **Unique ID**
- **Component Type**
- **Type**
- **Cardinality**
- **Parent/Child linkage**
- **Where referenced**
- **Items**
- **Content Type**
- **Content Mask**

* See slide notes for explanation
### Excel Spreadsheet View

The spreadsheet view is a tabular representation of data with rows and columns. Each row represents an item, and the columns include:

- **Type (ABIE, BBIE)**: The type of the item.
- **children**: Children elements related to the item.
- **properties as columns**: Properties of the item are listed as columns.
- **An item per row**: Each row represents a distinct item.

#### Example Row

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>children</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>it-AgentType</td>
<td>it-d3e950</td>
<td>ABIE</td>
<td>repeatable, optional</td>
<td>children</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>it-AgentIdentification</td>
<td>it-d3e966</td>
<td>BBIE</td>
<td>repeatable, optional</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>it-AgentQualityAssuranceIndicator</td>
<td>it-d3e983</td>
<td>BBIE</td>
<td>repeatable, optional</td>
<td>boolean, true/false</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>it-AgentStatusCodeText</td>
<td>it-d3e1002</td>
<td>BBIE</td>
<td>repeatable, optional</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>it-AgentStatusText</td>
<td>it-d3e1020</td>
<td>BBIE</td>
<td>repeatable, optional</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>
Mapping to Dictionaries

- You can compare a template of components to a dictionary
  - check within a domain for alignment to dictionary
  - check between domains for interoperability
  - merge new/existing components with dictionary
- Matches on physical names
- Reports matching items and details
- Reports statistics and percentages of matching
- Generates crosswalk xml file
- Compatible with Microsoft Excel
- Report can be used to do spell checking
Example cross-reference spreadsheet

Formatted view in Microsoft Excel of import of cross-reference report details (from generated XML file)

Matched details; item and alignment, definition

Microsoft Excel - example.xls

<table>
<thead>
<tr>
<th>Source</th>
<th>matched</th>
<th>type</th>
<th>annotated</th>
<th>annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. ResourceDatabase| string | | | |
2. Body| string | | | |
3. Description| string | | | |
4. Extension| string | | | |
5. LastModifiedDate| string | | | |
6. ID| string | | | |
7. ConversationId| string | | | |
8. TelecommunicationService| string | | | |
9. Service| string | | | |
10. IPAddress| string | | | |
11. Name| string | | | |
12. Address| string | | | |
13. AlternateAddress| string | | | |
14. Identifier| string | | | |
15. ContactPerson| string | | | |
16. ContactOrganization| string | | | |
17. ContactRole| string | | | |
18. ContactMethod| string | | | |
19. ContactType| string | | | |
20. ContactTelecommunicationService| string | | | |
21. ContactTelecommunication| string | | | |
22. ContactTelecommunicationService| string | | | |
23. ContactTelecommunication| string | | | |
24. ContactLocation| string | | | |
25. ContactPerson| string | | | |
26. ContactOrganization| string | | | |
27. ContactRole| string | | | |
28. ContactMethod| string | | | |
29. ContactType| string | | | |
30. ContactTelecommunicationService| string | | | |
31. ContactTelecommunication| string | | | |
32. ContactTelecommunicationService| string | | | |
33. ContactTelecommunication| string | | | |
34. ContactLocation| string | | | |
35. ContactPerson| string | | | |
36. ContactOrganization| string | | | |
37. ContactRole| string | | | |
38. ContactMethod| string | | | |
39. ContactType| string | | | |
40. ContactTelecommunicationService| string | | | |
41. ContactTelecommunication| string | | | |
42. ContactTelecommunicationService| string | | | |
43. ContactTelecommunication| string | | | |
44. ContactLocation| string | | | |
45. ContactPerson| string | | | |
46. ContactOrganization| string | | | |
47. ContactRole| string | | | |
48. ContactMethod| string | | | |
49. ContactType| string | | | |
50. ContactTelecommunicationService| string | | | |
51. ContactTelecommunication| string | | | |
52. ContactTelecommunicationService| string | | | |
53. ContactTelecommunication| string | | | |
54. ContactLocation| string | | | |
55. ContactPerson| string | | | |
56. ContactOrganization| string | | | |
57. ContactRole| string | | | |
58. ContactMethod| string | | | |
59. ContactType| string | | | |
60. ContactTelecommunicationService| string | | | |
61. ContactTelecommunication| string | | | |
62. ContactTelecommunicationService| string | | | |
63. ContactTelecommunication| string | | | |
64. ContactLocation| string | | | |
65. ContactPerson| string | | | |
66. ContactOrganization| string | | | |
67. ContactRole| string | | | |
68. ContactMethod| string | | | |
69. ContactType| string | | | |
70. ContactTelecommunicationService| string | | | |
71. ContactTelecommunication| string | | | |
72. ContactTelecommunicationService| string | | | |
73. ContactTelecommunication| string | | | |
74. ContactLocation| string | | | |
75. ContactPerson| string | | | |
76. ContactOrganization| string | | | |
77. ContactRole| string | | | |
78. ContactMethod| string | | | |
79. ContactType| string | | | |
80. ContactTelecommunicationService| string | | | |
81. ContactTelecommunication| string | | | |
82. ContactTelecommunicationService| string | | | |
83. ContactTelecommunication| string | | | |
84. ContactLocation| string | | | |
85. ContactPerson| string | | | |
86. ContactOrganization| string | | | |
87. ContactRole| string | | | |
88. ContactMethod| string | | | |
89. ContactType| string | | | |
90. ContactTelecommunicationService| string | | | |
91. ContactTelecommunication| string | | | |
Generating Dictionaries

Process Overview
Harvesting from existing sources
Using modelling tools
Manually from scratch
Specific step by step guides
(see slides at end for specific technique screen shots)
Overview of Dictionary Generation

1. **Source schema in XSD syntax**
   - **Import**

2. **OASIS CAM template**
   - **Apply**

3. **NDR Evaluation, Refactor, Renaming Tools**
   - **Build**

4. **Generate Standard Components Dictionary XML**

   - **Canonical XML Components Dictionary**

   - **XML**

---

Harvesting from Existing Sources

• Schema of exchange structures
  • Process each exchange schema – merge into dictionary
  • And / or process the schema of core components that the exchange schema all import / include

• Physical database models
  • Importing XSD of database schema
  • Run renaming tool to apply NDR on SQL names to make XML named components
  • Generate dictionary
Using Modelling Tools

• **Existing logical data models**
  • Export to Schema XSD
  • Import into new CAM template
  • Run refactor tool to remove model artifacts (optional)
  • Run renamer tool to make consistent XML named components (NDR compliant)

• **Creating new data models**
  • Build model from scratch in modelling tool
  • Process as above
Dictionary Building Processes

Option 1 – From Enterprise Data Model
Import XSD and refactor for use with OASIS CAM

Export Components in XSD syntax
Collection of objects from model

Model Components XSD schema

OASIS CAM template

NDR Evaluation, Refactor, Renaming Tool

Generate Standard Components Dictionary XML

XML

Dictionary of exchange components

Analyst Review

Legend

Automated
Manual

Option 2 – Derive from existing exchange XSD schema
Import each XSD and merge into CAM dictionary

Exchange XSD schemas

Model Components XSD schema

OASIS CAM template

NDR Evaluation, Refactor, Renaming Tool

Merge & Generate Dictionary XML

XML

Dictionary of exchange components

ebXML CCTS compatible (ABIE, BBIE, ASBIE)
Step by step from modelling tools

- Export XSD from modelling tool
- Ingest into CAM as new template
- Optionally run refactor to remove modelling artifacts
  - Compress (removes components flagged by refactor)
- Add namespace for domain
- Run Renaming tool
  - View progress in console window
  - When finished re-open CXF file, save to new CAM template
- Use evaluate template tool to detect errors and omissions
- Spell check
- Correct in modelling tool as needed and re-run
- Advanced techniques
  - Editing xslt word-utils for abbreviations / spellings
  - Setting Editor Preferences to run local updated xslt

[see slides at end for specific technique screen shots]
Step by step for harvesting schema

• Ingest each exchange XSD into new CAM templates
  • Optionally run template evaluator, adjust as desired
• Open template of first schema, run generate dictionary
  • use default empty dictionary as seed
  • pick namespace; if multiple namespaces use the ‘*’ option
• For each remaining template, run generate dictionary
  • use dictionary from previous run as input to next
• To create spreadsheet of dictionary – drag and drop XML into Excel
  • Format columns as desired (size and word wrap)
  • If Excel reports error (Excel has 30,000 character limit in cell size)
    – run context compress utility xslt to create temporary dictionary for import to Excel only
Step by step for harvesting components

- Ingest the components import XSD into new CAM template
  - Use “*” option to load all components
  - Optionally use evaluation tool and adjust as desired
- To add namespace for components
  - Add namespace declaration to root node of template
  - Run Renamer tool and insert desired namespace prefix
- From CAM template, run generate dictionary
  - use default empty dictionary as seed
  - pick namespace or use “*” for all
- To create spreadsheet of dictionary – drag and drop XML into Excel
  - Format columns as desired (size and word wrap)
Creating from Scratch

- **Outlining components in XML**
  - Use CAM editor or XML editor to create structure components
    - Add namespaces as needed (optional)
  - In CAM editor add content rules, cardinality and definitions
  - Run evaluation tool
  - Adjust accordingly
  - Run generate dictionary tool

- **Editing spreadsheet**
  - Start with Excel spreadsheet of CAM dictionary
  - Make changes and additions as desired
  - When finished, use Save As / XML Data to write details out in CAM dictionary XML format.
Dictionaries Catalog

Available dictionaries and domains
Dictionaries with CAM editor download

- LEXT 3.1.4 dictionary
- EDXL HAVE dictionary
- EML 6.0 dictionary
- NIEM properties (limited to items catalog only, no details)

Installed to:
\Program Files\jcam.org.uk\CAMeditor~\eclipse\workspace\CAMEditor\dictionary
Dictionary packs from Sourceforge.net

- Downloadable packs of dictionaries
- NIEM domains dictionary pack contains:
  - CBRN
  - Emergency
  - Family
  - Immigration
  - Infrastructure
  - Intelligence
  - Justice
  - Maritime
  - Screening
  - Trade
  - NIEM core
Dictionary Tools

Building Exchanges from Components
Renaming tool
Compare to dictionary cross-referencing
Create wiki pages of components template
Building Exchanges from Components

• **Dictionaries**
  • Provide reference sets of components to be used in exchanges

• **Blueprint**
  • Is the outline of the structure components to be used in an exchange schema
  • Can import components from one or more domain dictionary collections
  • Sketches out the desired information exchange with reuse of existing exchange component structures, plus any local additions / extensions / exclusions

• **Expander** tool reads the blueprint, references the dictionary, and constructs the complete exchange schema
Renaming Tool

- Expands abbreviations (has built-in lists with context)
- Corrects common typos and spelling errors (200+)
- Adjusts names to camel case with no separators (_)
- Retains acronyms
- Adds name suffix according to NIEM representation terms and type of content, e.g. date, number, code, etc
- Optionally refactor rules by their type for interoperability; make nillable optional; relax DB length restrictions
- Can optionally insert namespace prefix
- Can be configured by editing the lists in: word-utils.xsl
- To run modified renaming – then set preferences to local copy of CXF-rename-util.xsl in CAM editor preferences

(see slides at end for specific technique screen shots)
Compare to Dictionary

• Allows crosschecking of exchange structure to existing dictionary definitions
  • NIEM properties dictionary for entire NIEM items
  • Individual NIEM domain dictionaries for fine grained analysis
  • Other dictionaries – EDXL, EML, CIQ, GML
  • Enterprise Data Model (EDM) dictionary

• Produces
  • Statistical reporting of components and % use
  • Cross-reference spreadsheet
  • Special NIEM function – make “wantlist.xml”
“TiddlyWiki” Dictionary

Interactive wiki of exchange components:
- clickable index
- search tool
- orphan analysis
- edit definitions

Generate from View menu “Interactive document” option
Summary, Review, Resources

Round up of CAM dictionaries
Review

• Dictionary concepts overview
  • Canonical XML components
  • Definitions for components

• Generate Dictionaries
  • Various sources for harvesting components
  • Tools to enhance dictionary content

• Automated exchange package generation
  • Create exchange blueprint outline of components
  • Run expander tool and specify dictionaries to use
  • From generated exchange template, generate XSD schema style (default, NIEM, OASIS, custom)

• Additional Dictionary tools
  • Compare / document / download dictionary packs
Using Dictionaries Process Overview

Dictionary Generation
- Import EDM
- Import XSD
- Naming / NDR / spell check
- Generate Dictionary

Model Exchange
- Blueprints
- Dictionary

Expand, Review, Refine
- NDR Evaluation
- Completed Exchange Structures

Test examples and rules
- Validation Engine
- Test Cases

Exchange Content Modelling
- Dictionary

Expand and Evaluate Results
- Completed Exchange Structures

Review / Tailor Exchange Structures

Generate Package Artifacts and Test them

Package Exchange

Completed Exchange artifacts
- XSD schemas
- XML examples
- Rules Report
- Crosswalk
- Subset Schema
- Mapping Spreadsheet
- Rendering Stylesheet

Business Modelling
- Web based
- Desktop
- Excel

Search tools

OASIS CAM Enabled

Project Inception
- Work Group
- Charter

Search tools
OASIS Content Assembly Mechanism (CAM) & Integration Technologies Guide

- WHAT?
  - Provides lexicon of information content
  - Describes structure constructs
  - Arranges groups of information
  - Simple content typing
  - Software tooling interfaces

- HOW? WHY?
  - Provides actual use patterns (templates)
  - Supports context handling and rules
  - Rendering outputs and documentation for verification
  - Enables integration testing / certification
  - Versioning

- WHO?
  - Alignment of meaning and terms
  - Consistent domain definitions
  - Modelling methods and practice
  - Business information content building blocks
  - Versioning

- WHERE?
  - Shared resources of semantic definitions
  - Code lists
  - Dynamic rendering
  - Distributed versioning control
  - Role and access security management

- WHERE?
  - Domain classification systems
  - Ontology and reasoning definitions
  - Associations and linkages
  - Search and drilldown
  - Modelling tools

- WHEN?
  - Alerts
  - Process control
  - Workflow
  - Automated interfacing
  - Business Intelligence
Reference Materials

References and Links
Tips and Techniques
Links and Resources

• **DOWNLOADS -**
  • CAM Toolkit download
    • [https://sourceforge.net/projects/camprocessor](https://sourceforge.net/projects/camprocessor)

• **SUPPORTING MATERIALS -**
  • NIEM Naming and Design Rules (NDR) 1.3

• **RESOURCES –**
  • UN/CEFACT Core Components Technical Specification
  • Additional support slides (following)
Various Tips and Techniques

Instructions and illustrative screen shots
When to use ERwin refactoring?

• ERwin refactoring is needed when used “associations” check box in ERwin export dialog. This inserts empty nodes for associations into structure parent components and these must be pruned out.

• Also needed to rearrange order of items from model (typically alphabetic) and group related components together in logic sequence and / or under substructure.

• Will make all nillable items optional.

• After running refactor, do compress to delete the nodes flagged for removal; save result as new CAM template.

• Must be run prior to using the renamer tool.
Adding namespace to root element

1. Click on template root Structure element
2. Right click mouse for context menu
3. Enter prefix=URI definition
4. Review in Namespaces tab
Running Renaming Tool

-renamer added by default – otherwise edit name as desired

Leave blank or pick namespace to process

NIEM mode (with term representations); or CamelCase (Java style)

Leave as normal

Optionally insert prefix to all items without prefix

Set to false to retain all rules “as is”

Note: Produces renamer report XML file of old / new names. Can be reviewed as spreadsheet.
Create Dictionary

Use empty dictionary; or merge dictionary from prior dictionary create

Use ‘*’ for all, or pick prefix

Leave blank – or enter text to denote source schema

If true generates additional dictionary in CCTS style (slower)

False to insert rules “as is”; otherwise does smart filter on rules