

JCAM and Interoperability Tutorial

- Interoperability
Mechanisms
- Exchange Management
- Leveraging Open Standards
- XSD ingesting
- WSDL facilitation

David RR Webber

Chair OASIS CAM TC

(Content Assembly Mechanism)

E-mail: drwebber@acm.org

<http://wiki.oasis-open.org/cam>

Interoperability; why it is essential!



Overview

- **Goal is tools for better interoperability: clearer, quicker, easier**
- **Provide developers with tools to aid delivery, documenting and testing of solutions beyond XSD schema alone**
- **Provide extensible toolkit that can be customized easily**
- **Automate delivery of components for the publishing formal interoperability certification packages**
- **Leverage XML and open standards approach**

Approach



- **Open Public Standards**
 - W3C XML/XSD and
 - OASIS Content Assembly Mechanism (CAM) XML instance handling rules technology
- **CAM Designed for Interoperable Exchanges**
 - Augments W3C XSD – fills in the gaps
 - Easier WYSIWYG format than XSD syntax
 - Supports use of XSLT for tool development
- **jCAM Eclipse editor environment provides convenient open desktop toolset**
- **Tool components built with XSLT scripts**
- **Available as Open Source on SourceForge**

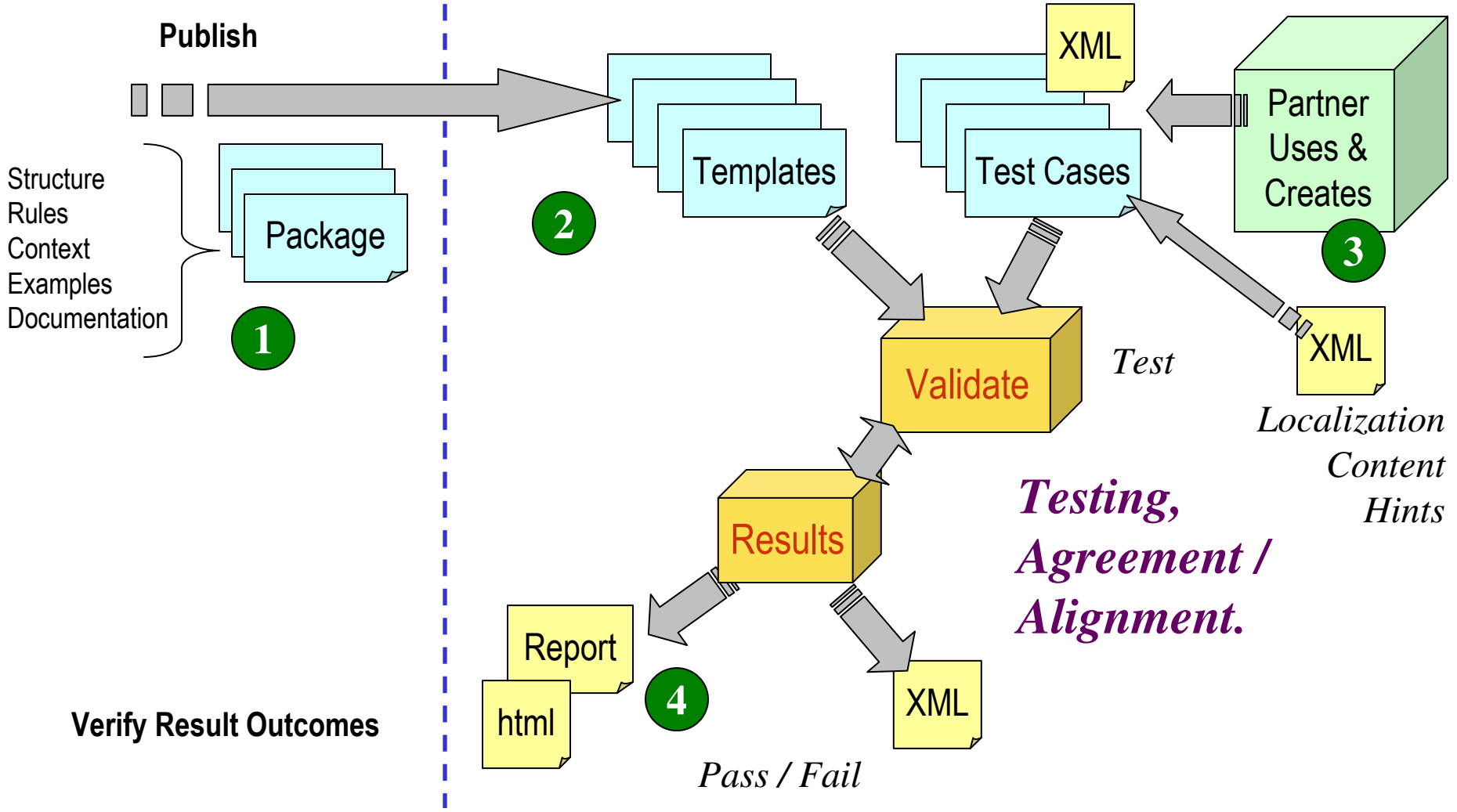
Creating a Package for Interoperability

- **Step 1 - Ingest XSD – extract rules, structure, annotations**
- **Step 2 - Select & mark out your use model in visual editor**
- **Generate and save your “want list” selections**

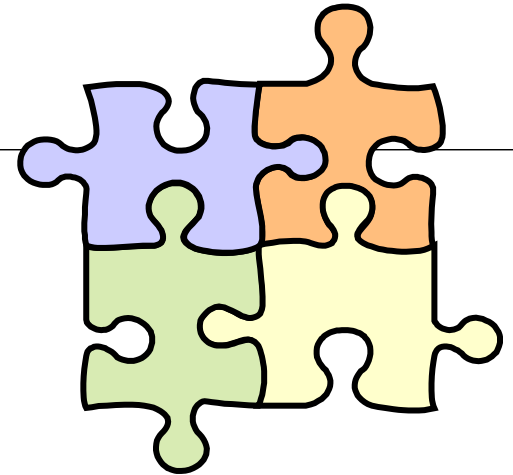
- **Step 3 - Generate your XSD schema subset (WSDL)**
- **Step 4 - Generate rich live test data examples**
 - (complete with content hints / pass / fail / random options)
- **Run rules engine - verify operation and outcomes**

- **Step 5 - Build business analyst documentation of structure elements and rules**
- **Package and Share with exchange partners**

Partner Conformance Testing



Interoperability check-list:



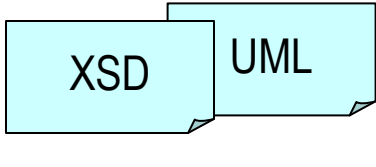
- XSD schema structure model
- CAM template + rules (deterministic)
- Documentation of use patterns (aka “want list” + subset XSD)
- Test cases and examples (pass/fail)
- Content hinting (localization)
- Validation engine for unit and regression testing
- Open standard, open platform and open source allows consistent agreements between participants

IEPD Package Contents

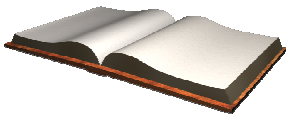
1



Domain Schemas Models



2



Documentation (Word / PDF / OpenDoc Excel / HTML)

Purpose Descriptions Tables Diagrams

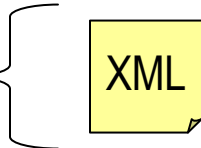


3



Want List in XML XSD subset

Localization to Requirements

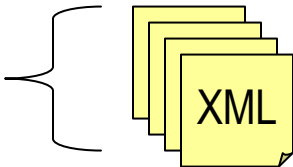


4

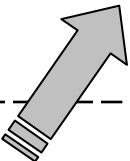


Examples & Test Cases

Generated Conformance Suite



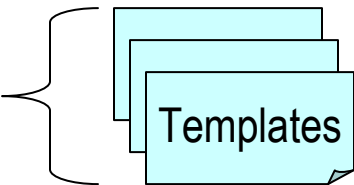
Generate, Manage and Test 1) thru 4) above



5



Structure Rules Context Vocabulary



Challenge: XSD is non-deterministic!

- The schema contains the superset of every exchange component variation
- XSD does not have direct context mechanisms
- Hence people make everything in schema optional
- Dependencies are not clear
- It is difficult to understand the constructs and to document the rules clearly for business users to verify
- It is hard to create test cases and instances (the “want list” tough to visualize)
- Disconnect between XML data types and legacy data – e.g. dates, telephone formats, post codes

Interoperability Mechanisms

**Creating an
Information Exchange Package Documentation
(IEPD) Package**

- **Ingesting XSD schema**
 - step by step example
- **Documenting the Exchange Patterns**
 - Creating “want list” selections
 - Subset XSD generation (for WSDL)
 - Documentation reporting options
- **Testing and Conformance**
 - Creating Test Case examples
 - Content Hinting
 - Running Test Cases
- **Advanced Techniques**

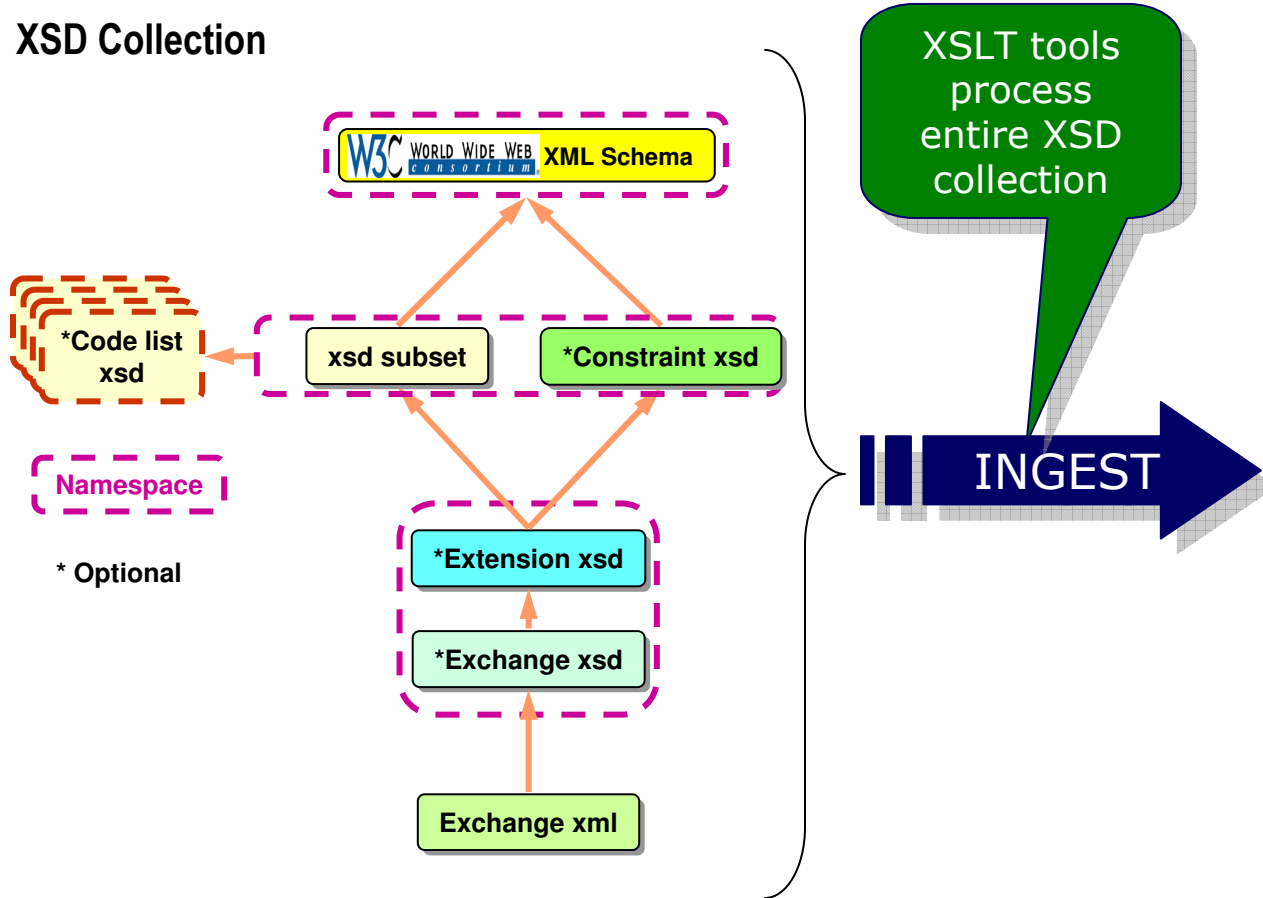
Ingesting XSD Schema

Using jCAM editor Wizard

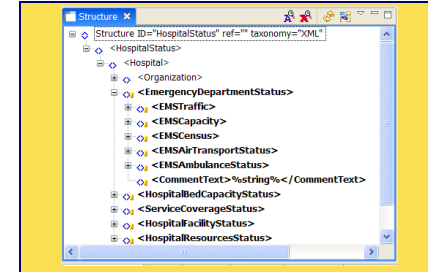
(<http://www.jcam.org.uk>)

Ingesting XSD to CAM template format

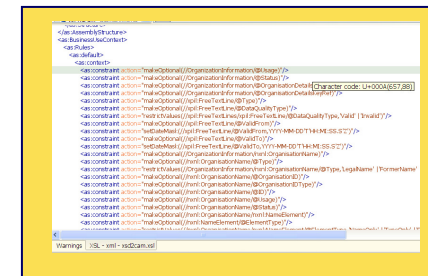
XSD Collection



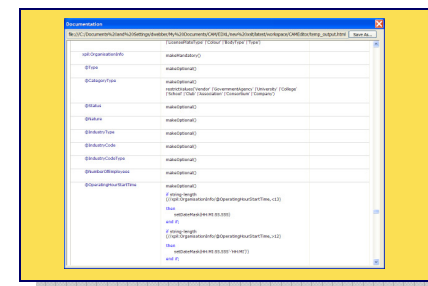
Structure



Rules

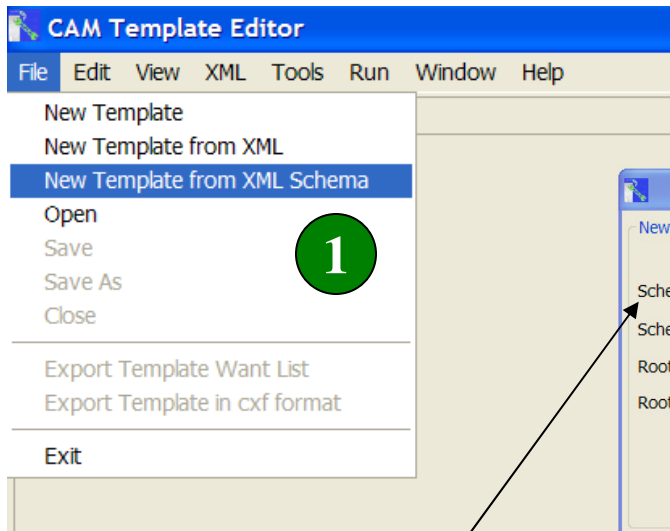


Documentation



CAM = Content Assembly Mechanism

Step 1 & 2 – Pick the XSD schema to ingest



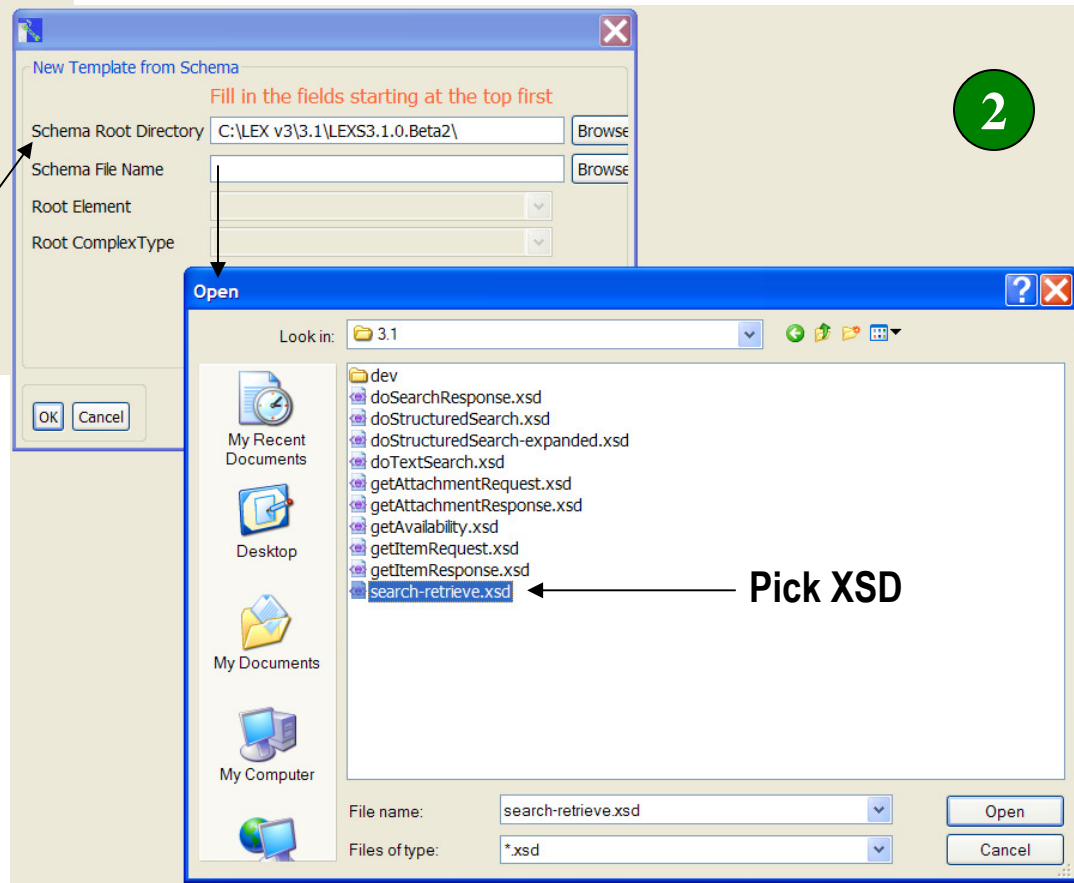
Choose File / New Option

First location is the root folder for the XSD collection.

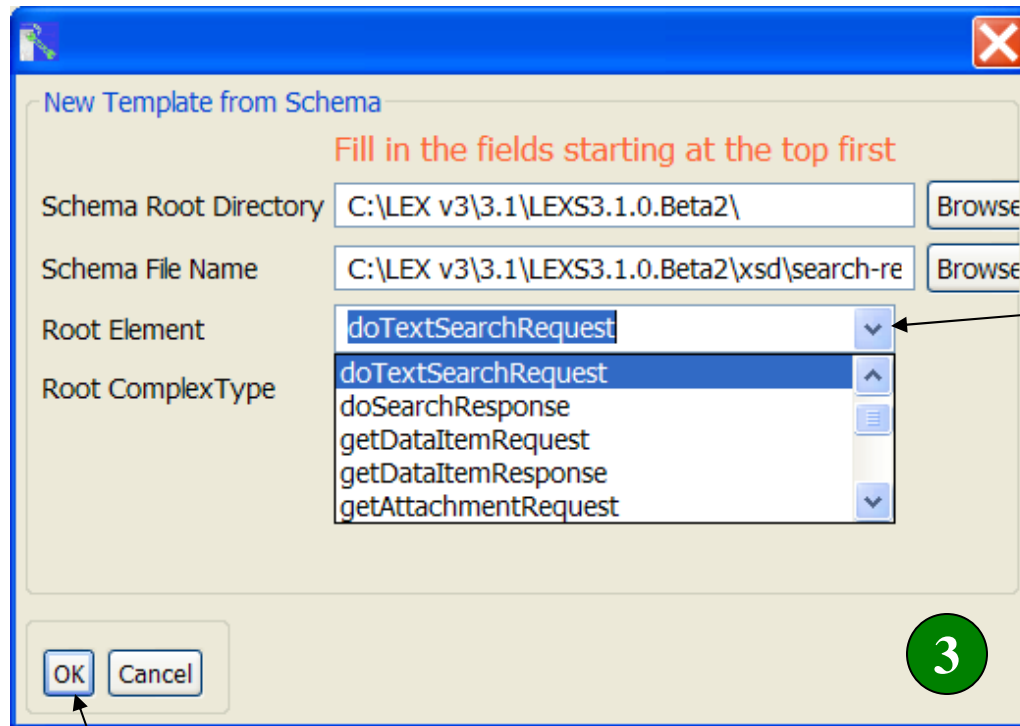
Typically this is the same as the location for the XSD schema you want to ingest.

(Complex XSD can have this in separate folder; hence need for option)

Specify XSD Locations



Step 3 – Choose the XSD parent element



From the dropdown list pick the correct root element you wish to use.

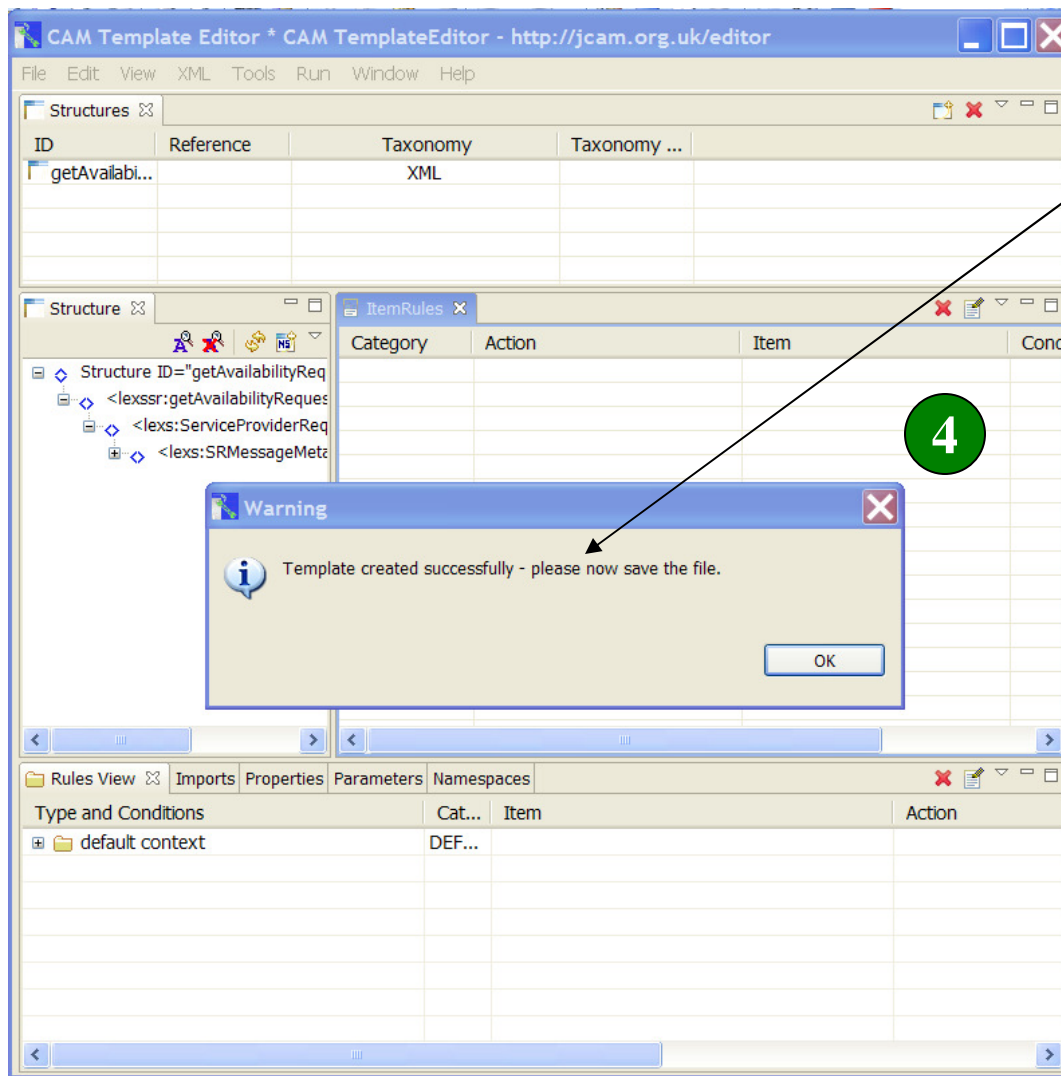
XSD's may have more than one collection in them (as shown here) or may not have the parent node element cleanly defined.

The Wizard shows you the list of all possible ones it finds – so you can select and confirm the right one.

Choose XSD parent element

Confirm and start the XSD ingesting

Step 4 – Ingesting complete – Save Results



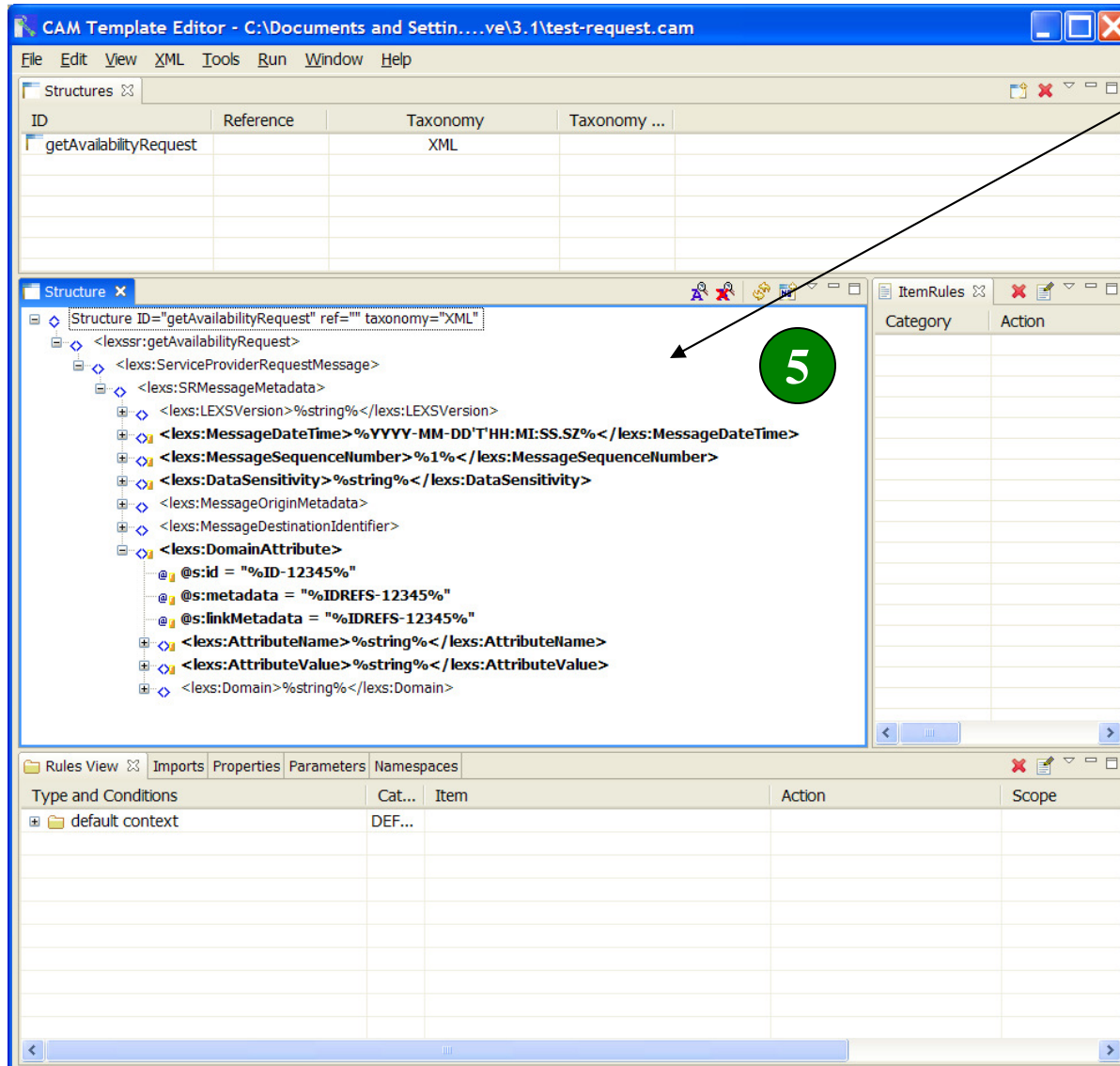
Completed template is loaded and save dialogue appears

Processing usually takes a few seconds.

Complex XSD can take over an hour to process however.

Tip: jCAM runs the ingesting as a background task – so you can continue to use the computer while such long ingesting is proceeding.

Step 5 – Review ingested structure



Completed template is shown in the structure navigator panel for review and editing

Template contains all the default content model and structure rules ingested from the XSD.

All annotations and documentation from XSD also ingested (show as "paperclip" symbol).

Code lists and typical content values inserted for easy visual reference.

Resolving XSD schema import / includes

- Normally the wizard should figure this all out for you
- Complex XSD can have deeply nested “trees” of imported definitions and type “libraries” of other XSDs – that may be elsewhere than the current folder for your particular XSD
- **Tip:** If the ingesting fails – repeat step 1 – but re-specify location for where your XSD collection is to be found
- **Tip:** Use Console to view log messages

Console Log view displays messages

The screenshot shows a software interface with a tree view on the left and a console log on the right. The tree view contains the following items:

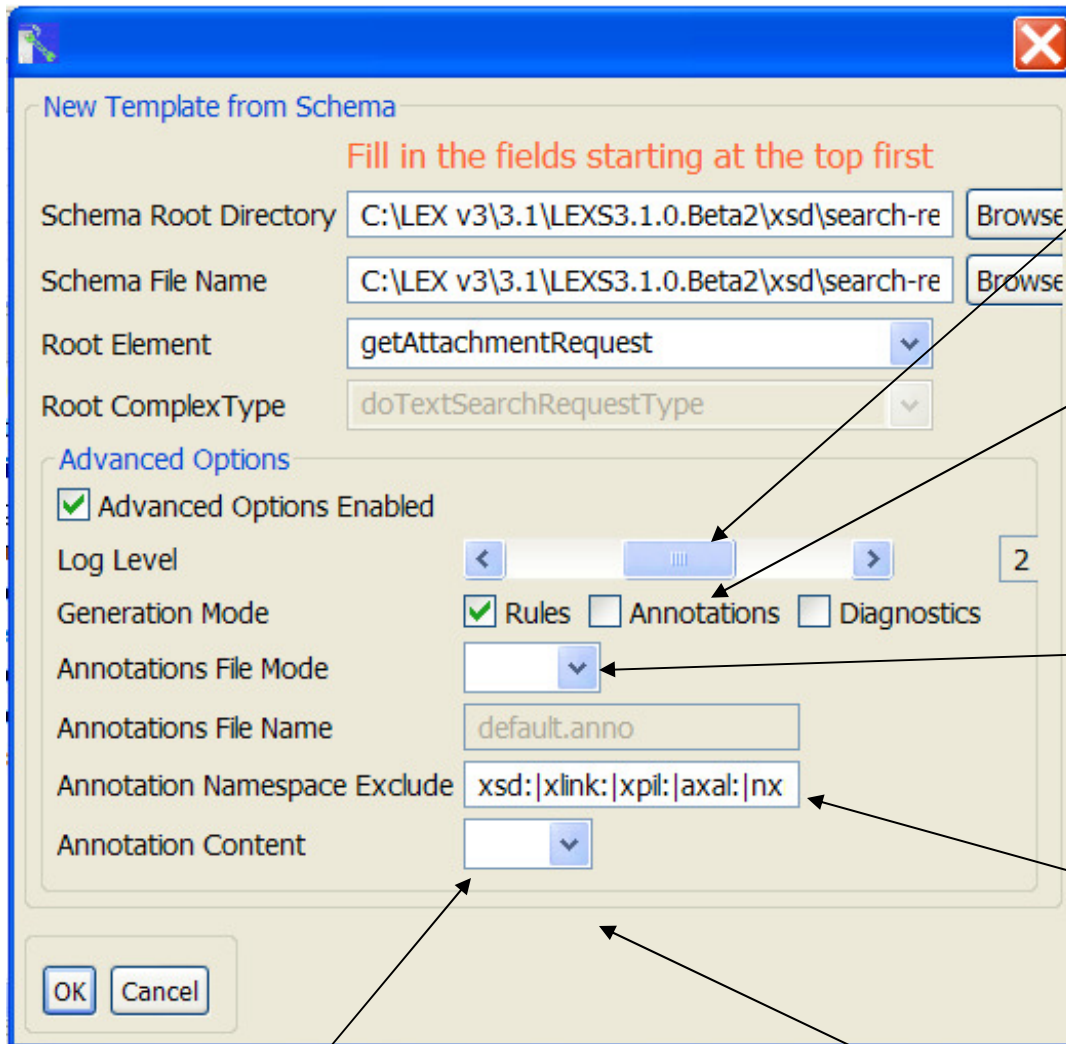
- <ServiceAllowanceCharges>
- <Shipping>
- <StatementOfWork>%string%</StatementOfWork>
- <WageDeterminationDetails>
- <ContractLineItems>
- <CDRI >

The console log is titled "System Output" and contains the following text:

```
ruleChanged:OpenStructure:AwardInstrument
ruleChanged:NewFile
ruleChanged:FileClean
ruleChanged:CloseTemplate
about to transform
xsd2Schema:bundleentry://56/xsl/xsd2cam/xsd2schema.xsl
processing request
transforming output...
[error]non-UTF8 char replaced: [Customer Pickup or Customer's Expense]
[error]non-UTF8 char replaced: [Customer Pickup or Customer's Expense]
[error]non-UTF8 char replaced: [An Employer Identification Number (EIN) issu
[error]non-UTF8 char replaced: [An Employer Identification Number (EIN) issu
[error]non-UTF8 char replaced: [An Employer Identification Number (EIN) issu
[error]non-UTF8 char replaced: [An Employer Identification Number (EIN) issu
[error]non-UTF8 char replaced: [An Employer Identification Number (EIN) issu
transformed
xml
uri:http://www.w3.org/XML/1998/namespace
as
uri:http://www.oasis-open.org/committees/cam
camed
uri:http://jcam.org.uk/editor
xsd
uri:http://www.w3.org/2001/XMLSchema
```

Examining details of log messages to determine if any resolution is needed

Optional Advanced Selections



Internal log message level – 1 is critical messages only, thru 4 which is “all”. Use this for debugging.

Generation Mode – “Rules” is the normal mode; check “Annotations” to ingest notes and comment text as well. Diagnostics is for advanced debugging only.

“inline” is normal mode; use “file” if your annotation results are too big for available memory

Annotation Exclude – this allows selection of only main annotations – not those from imports. Items from matching namespaces are ignored.

“text” is normal mode; use “all” if your annotations have embedded XML tags

“want list” optimization; will exclude items marked to be ignored

Anonymous namespace handling

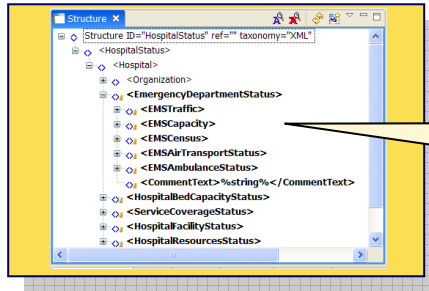
- Some schemas have by default an anonymous namespace declaration in their root `<xsd:schema>` element definition
- This causes a default prefix to be added to any non-qualified name
- If you desire this behavior (most people do not realize why their simple element names end up requiring a prefix) then use the option in the / Tools menu to add the prefix you want
- Typically this is technique is only for schema that may be included into another schema

Documenting the Exchange Patterns

**“Want lists”, documentation
and XSD subset generation**

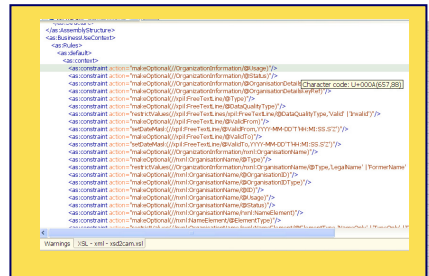
Building a Want List

Structure



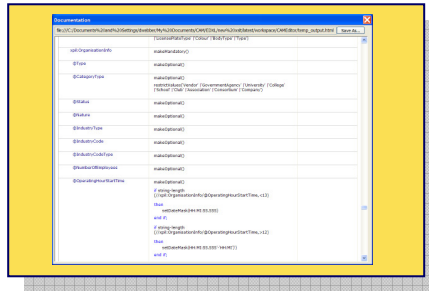
Structure likely extensive!!!

Rules



MARK WHAT IS NOT NEEDED

Documentation



Make Want List



```
<Organization>
  <OrganizationInformation Usage="string" Name="ct:StatusList" OrganisationID="ct:String" OrganisationKeyRef="ct:String" DateValidFrom="2008-04-22T08:37:15.0-04:00" DateValidTo="2008-04-22T08:37:15.0-04:00" LanguageCode="US">
    <xpil:FreeTextLines>
      <!--Excluded: xpil:FreeTextLine--></xpil:FreeTextLines>
    <nxnl:OrganisationName Type="LegalName" OrganisationID="ct:String" OrganisationID="ct:String">
      <nxnl:NameElement ElementType="NameOnly" Abbreviation="false">str
      <!--Excluded: nxnl:SubDivisionName--></nxnl:OrganisationName>
    <xpil:Addresses>
      <xpil:Address Type="Airport">
        <axal:FreeTextAddress>
          <!--Excluded: axal:AddressLine--></axal:FreeTextAddress>
        <axal:Country>
          <!--Excluded: axal:NameElement--></axal:Country>
        <axal:AdministrativeArea Type="City">
          <axal:NameElement>string</axal:NameElement>
          <!--Excluded Group: axal:SubAdministrativeArea--></axal:AdministrativeArea>
          <!--Excluded Group: axal:Locality--><!--Excluded Group: axal:Locality-->
          <axal:Identifier Type="Name" Abbreviation="false">string</axal:Identifier>
          <axal:PostCode>
          <!--Excluded Group: axal:RuralDelivery--><!--Excluded Group: axal:RuralDelivery-->
          <axal:LocationByCoordinates Meridian="string" MeridianCodeType="DatumCodeType" MeridianCodeType="ct:String" Projection="string" ProjectionCodeType="ct:String">

```

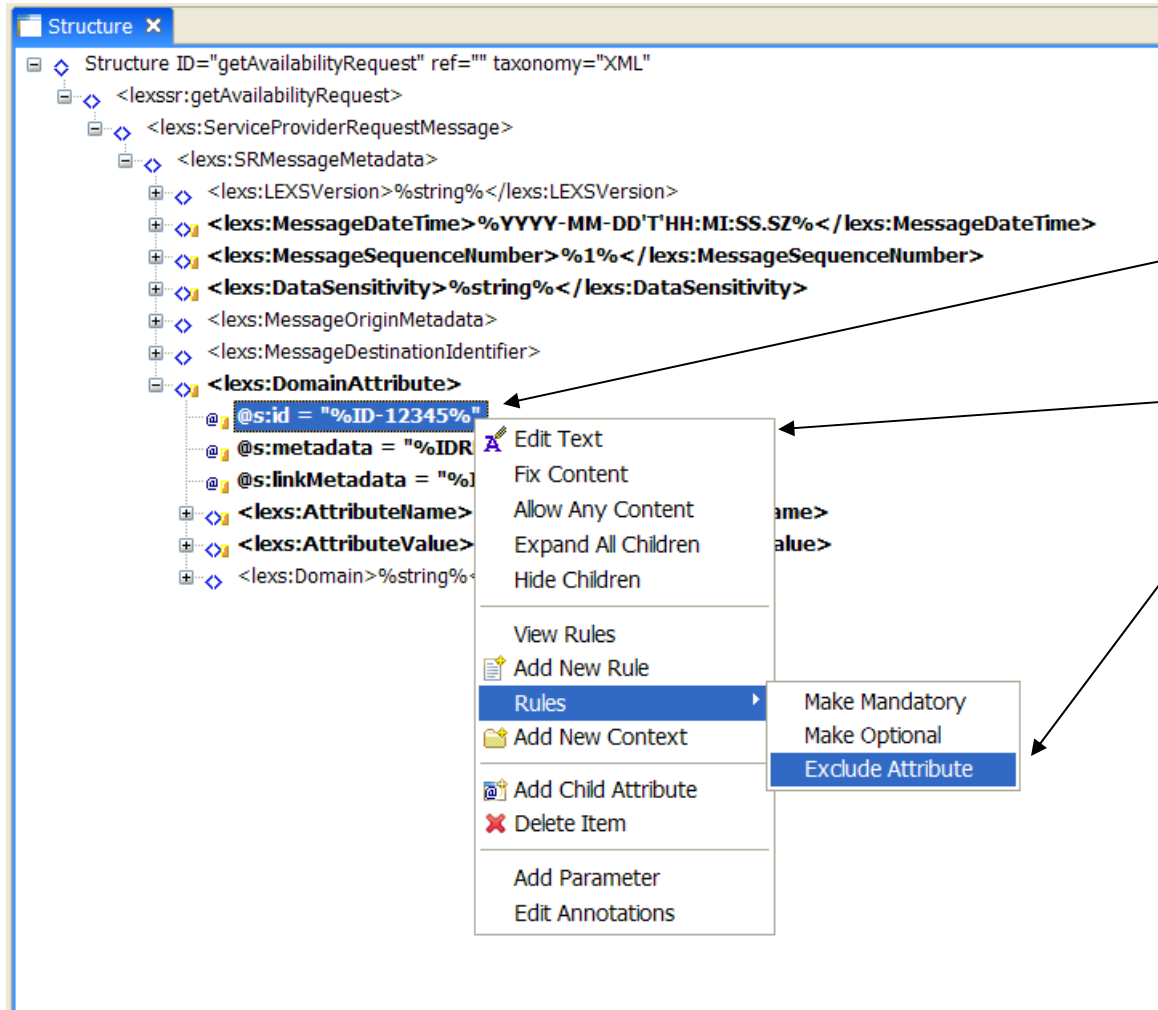
excluded items

DESIRED RESULTS

Marking Items for exclude - want list

- **Can exclude at all levels within the structure**
 - `excludeTree()`
 - `excludeElement()`
 - `excludeAttribute()`
- **Use XPath operators to control scope:**
 - Specific node
 - Group of nodes
 - Anywhere occurs in structure
 - Contextually based on condition
- **Can add new domain elements with own namespace and subset schema**

Using Editor to mark exclude items



Structure Editor Tools

Select focus on item

Invoke action menu
(right mouse click)

Pick action

Tip: exclude rule display
is context sensitive and
only available on
optional items

Tip: use "Add New Rule"
mode to specify
different XPath for
exclude (quick mode
assumes "current path")

Export and Save completed Want List

The screenshot shows the CAM Template Editor interface. The title bar reads "CAM Template Editor * C:\Documents and Settings\...ve\3.1\test-request.cam". The menu bar includes File, Edit, View, XML, Tools, Run, Window, and Help. The File menu is open, showing options: New Template, New Template from XML, New Template from XML Schema, Open, Save, Save As, Close, Export Template Want List (highlighted), Export Template in cxf format, and Exit. Below the menu is a table with columns "Taxonomy" and "Taxonomy ...", containing the text "XML".

The XML structure editor below shows a tree view for "Structure ID='getAvailabilityRequest' ref='' taxonomy='XML'". The tree includes elements like <lexssr:getAvailabilityRequest>, <lexs:ServiceProviderRequestMessage>, <lexs:SRMessageMetadata>, <lexs:LEXSVersion>, <lexs:MessageDateTime>, <lexs:MessageSequenceNumber>, <lexs:DataSensitivity>, <lexs:MessageOriginMetadata>, <lexs:MessageDestinationIdentifier>, <lexs:DomainAttribute> (highlighted in blue), @s:id, @s:metadata, @s:linkMetadata, <lexs:AttributeName>, <lexs:AttributeValue>, and <lexs:Domain>. Some elements have a red dot icon next to them, indicating they are excluded.

File Menu Option

Select Export and specify filename of destination.

Excluded items are designated with red "dot" in structure editor and italics font with no bold highlight

Want List Details

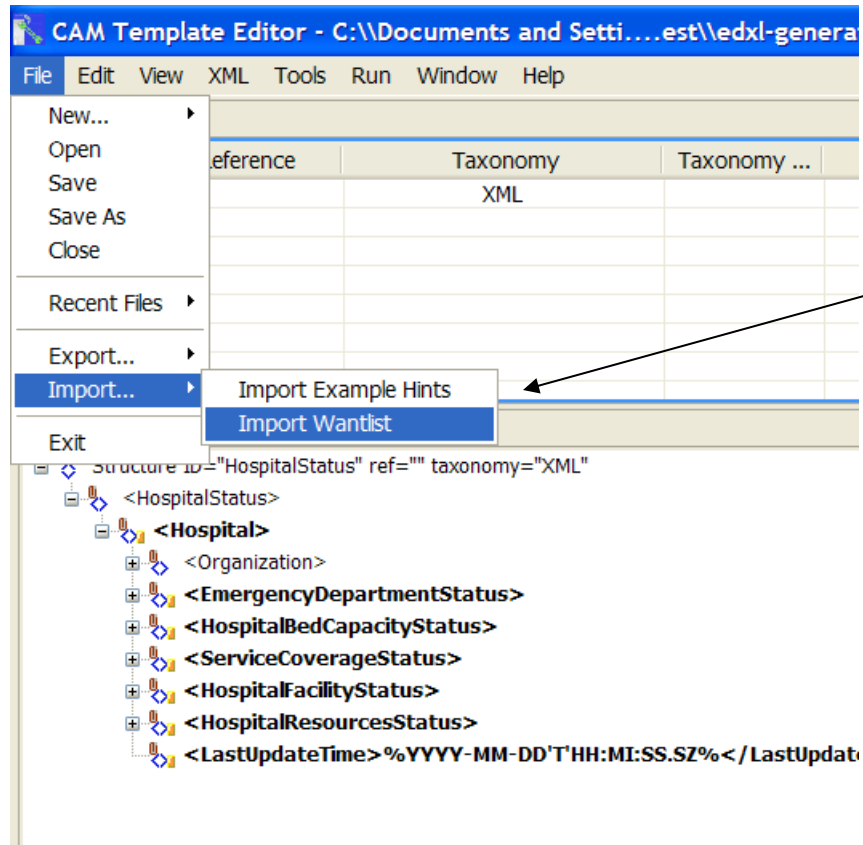
(Exported Example)

```
<?xml version="1.0" encoding="UTF-8" ?>
- <w:WantList xmlns:as="http://www.oasis-open.org/committees/cam" xmlns:camed="http://jcam.org.uk/editor" xmlns:
  xmlns:w="http://niem.gov/niem/wantlist/1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" w:release=
- <Group w:n="\HospitalStatus" w:isXclude="0">
- <Group w:n="HospitalStatus\Hospital" w:isXclude="0">
- <Group w:n="Hospital\Organization" w:isXclude="0">
- <Group w:n="Organization\OrganizationInformation" w:isXclude="0">
  <Attribute w:n="Organization\OrganizationInformation\@Usage" w:isXclude="0" />
  <Attribute w:n="Organization\OrganizationInformation\@Status" w:isXclude="0" />
  <Attribute w:n="Organization\OrganizationInformation\@OrganisationDetailsKey" w:isXclude="0" />
  <Attribute w:n="Organization\OrganizationInformation\@OrganisationDetailsKeyRef" w:isXclude="0" />
  <Attribute w:n="Organization\OrganizationInformation\@xlink:type" w:isXclude="1" />
  <Attribute w:n="Organization\OrganizationInformation\@xlink:label" w:isXclude="1" />
  <Attribute w:n="Organization\OrganizationInformation\@xlink:href" w:isXclude="1" />
  <Attribute w:n="Organization\OrganizationInformation\@DateValidFrom" w:isXclude="0" />
  <Attribute w:n="Organization\OrganizationInformation\@DateValidTo" w:isXclude="0" />
  <Attribute w:n="Organization\OrganizationInformation\@DataQualityType" w:isXclude="1" />
  <Attribute w:n="Organization\OrganizationInformation\@ValidFrom" w:isXclude="1" />
  <Attribute w:n="Organization\OrganizationInformation\@ValidTo" w:isXclude="1" />
  <Attribute w:n="Organization\OrganizationInformation\@LanguageCode" w:isXclude="0" />
- <Group w:n="OrganizationInformation\xpil:FreeTextLines" w:isXclude="0">
- <Element w:n="OrganizationInformation\xpil:FreeTextLines\xpil:FreeTextLine" w:isXclude=
  <Attribute w:n="xpil:FreeTextLines\xpil:FreeTextLine\@Type" w:isXclude="0" />
  <Attribute w:n="xpil:FreeTextLines\xpil:FreeTextLine\@DataQualityType" w:isXclude="0" />
  <Attribute w:n="xpil:FreeTextLines\xpil:FreeTextLine\@ValidFrom" w:isXclude="0" />
  <Attribute w:n="xpil:FreeTextLines\xpil:FreeTextLine\@ValidTo" w:isXclude="0" />
  </Element>
</Group>
- <Group w:n="OrganizationInformation\nxn:OrganisationName" w:isXclude="0">
  <Attribute w:n="OrganizationInformation\nxn:OrganisationName\@Type" w:isXclude="0" />
  <Attribute w:n="OrganizationInformation\nxn:OrganisationName\@OrganisationID" w:isXclude="0" />
  <Attribute w:n="OrganizationInformation\nxn:OrganisationName\@OrganisationIDType" w:isXclude="0" />
  <Attribute w:n="OrganizationInformation\nxn:OrganisationName\@ID" w:isXclude="0" />
  <Attribute w:n="OrganizationInformation\nxn:OrganisationName\@Usage" w:isXclude="1" />
  <Attribute w:n="OrganizationInformation\nxn:OrganisationName\@Status" w:isXclude="1" />
  <Attribute w:n="OrganizationInformation\nxn:OrganisationName\@xlink:type" w:isXclude="1" />
  <Attribute w:n="OrganizationInformation\nxn:OrganisationName\@xlink:label" w:isXclude="1" />
  <Attribute w:n="OrganizationInformation\nxn:OrganisationName\@xlink:href" w:isXclude="1" />
  <Attribute w:n="OrganizationInformation\nxn:OrganisationName\@DateValidFrom" w:isXclude="1" />
  <Attribute w:n="OrganizationInformation\nxn:OrganisationName\@DateValidTo" w:isXclude="1" />
```

EXCLUDE FLAG VALUE

Want Lists provide a handy way to catalogue the exchange model and can be re-used later by importing into other templates

Importing Want list operation



File Menu Option

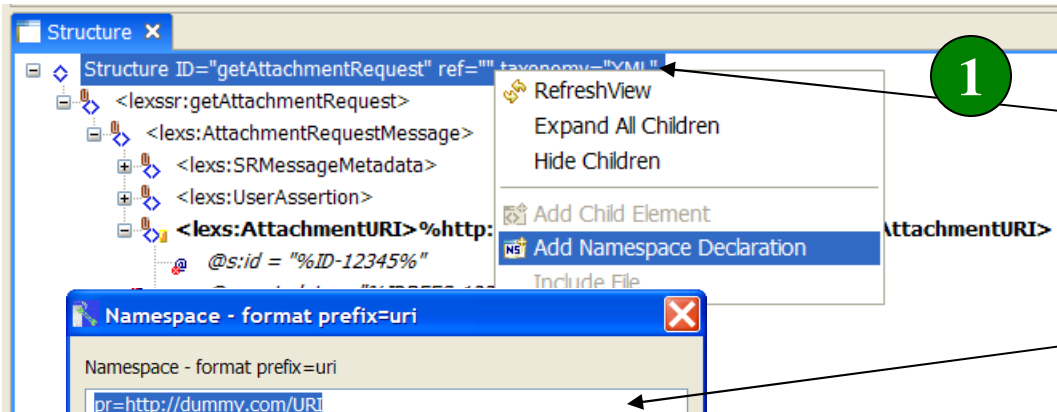
Select Import and specify filename of your existing wantlist xml.

Import process matches the path expressions in your want list to the XPath expressions in the template.

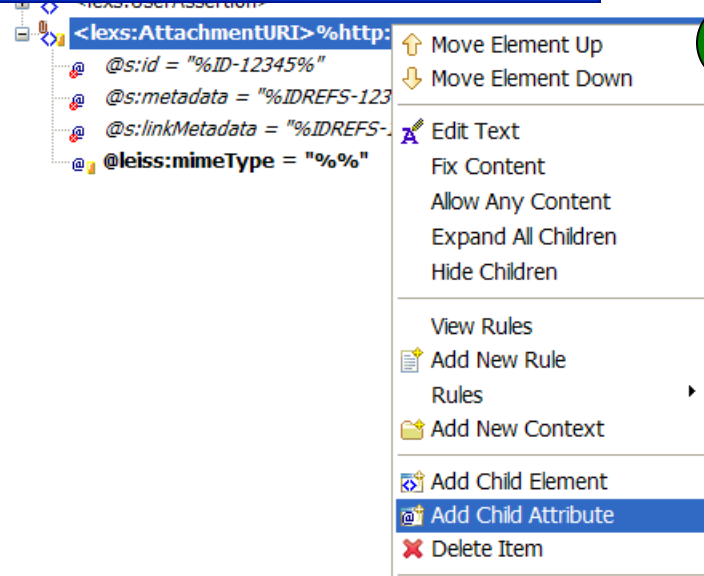
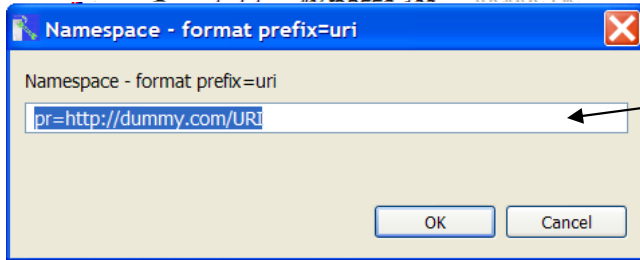
Exclude statements generated for matching items.

Makes it easy to re-apply a want list on new versions of schemas, or on similar schemas with same blocks of content – address, company, person, etc.

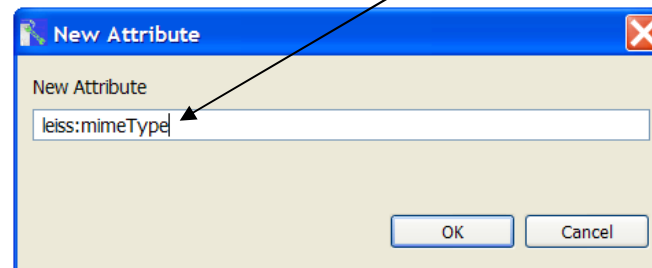
Adding New Domain Elements



Select root element node in structure; right mouse click for context menu; select "Add Namespace"; then enter prefix and URI



Select element node in structure; right mouse click for context menu; select "Add Child Attribute / Element"; then enter prefix and name



Rule Validation + Interoperability Checks

The image shows two overlapping windows from a software application. The top window is titled 'Documentation' and contains the 'CAM Template Validator' interface. The bottom window is titled 'CAM Template Editor' and shows a 'Tools' menu with 'Validate CAM Template' selected. A yellow callout box with a black border is positioned over the top window, containing two bullet points. Arrows point from the callout box to the statistics and warnings sections of the validator window, and from the 'Validate CAM Template' menu item to the callout box.

Documentation

file:///C:/Documents%20and%20Settings/dwebber/My%20Docume

CAM Template Validator

Version 1.02

CAM Template HEADER information:

Description: Generated for : getAttachmentRequest
Owner: To be Completed
Date: 2008-06-26T11:10:51
Version: 0.1 generator v1.04

- Number of Elements: 72
- Number of Attributes: 205
- Number of constraint Rules: 774
- Annotations: 28
- Hints: 1
- Excluded Elements: 0
- Excluded Attributes: 204

WARNINGS:

- No problems found

XSD SCHEMA GENERATION:

- Ignore Nillable = false
- Ignore SetLength = false
- Ignore SetLimit = false
- Use of nillable() can cause interoperability issues. Consider adding an attribute instead to convey meaning of empty items

Tools Menu Option

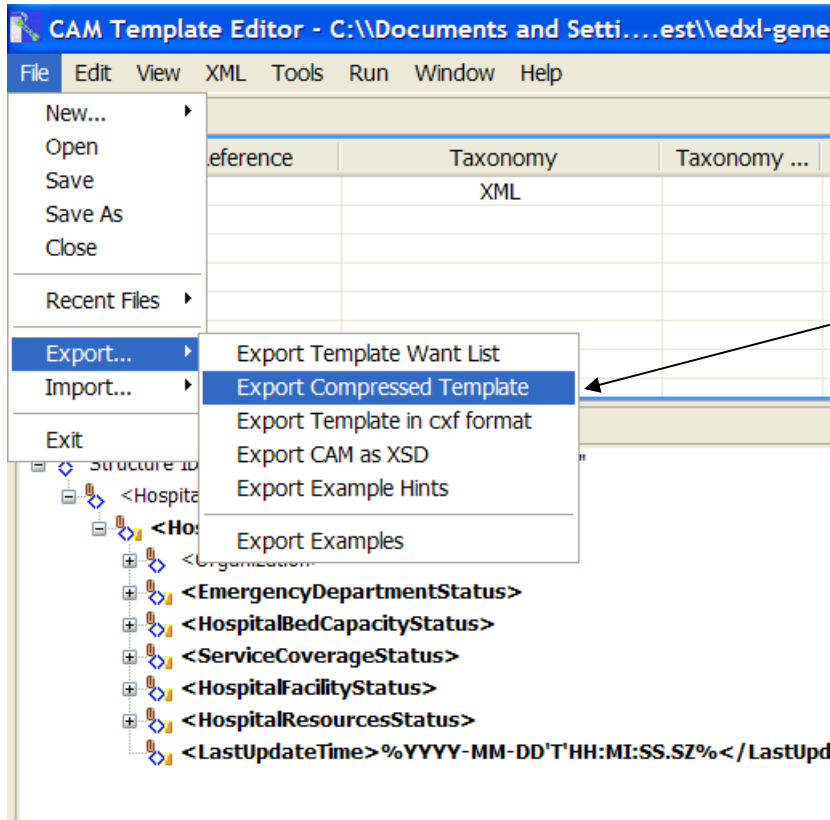
File Edit View XML **Tools** Run Window Help

- Refresh F5
- Reset Perspective
- Validate CAM Template**
- Convert UBL Codelist to CAM Lookup List

Callout Box:

- This option runs an analysis of your template and reports potential problems that it finds
- Also shows useful statistics about your template

Compress Operation



File Menu Option

Select option and specify filename for new copy of your template.

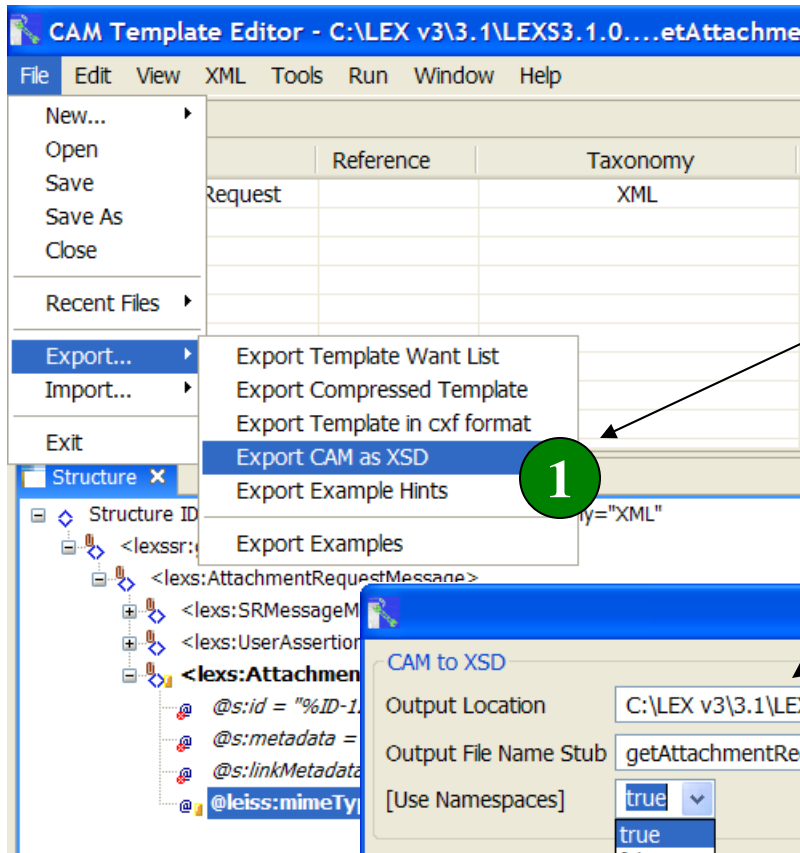
Compress process removes all rules and structure items marked with an exclude statement.

Note: ignores excludes that have a conditional context expression.

Compress is OPTIONAL. You only need to do it for two reasons:

- a) to generate documentation of only your structure items
- b) to generate a new subset XSD schema

Generating sub-set schema



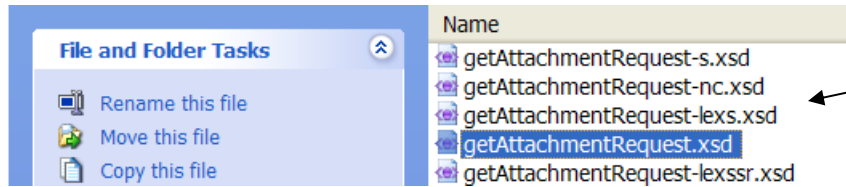
File Menu Option

Select Export CAM as XSD menu option

Confirm the location and filename, and namespace mode.

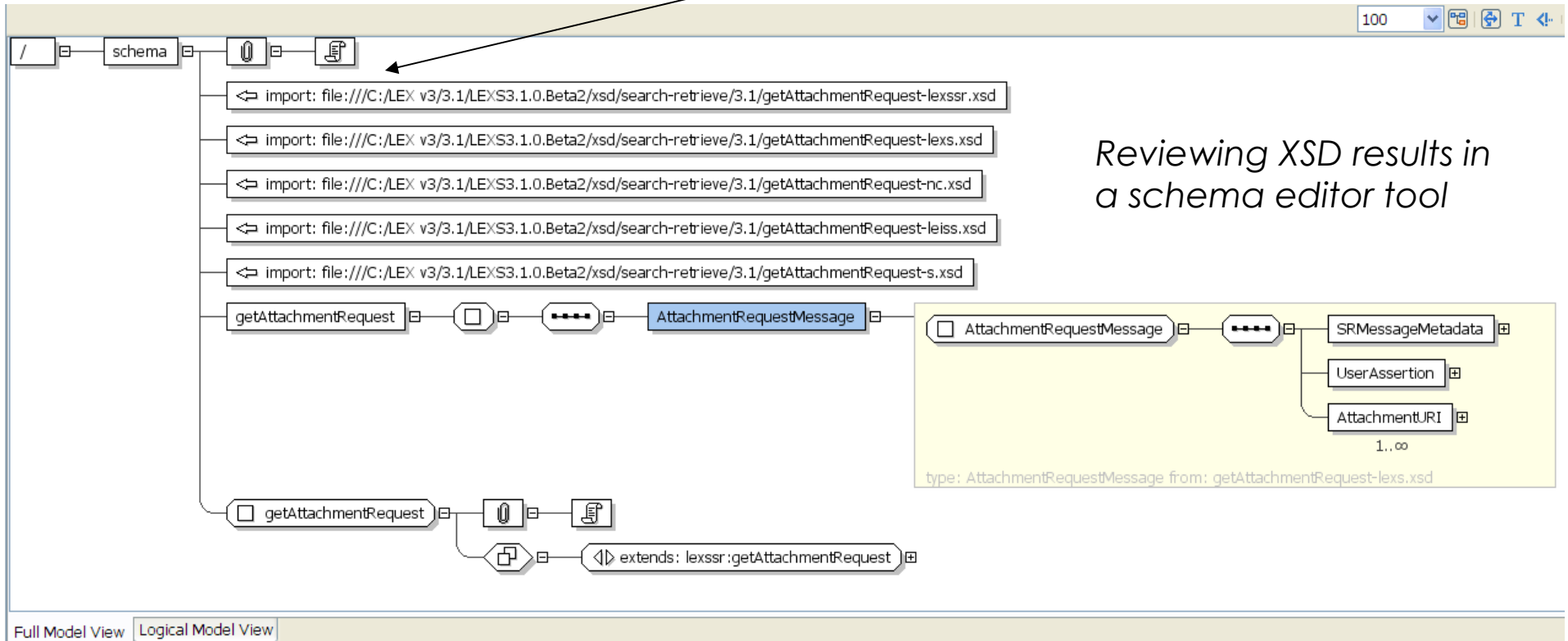
Select 'false' for namespace use will minimize the use and requirement for namespaces in the subset schema and corresponding XML instance documents.

Schema sub-set generated



Set of XSD files with filename and namespace suffix

Each namespace file is import for those specific type definitions



Reviewing XSD results in a schema editor tool

Constraint Schema Considerations

- The CAM template allows full use of XPath conditional expressions and a rich set of over 30 functions including:
 - `setNumberRange()`, `setLength()`, `setValue()`, `setLimit()`,
`setDateMask()`, `makeRepeatable()`, `restrictValues()`, `excludeTree()`
- Those that are compatible with XSD constraints will cause constraint schema assertions to be written out when exporting to schema
- In the advanced topics section we will look at cross field validations using XPath conditional rules

Generating Testing and Conformance Examples

Selecting valid and invalid modes
Run rules validation check
Customizing content with Hints

Test Case Generation Quick Start

The image shows a screenshot of the CAM Template Editor application. The File menu is open, showing options like New..., Open, Save, Save As, Close, Recent Files, Export..., Import..., and Exit. The 'Export Examples' option is selected. A dialog box titled 'Generate Examples' is displayed in the foreground. The dialog box has the following fields and controls:

- Output Location: C:\LEX v3\3.1\LEXS3.1.0.Beta2\xsd\ (with a Browse... button)
- Output file name stub: example
- Schema file name: (with a Browse... button)
- [Number of example]: 3
- [Number of repeats]: 3
- Optional Values: (dropdown menu)
- Generate Random Errors: (checkbox)
- Default Content: (dropdown menu)
- Use namespaces: (checkbox)
- Advanced Options section:
 - [Hints File]: (with a Browse... button)
 - Seed: (slider and input field)
- Buttons: OK and Cancel

Annotations with arrows point to the following elements:

- 'File Menu Option' points to the File menu.
- 'Default directory to write examples into' points to the Output Location field.
- 'Name to be used for the examples' points to the Output file name stub field.
- 'How many examples to create' points to the [Number of example] spinner.
- 'Repeating elements count' points to the [Number of repeats] spinner.
- 'OK' points to the OK button.

for Quick Test – just click “OK” to use default settings

Test Case Results

Documentation

file:///C:/Documents%20and%20Settings/dwebber/My%20Documents/CAM/EDXL/new%20xslt/latest/workspace/CAMEditor/temp_output.html Save As...

CAM Template Example Generator

Version 1.43

For output to file:///C:/Documents and Settings/dwebber/My Documents/test/EDXL-tests-cases-*.xml (3 examples)

CAM Template HEADER information:

Description: Generated for : HospitalStatus
Owner: To be Completed
Date: 2008-06-16T14:48:59
Version: 0.1 generator v1.01

SETTINGS:

Seed: 3246975
Hints file:
Optionals items: all
Default content: type
Random error mode: false
Schema: edxl-generated.xsd
Require namespaces: true

- [Outputting example : file:///C:/Documents and Settings/dwebber/My Documents/test/EDXL-tests-cases-1.xml](file:///C:/Documents and Settings/dwebber/My Documents/test/EDXL-tests-cases-1.xml)
- [Outputting example : file:///C:/Documents and Settings/dwebber/My Documents/test/EDXL-tests-cases-2.xml](file:///C:/Documents and Settings/dwebber/My Documents/test/EDXL-tests-cases-2.xml)
- [Outputting example : file:///C:/Documents and Settings/dwebber/My Documents/test/EDXL-tests-cases-3.xml](file:///C:/Documents and Settings/dwebber/My Documents/test/EDXL-tests-cases-3.xml)

Active links to view the generated examples

Advanced Generation Options

The screenshot shows the 'Generate Examples' dialog box with the following fields and options:

- Output Location:** C:\LEX v3\3.1\LEXS3.1.0.Beta2\xsd\ (with a 'Browse...' button)
- Output file name stub:** example
- Schema file name:** (with a 'Browse...' button)
- [Number of example]:** 3
- [Number of repeats]:** 3
- Optional Values:** (dropdown menu)
- Generate Random Errors:** (dropdown menu)
- Default Content:** (dropdown menu)
- Use namespaces:** (dropdown menu)
- Advanced Options:**
 - [Hints File]:** (with a 'Browse...' button)
 - Seed:** (with a slider and a text input field)

Annotations and their targets:

- Optional schema file validation link; use this to have example validate with schema or sub-set schema** (points to the 'Schema file name' field)
- Use content type or item name (name is useful for checking backend transform processing)** (points to the 'Optional Values' dropdown)
- How to handle optional items: all | random | none** (points to the 'Optional Values' dropdown)
- If you want deliberate errors for fail testing; (will give variety of data and structure errors)** (points to the 'Generate Random Errors' dropdown)
- Use namespaces or not; if 'false' is selected - then XML instances are created with minimized namespace usage.** (points to the 'Use namespaces' dropdown)
- Optional content hints (explained next)** (points to the '[Hints File]' field)
- Use slider to pick a specific seed value - or leave blank for random seed** (points to the 'Seed' slider)

Test Case Generator Feature Summary

- **Make both Pass / Fail testing examples**
- **Content hinting so examples use real not fake data**
- **Test optional item logic with: all / random / none**
- **Uses exclude() assertions so does not include those items – makes realistic examples of your use pattern**
- **Can pass in seed value – use when adding and testing hints (each test case is labelled with its seed value)**
- **Make hundreds of test cases without manual editing**
- **Can link test case to XSD schema for structure tests**
- **You can modify XSLT to meet own testing needs**

Run CAM Rules Check on Examples

Run Menu Option

The screenshot shows the CAM Template Editor interface. The 'Run' menu is open, showing 'Run JCam...' and 'Run XML File'. The 'Run JCam' wizard is displayed, with a green circle '1' next to the introductory text. The wizard fields include: Template: C:\LEX v3\3.1\LEXS3.1.0.Beta....3.1\getAttachmentRequest.cam; XML File: (empty); Structure ID: getAttachmentRequest. Under 'Run Time Options', 'VALIDATE' is selected. An 'Open' dialog box is overlaid on the wizard, showing a file selection process. A green circle '2' is next to the 'example-1.xml' file in the file list. The 'Finish' button in the wizard is highlighted with a green circle '3'. An arrow points from the 'Finish' button to the 'Run Results' tab in the main editor.

Pick Test Case Example to VALIDATE; click Finish to run validation rules

Review validation results

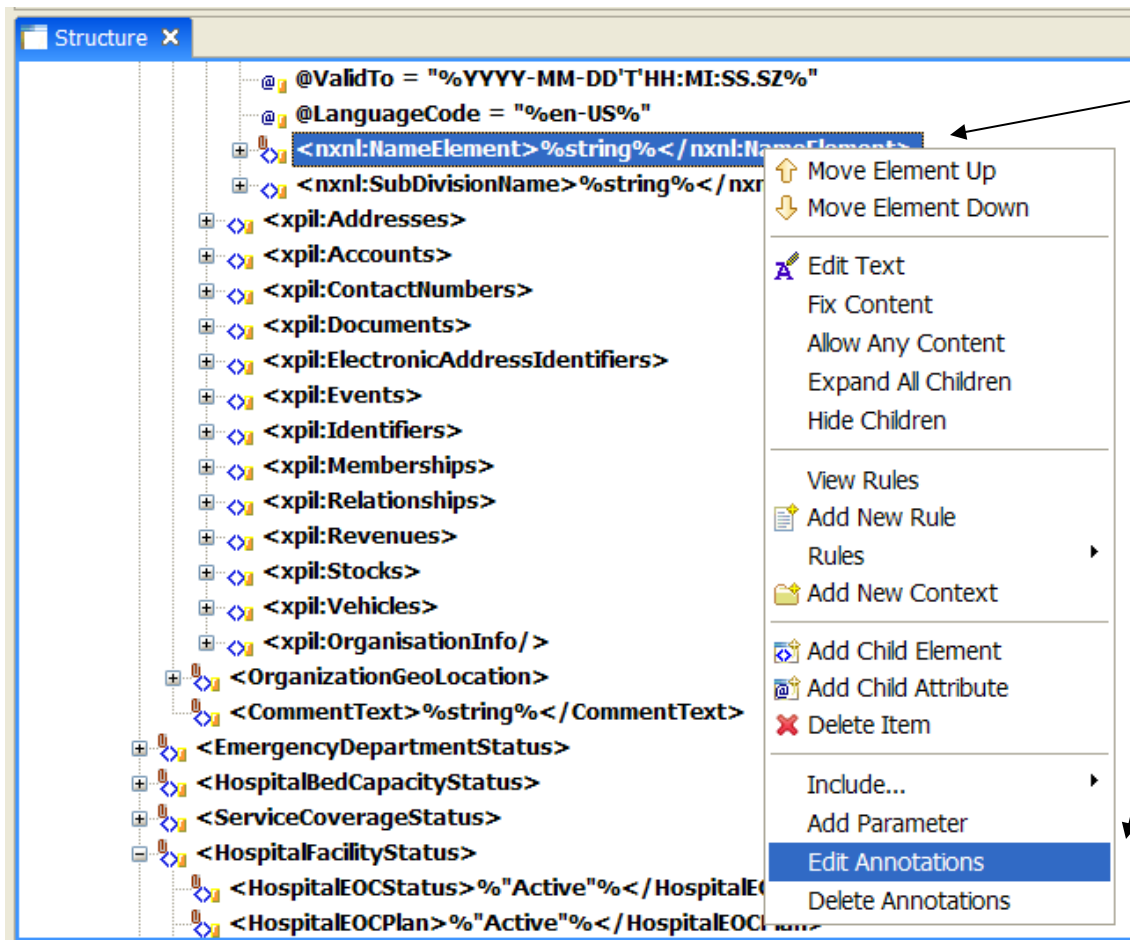
Rules View Imports Properties Parameters Namespaces Run Results x

- lexssr:getAttachmentRequest - Number of Errors = 0
 - <lexs:AttachmentRequestMessage>
 - <lexs:SRMessageMetadata>
 - <lexs:UserAssertion>
 - <lexs:AttachmentURI>http://wiki.oasis-open.org/cam/XSD_and_CAM</lexs:AttachmentURI>

Content Hinting Mechanisms

- Designed to create realistic data examples
- Hints can be provided in two ways
- Firstly - using 'Value' notes in annotations on specific items in the structure editor
- Second – create your own Hints XML file and add matching rules to globally apply across your template(s) – e.g. FirstName, LastName, Address, BirthDate, etc.
- Can export from one template, import into another

First Approach: annotation Value Hints

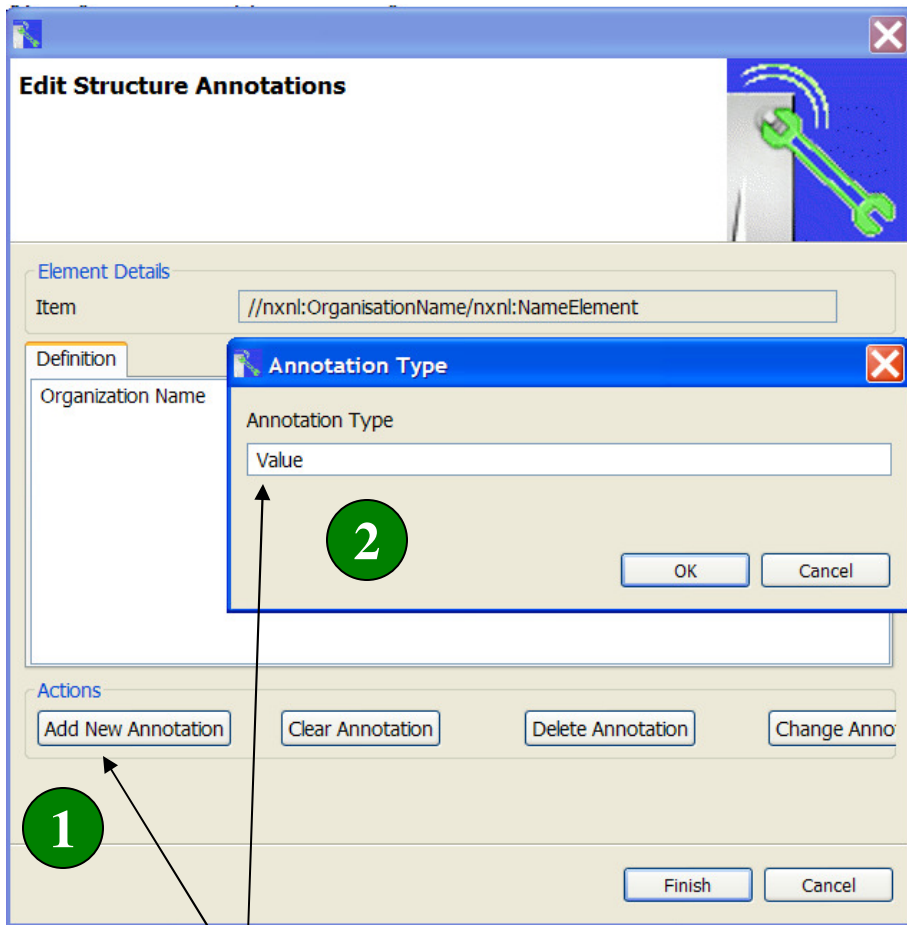


Select focus on structure item

Invoke action menu (right mouse click)

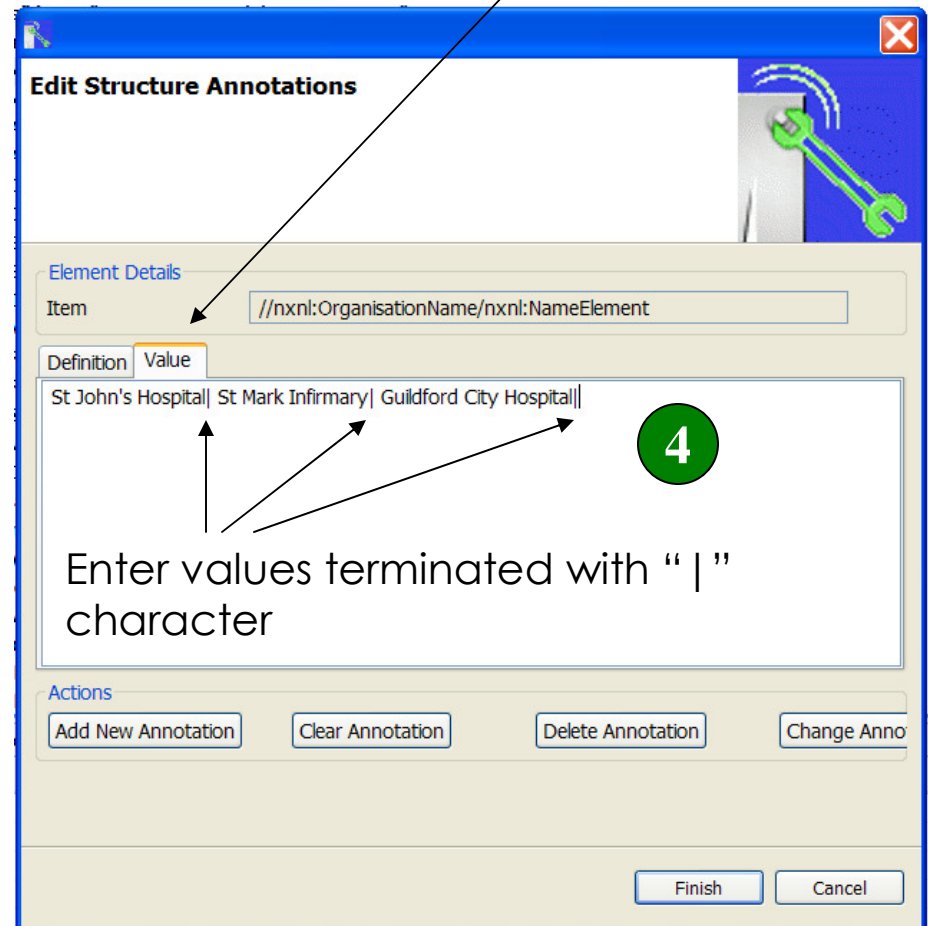
Choose "Edit Annotations"

Then add Value annotation item



Click on "Add New", then enter "Value" as Type and confirm "OK"

3 Select "Value" tab



5 Re-run Example export to see results

Second: Hints File Mechanism (XML file)

```
<?xml version="1.0" encoding="UTF-8"?>
<as:HINTS xmlns:as="http://www.oasis-open.org/committees/cam" CAMlevel="1" version="1.0">
  <as:Header>
    <as:Description>HospitalStatus content hints</as:Description>
    <as:Owner>To be Completed</as:Owner>
    <as:Version>0.1</as:Version>
    <as:DateTime>2008-05-23T14:02:45</as:DateTime>
  </as:Header>
  <as:HintDetails>
    <as:Hint like="name" with="first">
      <as:Value>Fred</as:Value>
      <as:Value>John</as:Value>
    </as:Hint>
    <as:Hint key="ContactEmailID">
      <as:Value>support@cbc.hhs.gov</as:Value>
      <as:Value>helpdesk@info.dhs.gov</as:Value>
      <as:Value>help@www.ijis.org</as:Value>
    </as:Hint>
    <as:Hint key="TelephoneNumberFullID"> [5 lines]
    <as:Hint key="AttachmentURI"> [4 lines]
    <as:Hint key="PersonName">
      <as:Value>John Smithly</as:Value>
      <as:Value>Frank O'Toole</as:Value>
      <as:Value>Bill McFinkish</as:Value>
    </as:Hint>
    <as:Hint key="AddressLine"> [3 lines]
    <as:Hint key="ElectronicAddressIdentifier"> [4 lines]
    <as:Hint key="NameElement" parent="Country">
      <as:Value>USA</as:Value>
      <as:Value>United States</as:Value>
      <as:Value>US</as:Value>
    </as:Hint>
    <as:Hint key="NameElement" parent="PersonName">
      <as:Value>Fred Smythe</as:Value>
      <as:Value>William Clinton</as:Value>
      <as:Value>George Washington</as:Value>
    </as:Hint>
  </as:HintDetails>
</as:HINTS>
```

1 like / with partial name matching
use for component match on items – e.g. first with name matches <nxnl:first_name>

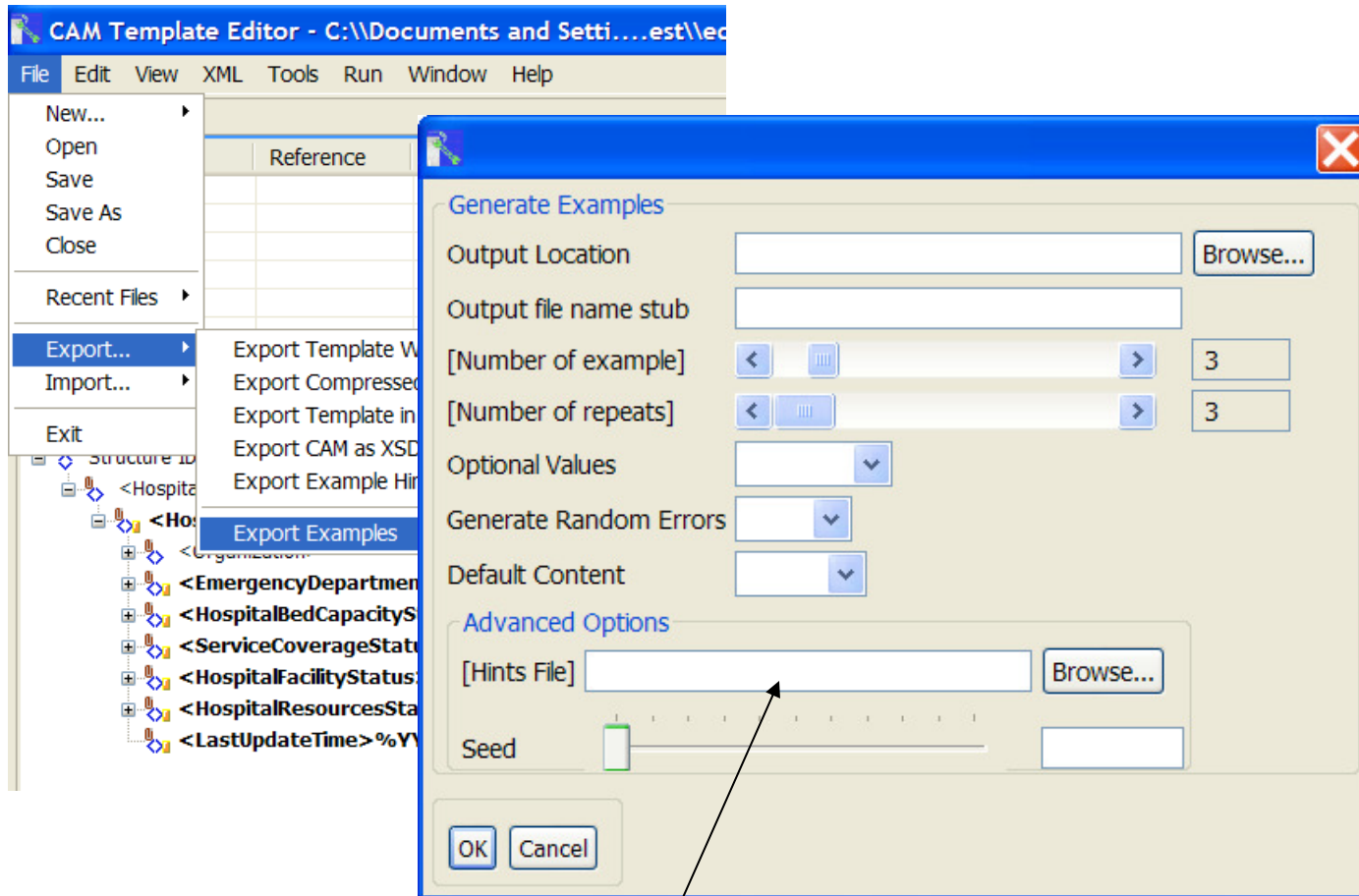
2 key matching on tag name
use for exact match of items

3 key / parent path matching
use when same name occurs within different parents – e.g. Country and Person / NameElement with different content and context

Note: matching is case sensitive but ignores namespaces

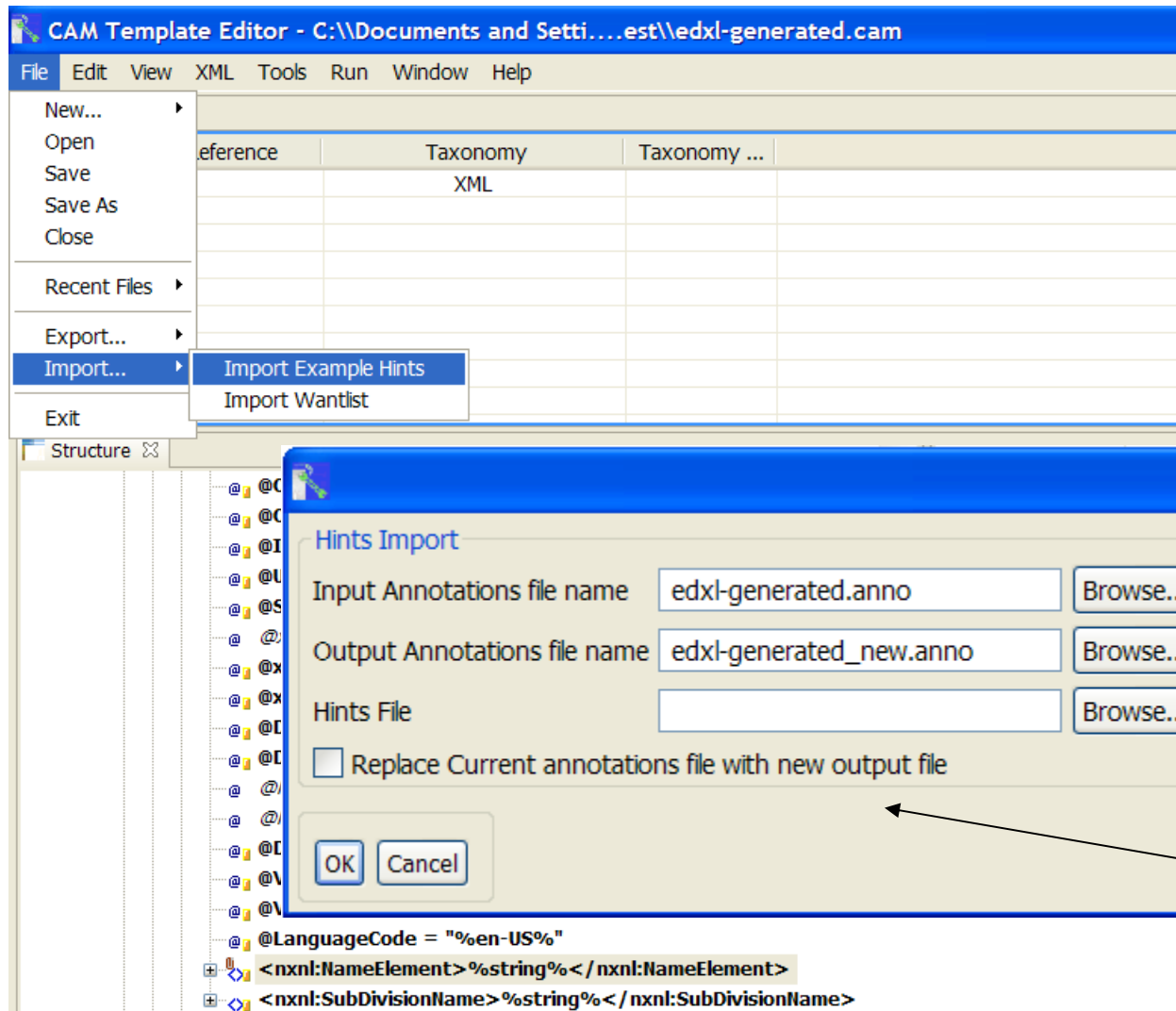
TIP: can use Export Hints to create initial XML file for editing

A- Using Examples Generator with Hints



Select XML hints file to be used here

B- Import Hints into Annotations (merge)



set and select as needed

Option to auto-reload new anno file into current template

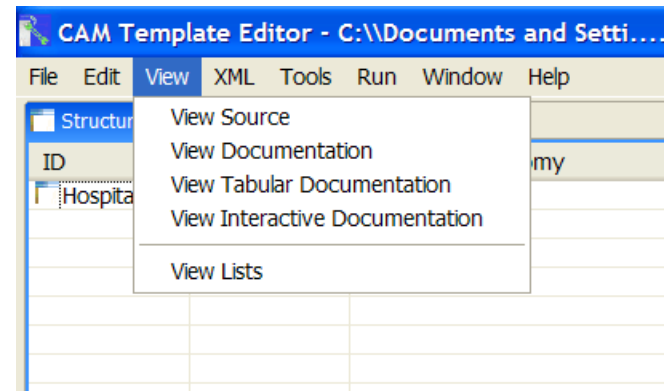
Documentation

Default reporting options

Documentation Layouts

■ Five options

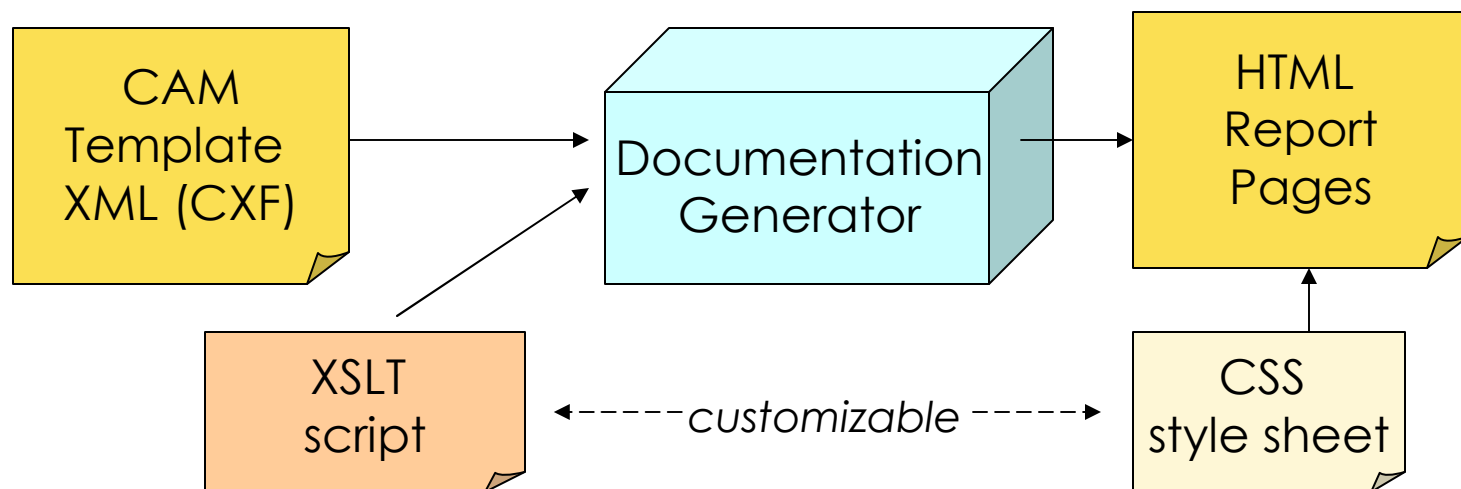
- Source XML
- Component details (XML)
- Tabular format (HTML)
- Interactive web page (wiki)
- Code list



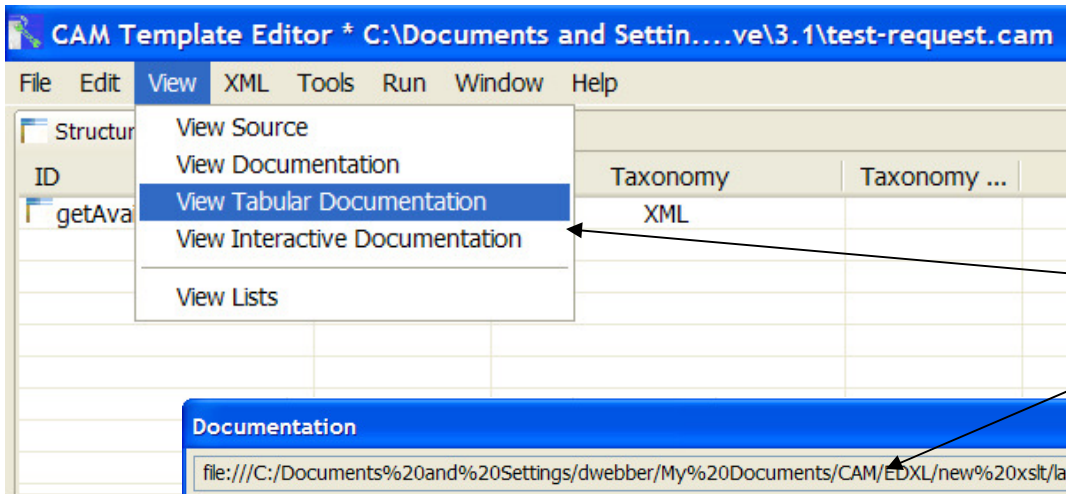
- Tabular format page layout for data analyst use and designed to make rules and use patterns clear
- Each documentation layout XSLT script can be customized as desired

Open Documentation Mechanism

- Structure Editor runs XSLT on CAM CXF to output results as HTML document
- External CSS style sheet controls HTML content formatting, colors, fonts.
- Editor Preferences menu allows overriding of default documentation style sheets

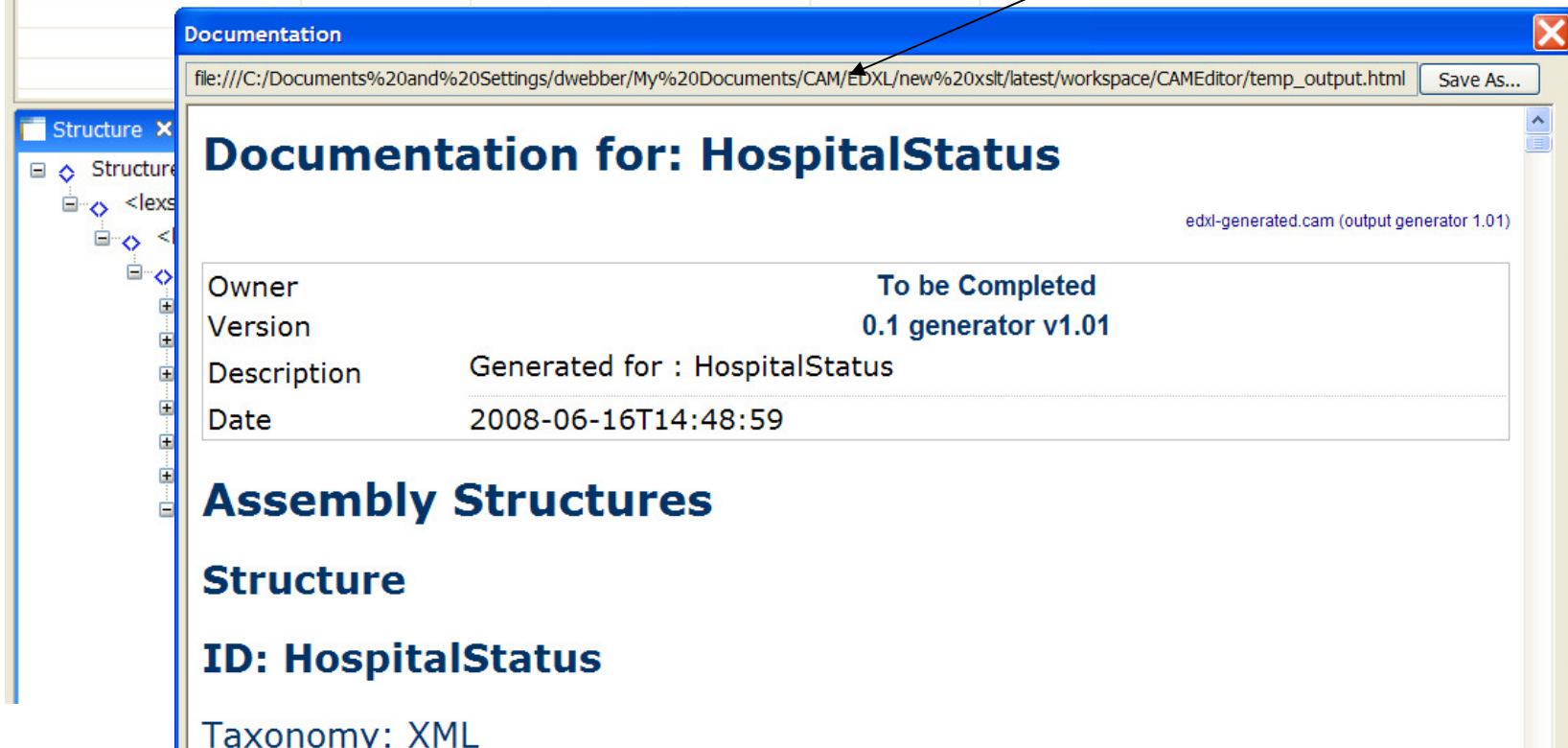


Tabular Documentation



Documentation Menu Option

Select Format, resulting HTML opens in browser viewer



Tabular HTML Content Details

Documentation		
file:///C:/Documents%20and%20Settings/dwebber/My%20Documents/CAM/EDXL/new%20xslt/latest/workspace/CAMEditor/temp_output.html Save As...		
	optional	Clear use pattern
@Abbreviation	optional allowed values are 'true' 'false' only	
axal:PostalDeliveryPoint	optional with content type of "anyAttribute"	
@Type	optional allowed values are 'GPOBox' 'POBox' 'LockedBag' 'MailStop' 'PigeonHole' 'PrivateBag' only	Extended Code list handling
@DataQualityType	optional allowed values are 'Valid' 'Invalid' only	Annotations
	Valid - The data was validated and is considered to be true and correct. Invalid - Indicates that at least some part of the content is known to be incorrect.	
@ValidFrom	optional if string-length(//axal:PostalDeliveryPoint/@ValidFrom) <26 then setDateMask(YYYY-MM-DD'THH:MI:SSZ) end if; if string-length(//axal:PostalDeliveryPoint/@ValidFrom) >25 then setDateMask(YYYY-MM-DD'THH:MI:SS.SZ) end if;	Enhanced Data type Logic
	XPath references and functions	
@ValidTo	optional	

Summary

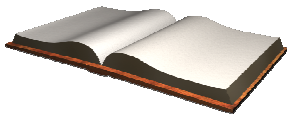
- Ingesting XSD
- Creating use pattern (aka want list)
- Generating test examples
- Hints system
- Generate XSD schema subset
- Running tests

IEPD Package Contents Review

1



2



Documentation
(Word / PDF / OpenDoc
Excel / HTML)



3



Want List in XML
XSD subset



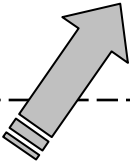
4



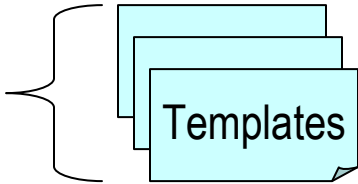
Examples &
Test Cases



5



Structure
Rules
Context
Vocabulary



Summary

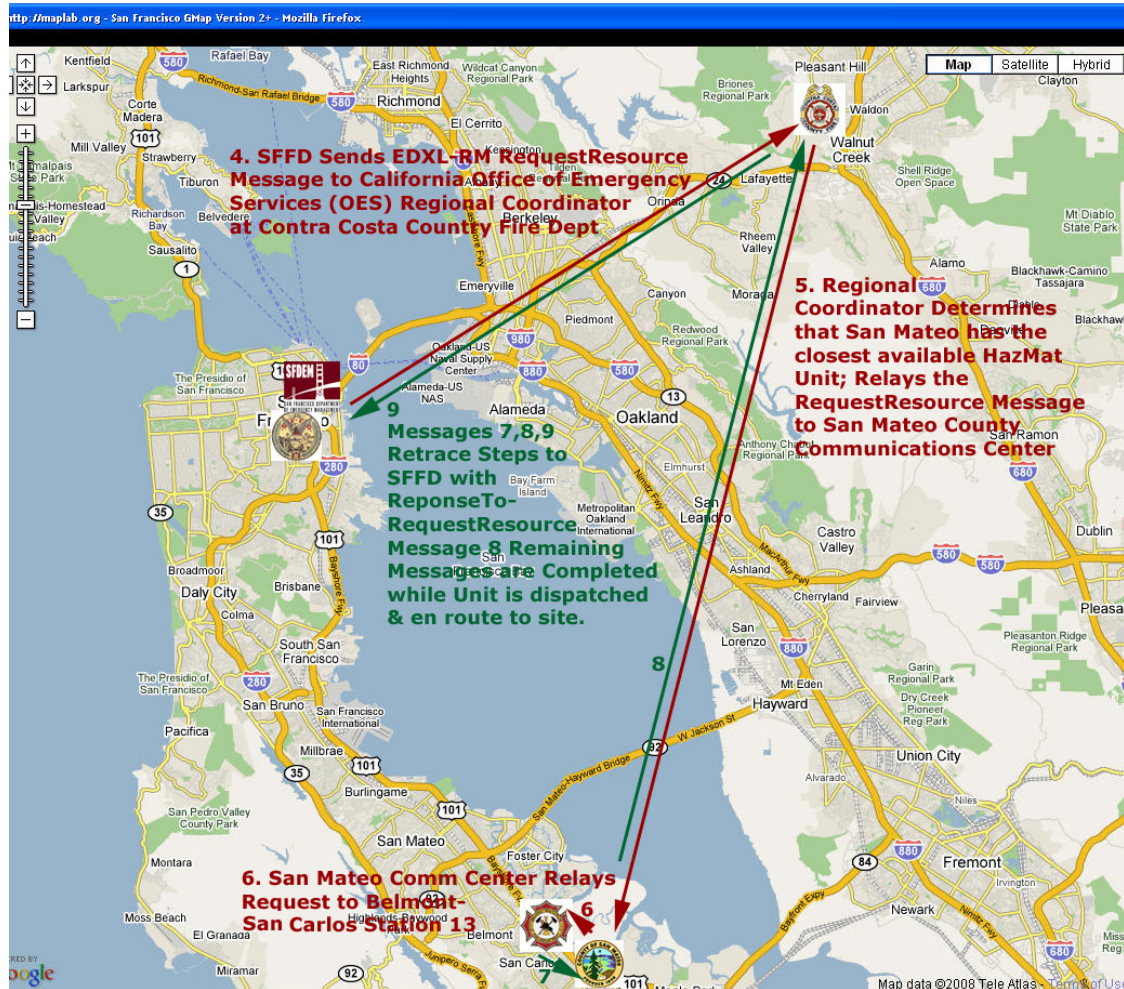
- **Capabilities covered**
 - Ingesting XSD
 - Creating use pattern (aka want list)
 - Generating test examples
 - Hints system
 - Generate XSD schema subset
 - Running tests
- **Applicable to board range of domains and schemas**
- **Enhanced interoperability through consistent method, testing and shared clear exchange package definitions**
- **Enables SOA implementations**

Advanced Techniques

- Extending rules for actual use cases
- Using XPath selector wizard
- Handling ingestion recursion issues
- SourceForge XSLT repository

Extending Rules for actual use cases

- Emergency Response Services Workflow using OASIS EDXL exchanges



Illustrative EDXL requirements

- When AdultICU Bed type > 0
 - AND Triage Quantity > 5 **Facility Matching**
- Require Facility State = CA, NV, NM **Region Restriction**
- When Admissions Total > 50
 - AND Deaths > 0 **Outbreak alerting**

Template rules syntax

```
<as:BusinessUseContext>
```

```
<as:Rules>
```

```
<as:default>
```

```
<as:context>
```

```
<as:constraint condition="//Activity24Hr /Admissions > '50'  
and( // Activity24Hr /Deaths > '0')"  
action="restrictValues(//  
Activity24Hr /Admissions, 'Alert – possible outbreak') "/>
```

```
<as:constraint action="restrictValues(//ns5:AdministrativeArea  
/ns5:NameElement , 'CA | NV | NM')"/>
```

```
</as:context>
```

```
</as:default>
```

```
</as:Rules>
```

```
</as:BusinessUseContext>
```

XPath selector wizard

Add New Constraint Wizard

Item and Action Details

Item: @leiss:mimeType

XPath: //leix:AttachmentURI/@leiss:mimeType

Rule XPath: None Full All Children

Position Parent Children

Action: makeOptional

Context and Condition

Context Position: [dropdown]

Condition: None

Rule Category: default context

Conditional? No Yes

Annotation

Annotation: [text area]

Finish Cancel

XPath syntax is automatically generated for you

Check boxes allow user to select scope of rule action

Conditional context rules can be optionally used

Documentation notes can be added here

Ingestion Recursion Handling

- In XSD schema syntax – recursive links and type references are not marked as such
- Ingestion process has no way of knowing when recursion is about to occur
- Solution – add annotation to XSD schema element definition:
 - `<xsd:annotation>`
 - `<xsd:documentation>`
 - `<recursive/>`
 - `</xsd:documentation>`
 - `</xsd:annotation>`

SourceForge XSLT svn repository

- Using any svn client the XSLT scripts can be retrieved from:
 - <https://camprocessor.svn.sourceforge.net/svnroot/camprocessor/camed/uk.org.jcam.camed/trunk/xsl/>

“CAM Kit” of XSLT tools used

■ XSD 2 Schema

- expands original target schema resolving imports and includes

■ XSD 2 CAM

- extracts structure and rules and builds template from schema

■ XML 2 Wantlist

- builds want list from any XML instance (uses Level Depth setting)

■ CAM 2 Examples

- Generates a collection of XML instance test cases

■ Import/Export Hints

- Manage and apply content hinting across CAM templates

■ CAM 2 XSD export

- Creates a subset XSD from CAM template (applies want list)

Resources / Installation

**Selection of useful links and
additional technical details**

Quick Install for Eclipse jCAM Editor

- **Download the latest editor ZIP file from the download site on SourceForge:**
 - <http://downloads.sourceforge.net/sourceforge/camprocessor>
- **Create folder c:\jCAM**
- **Open up the ZIP file and extract the CAMed folder into c:\jCAM\CAMed**
- **From the c:\jCAM\CAMed directory click on the CAMed.exe icon to run the program**
- **Create shortcut to the CAMed.exe by right click on icon and select create shortcut**
- **Drag and drop shortcut to desktop**

NIEM IEPD Ancillary XSLT

- Ability to create a spreadsheet of NIEM core component elements using lookup from schema components
- 4 files
 - NIEM-repository.xsl
 - NIEM-lookup.xsl
 - NIEM-repository.xml
 - Property.xml
- The repository is extracted from the main NIEM properties.xml (exported from NIEM Access database)
- NIEM-lookup then reads the CXF of template and writes out cross-reference xml that is then opened in Excel as a spreadsheet

Resources:

www.jcam.org.uk

wiki.oasis-open.org/cam

www.oasis-open.org/committees/cam

docs.oasis-open.org/cam

www.oasis-open.org/committees/emergency

www.niem.gov



Credits:

A special mention for our contributors to the CAM and jCAM work:

- UK - Martin Roberts and team from BTplc
- US - Michael Sorens for review and testing

