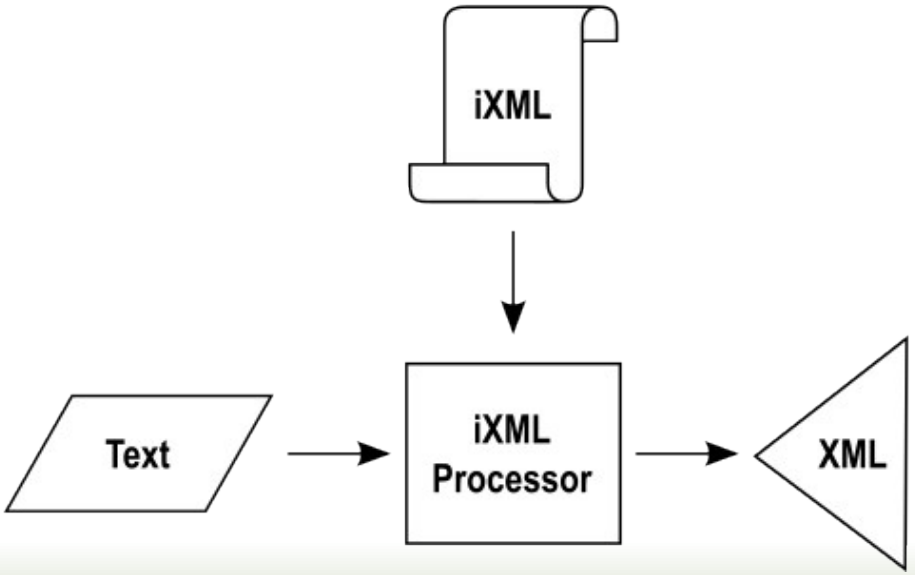
Configuring text conversion environments employing iXML

Crane-txt2xml

G. Ken Holman
Crane Softwrights Ltd.

How I learned iXML

```
date = day, " ", month, " ", year.
-digit = ["0"-"9"].
year = digit, digit, digit, digit.
month: "January"; "February"; "March";
     "April"; "May"; "June";
     "July"; "August"; "September";
     "October"; "November"; "December".
day: digit; digit, digit.
```

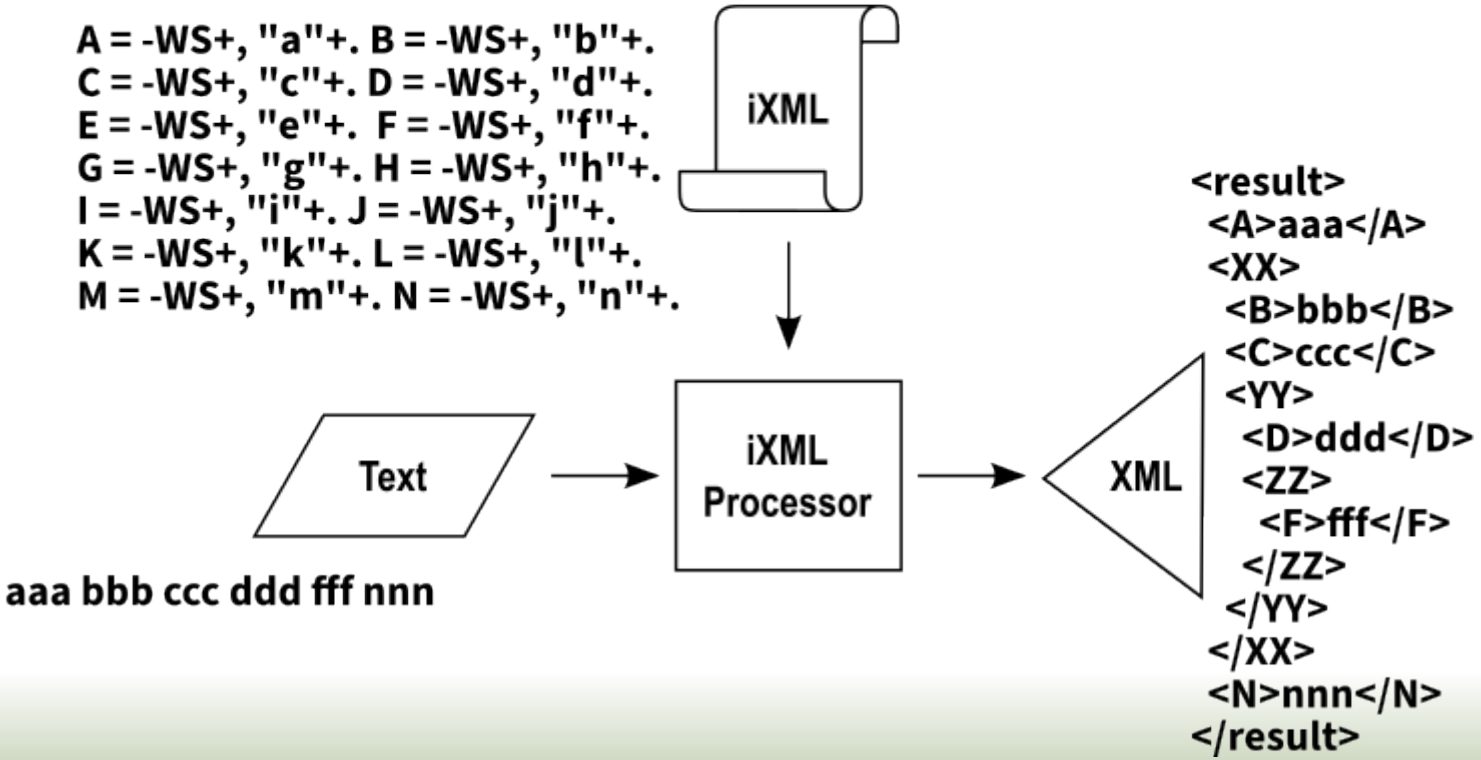


4 November 2021

```
<date>
  <day>4</day>
  <month>November</month>
  <year>2021</year>
</date>
```

My Revelation

```
result = A, XX, N?.
XX = B, C?, YY?, K?, L?, M?.
YY = D?, E?, ZZ?, I?, J?.
ZZ = F?, G?, H?.

SP     = -[ " " | #9 | #D | #A ].
WS     = -SP, -wsMore. wsMore  = -SP, -wsMore | .

A = -WS+, "a"+. B = -WS+, "b"+.
C = -WS+, "c"+. D = -WS+, "d"+.
E = -WS+, "e"+.  F = -WS+, "f"+.
G = -WS+, "g"+. H = -WS+, "h"+.
I = -WS+, "i"+. J = -WS+, "j"+.
K = -WS+, "k"+. L = -WS+, "l"+.
M = -WS+, "m"+. N = -WS+, "n"+.
```




```
<result>
 <A>aaa</A>
 <XX>
  <B>bbb</B>
  <C>ccc</C>
  <YY>
   <D>ddd</D>
   <ZZ>
    <F>fff</F>
   </ZZ>
  </YY>
 </XX>
 <N>nnn</N>
</result>
```

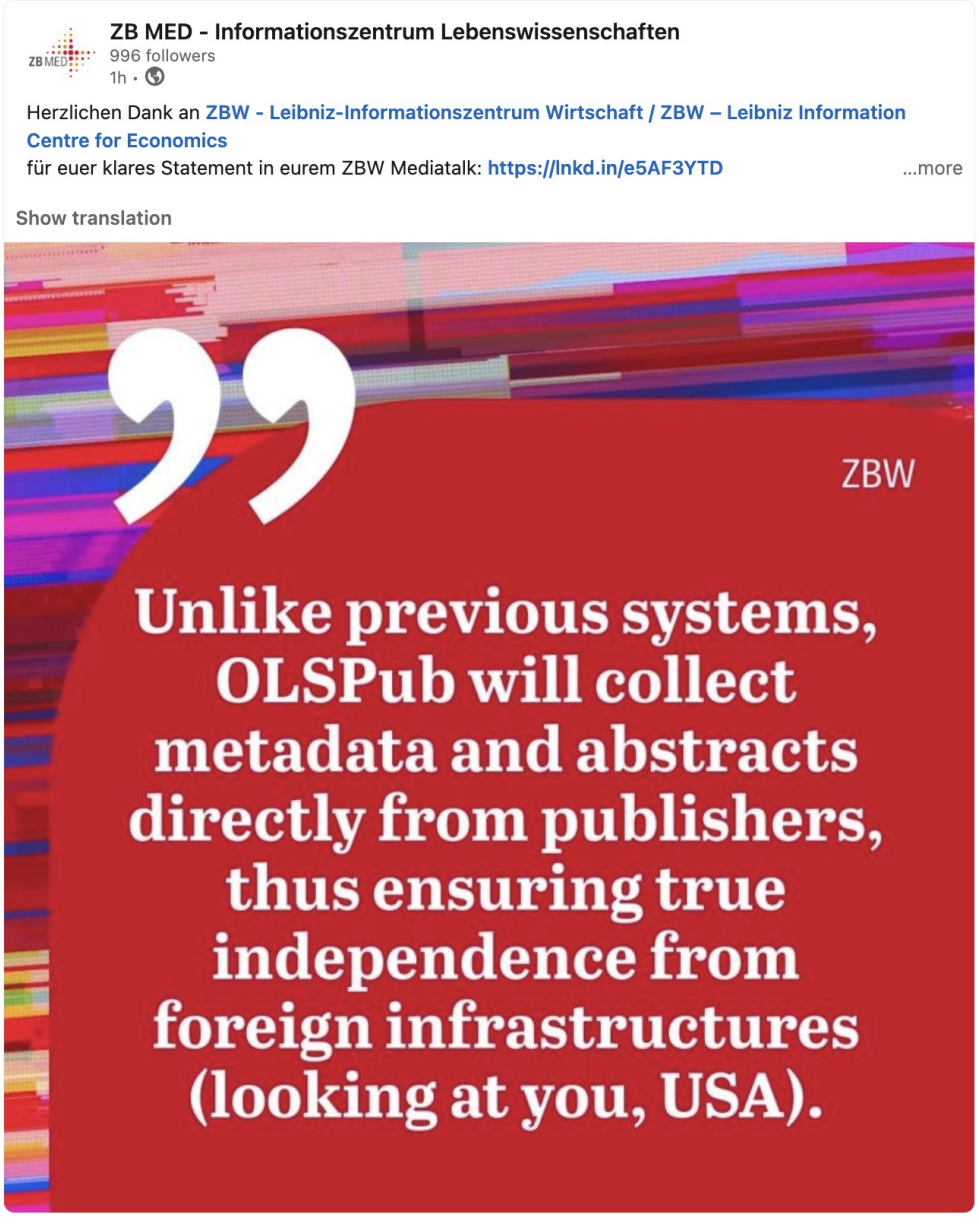
My current (XML-related) passions (1 of 3)

Information Centre for Life Sciences

ZB MED

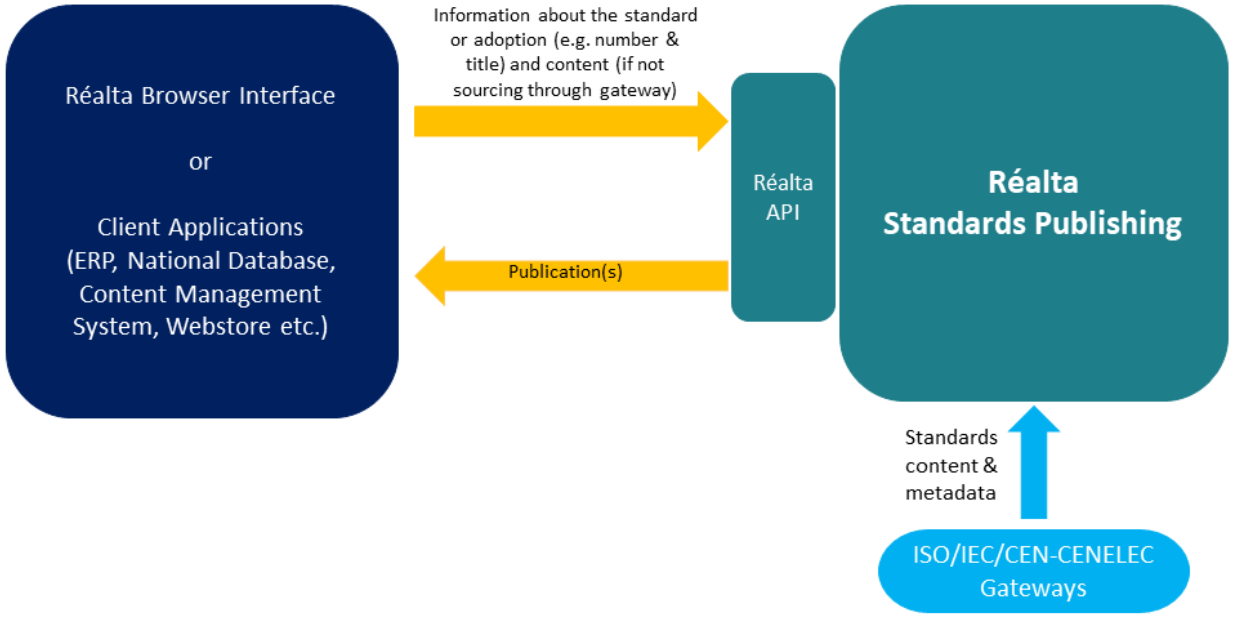
OLSPub

Open Life Science Publications Database: Building a Resilient European Biomedical and Life Science Infrastructure



Feb 27, 2026 08:40Z

My current (XML-related) passions (2 of 3)

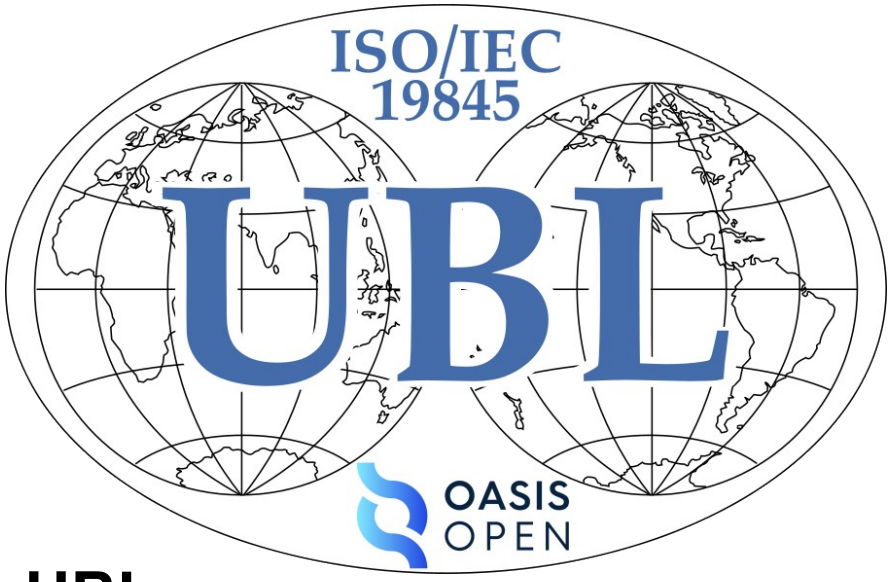


RealtaOnline.com

Réalta Online Publishing Solutions Limited (Ireland)

Using simple NISO-STS metadata to specify the adoption of dynamically-fetched content of specified standards from regional and international bodies.

My current (XML-related) passions (3 of 3)



UBL -
Universal Business Language
OASIS -
Organization for the Advancement of Structured Information Standards

ISO/IEC 19845

Peppol
The future is open

20+ countries with UBL e-invoicing mandates (and other UBL documents)

US Federal Reserve Bank of Minneapolis

UBL-based specification for rollout in the USA

What I need for users needing PubMed XML

PubMed Input Metadata

realtaonline/
PubNote

Making XML metadata approachable for OLSPub and PubMed users, <PubNote> works for you between the angle brackets.

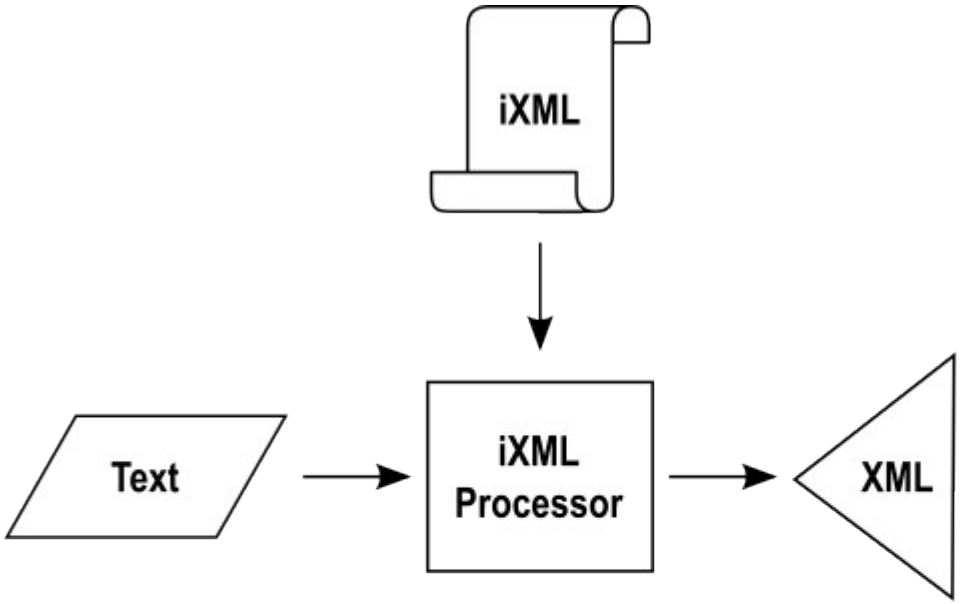
2 Contributors | 0 Issues | 2 Discussions | 0 Stars | 0 Forks

GITHUB.COM

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ArticleSet PUBLIC
  "-//NLM//DTD PubMed 2.8//EN"
  "https://dtd.nlm.nih.gov/ncbi/pubmed/in/PubMed.dtd">
<ArticleSet>
  <Article>
    <Journal>
      <PublisherName>DOVE Medical Press</PublisherName>
      <JournalTitle>Journal of Placeholder Studies</JournalTitle>
      <Issn>0000-0000</Issn>
      <Volume>12</Volume>
      <Issue>3</Issue>
      <PubDate PubStatus="ppublish">
        <Year>2000</Year>
        <Month>01</Month>
      </PubDate>
    </Journal>
    <ArticleTitle>A Study on Placeholder Data</ArticleTitle>
    <FirstPage>2955</FirstPage>
    <LastPage>2967</LastPage>
    <ELocationID EIdType="doi">10.9999/fake.doi.00001</ELocationID>
    <Language>EN</Language>
    <AuthorList>
      <Author>
        <FirstName>Emma</FirstName>
        <LastName>Crusoe</LastName>
        <Affiliation>Department of Placeholder Studies, Fictional University
```

Article
- **Journal**
  - **PublisherName**: DOVE Medical Press
  - **JournalTitle**: Journal of Placeholder Studies
  - **Issn**: 0000-0000
  - **Volume**: 12
  - **Issue**: 3
  - **PubDate** [PubStatus="ppublish"]
    - **Year**: 2000
    - **Month**: 01
- **ArticleTitle**: A Study on Placeholder Data
- **FirstPage**: 2955 [LZero="delete"]
- **LastPage**: 2967
- **ELocationID**: 10.9999/fake.doi.00001 [EIdType="doi" ValidYN="Y"]
- **Language**: EN
- **AuthorList**
  - **Author**
    - **FirstName**: Emma [EmptyYN="N"]
    - **LastName**: Crusoe
    - **Affiliation**: Department of Placeholder Studies, Fictional University
  - **Author**

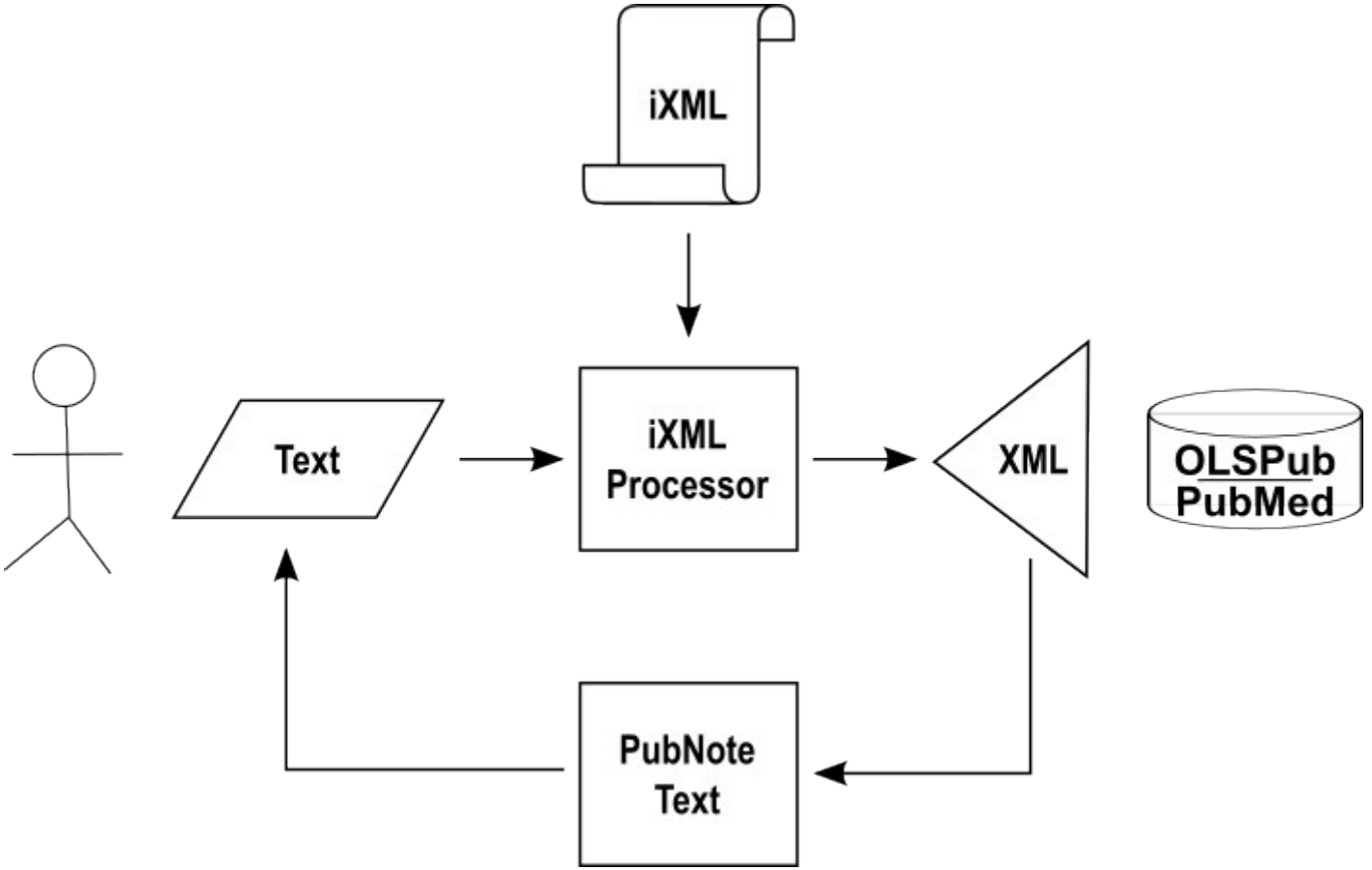
Could I use iXML to make PubMed data entry easy?



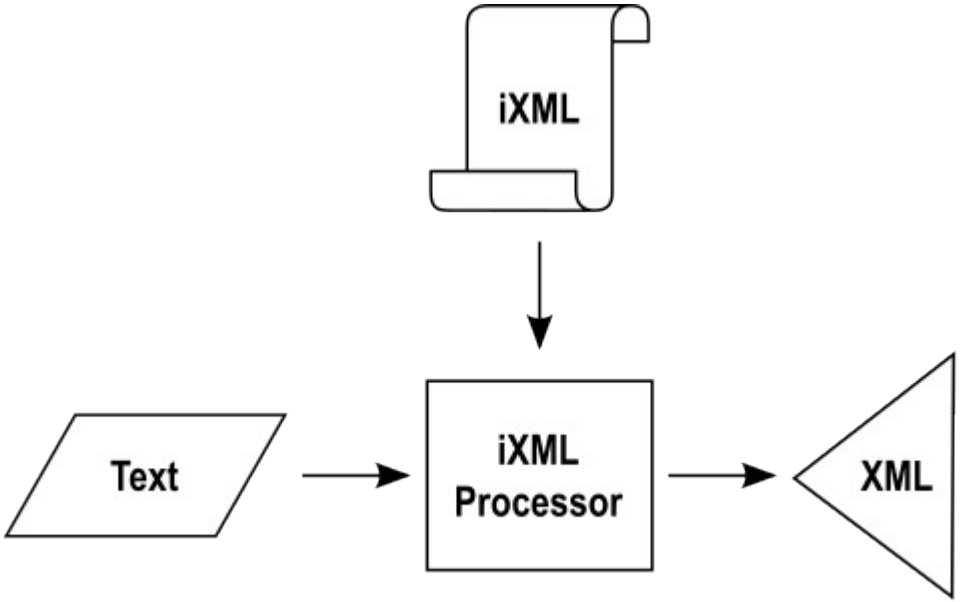

```
Article:
  Journal:
    PublisherName: "DOVE Medical Press"
    JournalTitle: "Journal of Placeholder Studies"
    Issn: 0000-0000
    Volume: 12
    Issue: 3
    PubDate: @PubStatus:ppublish
      Year: 2000
      Month: 01
  ArticleTitle: A Study on Placeholder Data
  FirstPage: @LZero:delete 2955
  LastPage: 2967
  ELocationID: @EIdType:doi @ValidYN:Y 10.9999/fake.doi.00001
  Language: EN
  AuthorList:
    Author:
      FirstName: @EmptyYN=N Emma
      LastName: Crusoe
      Affiliation: "Department of Placeholder Studies, Fictional University"
```

```
<ArticleSet>
 <Article>
  <Journal>
   <PublisherName>DOVE Medical Press</PublisherName>
   <JournalTitle>Journal of Placeholder Studies</JournalTitle>
   <Issn>0000-0000</Issn>
   <Volume>12</Volume>
   <Issue>3</Issue>
   <PubDate PubStatus="ppublish">
    <Year>2000</Year>
    <Month>01</Month>
    </PubDate>
   </Journal>
  <ArticleTitle>A Study on Placeholder Data</ArticleTitle>
  <FirstPage>2955</FirstPage>
  <LastPage>2967</LastPage>
  <ELocationID EIdType="doi">10.9999/fake.doi.00001</ELocationID>
  <Language>EN</Language>
  <AuthorList>
   <Author>
    <FirstName>Emma</FirstName>
    <LastName>Crusoe</LastName>
    <Affiliation>Department of Placeholder Studies, Fictional University
```

Could I introduce round-tripping for non-XML'ers?



What I need for users needing NISO-STS



standard:
front:
std-meta: @std-meta-type: national
title-wrap:
release-version:
permissions: copyright-year: copyright-holder:

std-meta: @std-meta-type: european
custom-meta-group: @originator: realta
custom-meta: meta-name: realta-fetch cen xml meta-value: 74573

std-meta: @std-meta-type: international
custom-meta-group: @originator: realta
custom-meta: meta-name: realta-fetch iso xml meta-value: 83438

```
<?xml version="1.0" encoding="UTF-8"?>
<standard xmlns:xlink="http://www.w3.org/1999/xlink">
 <front>
  <std-meta std-meta-type="national">
   <title-wrap>
   </title-wrap>
   <release-version/>
   <permissions>
    <copyright-year> </copyright-year>
    <copyright-holder/>
   </permissions>
  </std-meta>
  <std-meta std-meta-type="european">
   <custom-meta-group originator="realta">
    <custom-meta>
     <meta-name>realta-fetch cen xml</meta-name>
     <meta-value>74573</meta-value>
    </custom-meta>
   </custom-meta-group>
  </std-meta>
  <std-meta std-meta-type="international">
   <custom-meta-group originator="realta">
    <custom-meta>
     <meta-name>realta-fetch iso pdf</meta-name>
     <meta-value>83438</meta-value>
    </custom-meta>
   </custom-meta-group>
  </std-meta>
 </front>
 <body/>
 <back/>
</standard>
```

What I need for users needing UBL XML

Universal Business Language (UBL)

```
<?xml version="1.0" encoding="UTF-8"?>
<Invoice xmlns:cbc="urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComp
onents-2" xmlns:cac="urn:oasis:names:specification:ubl:schema:xsd:CommonAggregat
eComponents-2" xmlns="urn:oasis:names:specification:ubl:schema:xsd:Invoice-2">
  <cbc:ID>1</cbc:ID>
  <cbc:IssueDate>2026-02-01</cbc:IssueDate>
  <cac:AccountingSupplierParty>
    <cac:Party>
      <cac:PartyName>
        <cbc:Name>Joe Bloggs</cbc:Name>
      </cac:PartyName>
    </cac:Party>
  </cac:AccountingSupplierParty>
  <cac:LegalMonetaryTotal>
    <cbc:PayableAmount currencyID='CAD'>3000.00</cbc:PayableAmount>
  </cac:LegalMonetaryTotal>
  <cac:InvoiceLine>
    <cbc:ID>1</cbc:ID>
    <cbc:LineExtensionAmount currencyID="CAD">1000.00</cbc:LineExtensionAmount>
    <cac:Item>
      <cbc:Description>Gizmo</cbc:Description>
    </cac:Item>
  </cac:InvoiceLine>
  <cac:InvoiceLine>
    <cbc:ID>2</cbc:ID>
    <cbc:LineExtensionAmount currencyID="CAD">2000.00</cbc:LineExtensionAmount>
    <cac:Item>
      <cbc:Description>Omzig</cbc:Description>
    </cac:Item>
  </cac:InvoiceLine>
</Invoice>
```

The UBL document model

UBL-Invoice-2.4

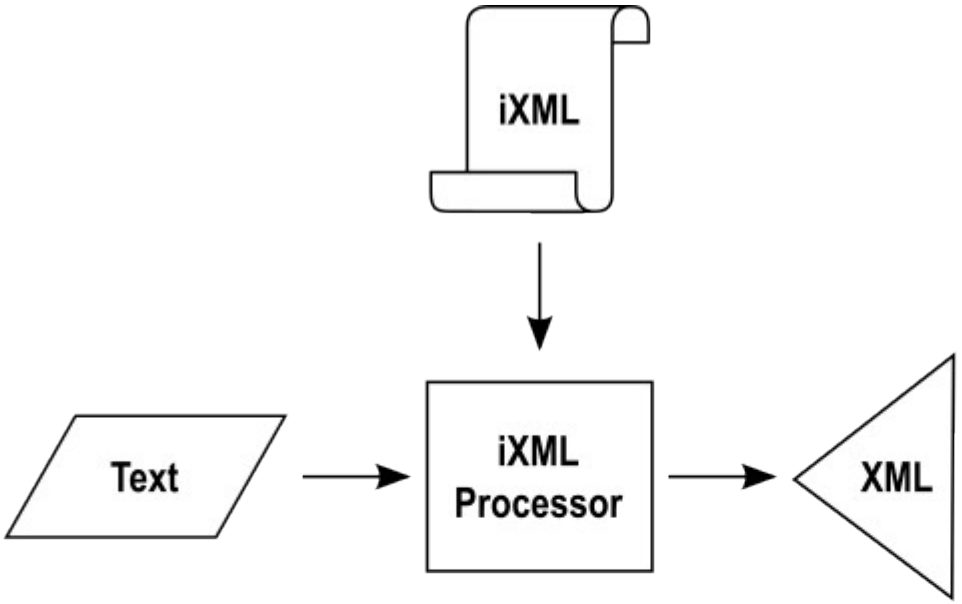
	Name	Card.	Rep. Term	Alt. Business Terms
2	Invoice	A document used to request payment.		
3	UBLVersionID	Identifies the earliest version of the UBL 2 sc		
		0..1	Identifier	
4	CustomizationID	Identifies a user-defined customization of UB		
		0..1	Identifier	
5	ProfileID	Identifies a user-defined profile of the custom		
		0..1	Identifier	
6	ProfileExecutionID	Identifies an instance of executing a profile, t		
		0..1	Identifier	
7	**ID**	An identifier for this document, assigned by t		
		1	Identifier	Invoice Number
8	CopyIndicator	Indicates whether this document is a copy (tr		
		0..1	Indicator	
9	UUID	A universally unique identifier for an instance		
		0..1	Identifier	
10	**IssueDate**	The date, assigned by the sender, on which t		
		1	Date	Invoice Date
11	IssueTime	The time, assigned by the sender, at which th		
51	PaymentAlternativeExchangeRate	The exchange rate between the document c		
		0..1	Exchange Rate	
52	TaxTotal	The total amount of a specific type of tax.		
		0..n	Tax Total	
53	WithholdingTaxTotal	The total withholding tax.		
		0..n	Tax Total	
54	**LegalMonetaryTotal**	The total amount payable on the Invoice, inc		
		1	Monetary Total	
55	**InvoiceLine**	A line describing an invoice item.		
		1..n	Invoice Line	

1553	MonetaryTotal	A class to define a monetar
1554	LineExtensionAmount	The monetary amount of an
		0..1 Amount
1555	TaxExclusiveAmount	The monetary amount of an
		0..1 Amount
1556	TaxInclusiveAmount	The monetary amount inclu
		0..1 Amount
1557	AllowanceTotalAmount	The total monetary amount
1563	PayableRoundingAmount	The rounding amount (posit
		0..1 Amount
1564	**PayableAmount**	The amount of the monetary
		1 Amount
1565	PayableAlternativeAmount	The amount of the monetary
		0..1 Amount

Dict. Entry Name (Base type XSD)	Content and supplementary components	
	(property)	(attribute)
Amount. Type (xsd:decimal)	**AmountContent**	A number of monetary units
	AmountCurrencyIdentifier	**currencyID**
	AmountCurrencyCodeListVersionIdentifier	currencyCodeListVersionID

https://docs.oasis-open.org/ubl/os-UBL-2.4/mod/summary/reports/All-UBL-2.4-Documents.html

Could I use iXML to make UBL data entry easy?



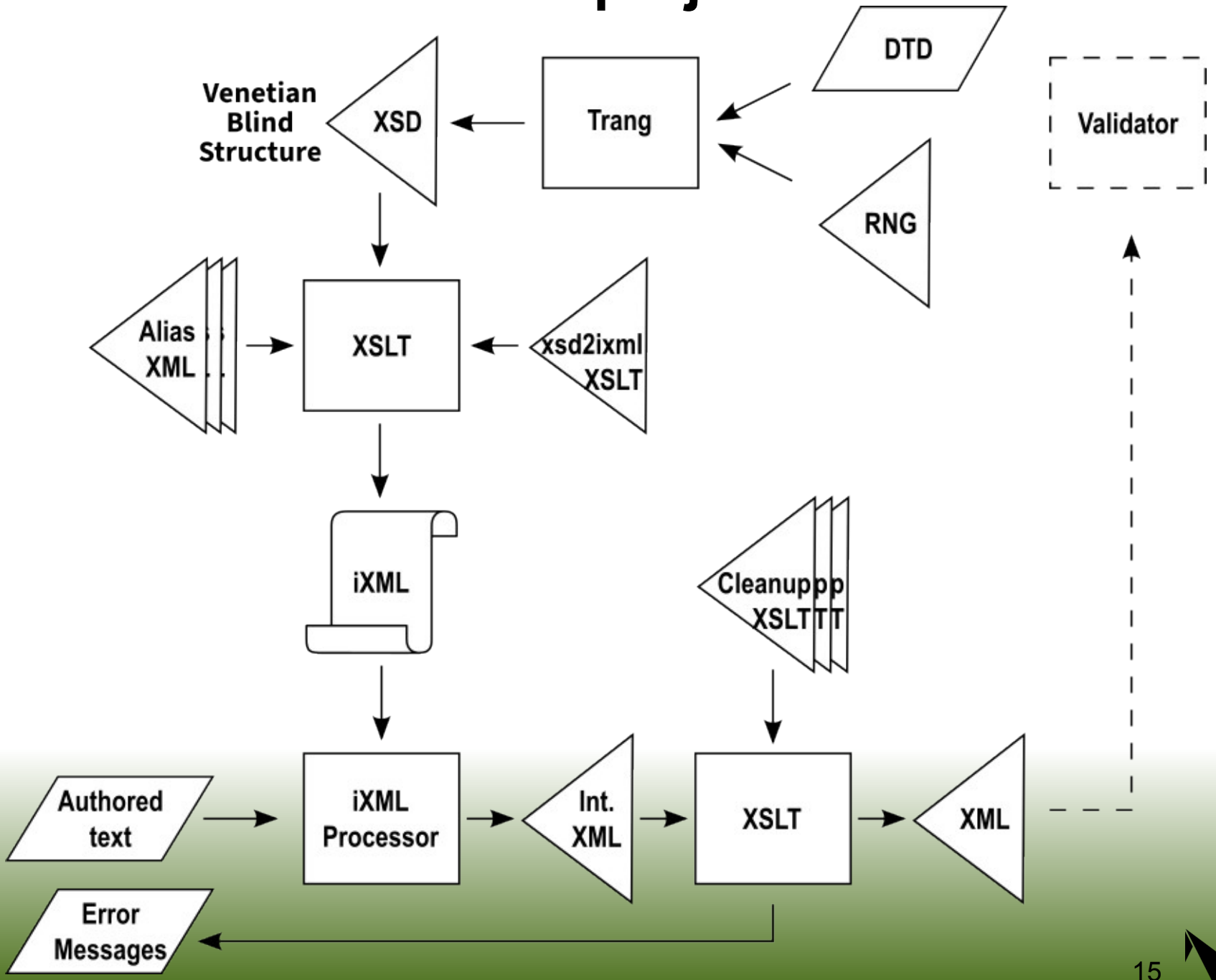

```
Invoice:
ID: 1
IssueDate: 2026-02-01
AccountingSupplierParty: Party: PartyName: Name: "Joe Bloggs"
LegalMonetaryTotal: PayableAmount: @currencyID:CAD 3000.00
Invoice  Line: ID: 1 LineExtensionAmount:
 @currencyID:"CAD" 1000.00
Item: Description: Gizmo
InvoiceLine: ID: 1 LineExtensionAmount: @currencyID: CAD
2000.00
Item: Description: "Omzig \"Alternative\""
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Invoice xmlns:cbc="urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2"
xmlns:cac="urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2"
xmlns="urn:oasis:names:specification:ubl:schema:xsd:Invoice-2">
 <cbc:ID>1</cbc:ID>
 <cbc:IssueDate>2026-02-01</cbc:IssueDate>
 <cac:AccountingSupplierParty>
  <cac:Party>
   <cac:PartyName>
    <cbc:Name>Joe Bloggs</cbc:Name>
   </cac:PartyName>
  </cac:Party>
 </cac:AccountingSupplierParty>
 <cac:LegalMonetaryTotal>
  <cbc:PayableAmount currencyID="CAD">3000.00</cbc:PayableAmount>
 </cac:LegalMonetaryTotal>
 <cac:InvoiceLine>
  <cbc:ID>1</cbc:ID>
  <cbc:LineExtensionAmount currencyID="CAD">1000.00</cbc:LineExtensionAmount>
  <cac:Item>
   <cbc:Description>Gizmo</cbc:Description>
  </cac:Item>
 </cac:InvoiceLine>
 <cac:InvoiceLine>
  <cbc:ID>2</cbc:ID>
  <cbc:LineExtensionAmount currencyID="CAD">2000.00</cbc:LineExtensionAmount>
  <cac:Item>
   <cbc:Description>Omzig "Alternative"</cbc:Description>
  </cac:Item>
 </cac:InvoiceLine>
</Invoice>
```

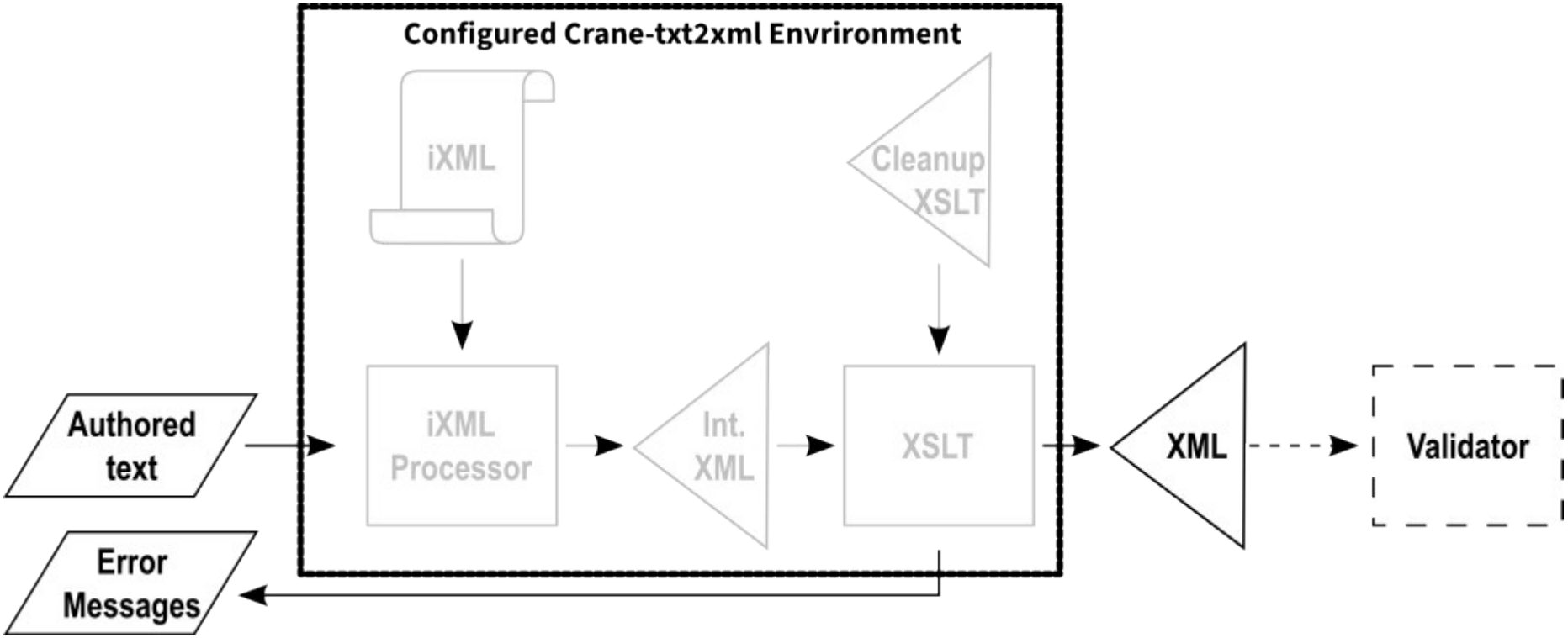
Mockup iXML (by hand) of UBL subset

```
{[+pragma nineml "https://nineml.org/ns/pragma/"]}
{[+nineml ns "https://cranesoftwrights.github.io/ns/Crane-txt2xml?
output=urn:oasis:names:specification:ubl:schema:xsd:Invoice-2"]}

Invoice = __WS*, -"Invoice:", __content, ( ProfileID?, ID, UUID?, IssueDate, IssueTime?, AccountingSupplierParty,
LegalMonetaryTotal, InvoiceLine+ ), __WS*.

ProfileID = __WS*, -"ProfileID:", __content, ().
ID = __WS*, -"ID:", __content, ().
UUID = __WS*, -"UUID:", __content, ().
IssueDate = __WS*, -"IssueDate:", __content, ().
IssueTime = __WS*, -"IssueTime:", __content, ().

AccountingSupplierParty = __WS*, -"AccountingSupplierParty:", __content, ( AdditionalAccountID?, Party* ).
AdditionalAccountID = __WS*, -"AdditionalAccountID:", __content, ().

Party = __WS*, -"Party:", __content, ( PartyName* ).
PartyName = __WS*, -"Party", __WS*, -"Name:", __content, ( Name ).
Name = __WS*, -"Name:", __content, ().

LegalMonetaryTotal = __WS*, -"LegalMonetaryTotal:", __content, ( PrepaidAmount?, PayableAmount,
PayableAlternativeAmount? ).
PrepaidAmount = __WS*, -"PrepaidAmount:", __content, ().
PayableAmount = __WS*, -"PayableAmount:", __content, ().
PayableAlternativeAmount = __WS*, -"PayableAlternativeAmount:", __content, ().

InvoiceLine = __WS*, -"Invoice", __WS*, -"Line:", __content, ( ID, Note?, LineExtensionAmount, TaxPointDate?, Item ).
Note = __WS*, -"Note:", __content, ().
LineExtensionAmount = __WS*, -"LineExtensionAmount:", __content, ().
TaxPointDate = __WS*, -"TaxPointDate:", __content, ().

Item = __WS*, -"Item:", __content, ( Description ).
Description = __WS*, -"Description:", __content, ().
Price = __WS*, -"Price:", __content, ().
```

Crane-txt2xml project mission



What the author sees



Where I am going with this

- first: element-content ISO/OASIS UBL
  - release to waiting community users from Crane's GitHub
    - CraneSoftwrights/Crane-txt2ubl
    - *not there yet!*
- next: limited-mixed-content PubMed input
  - release to biomedical community from Réalta's <PubNote>
  - realtaonline/PubNote/PubNote-txt2pubmedin (-pubmedout?)
  - *not there yet!*
- then: NISO-STS metadata
  - plans not ironed out yet; may be free or part of the commercial service offered to national standards bodies

- reusable portion also maintained in GitHub:
  - CraneSoftwrights/Crane-txt2xml
  - *not there yet!*

Anticipated reveal of the working environment:

- **XML Prague 2026**

Questions for the experts, please

1) Consider that I have used the choices between two syntaxes to capture a string differently, a quoted or unquoted value:

```
{something} = __value.
{something-else} = __value.

__value = __quoted | __unquoted, __WS.
-__quoted = -'"', -'"' | -'"', -__more_quoted, -'"'.
-__more_quoted = ( __BS | ~['"' | "\"] ), __more_quoted?.
-__unquoted = ( __BS | __NWSBQAC ) , __unquoted?.
```

... how can I "recurse"(?)/"reparse(?)" the resulting `__value` through another set of parse rules that don't need to consider which matching patterns got me there?

~~**Or ... could I run a separate iXML parse from inside (free) XSLT to do that reparsing?**~~

YES! (Thank you, John L and Norm TW)

BTW ... that `__value` is the simple Markdown subset needed for inline markup

Questions for the experts, please (cont.)

2) I'm anxious to employ any modularization and name-scoping techniques so that the generated portion of the grammar doesn't interfere with the boilerplate portion of the grammar

- **the output element names are governed by an arbitrary XSD schema**
- **without name-scoping it is possible the names used in the boilerplate will conflict with the names in the schema**
- **using "__" prefix currently for the common Crane-xml2txt iXML rules**

What is available?

Could some name-scoping be mappable to output XML namespaces?

Questions for the experts, please (cont.)

3) Please could there be an end-of-file terminator that matches the end of the input stream?

What I have requires the authored input to be appended with a space in case ending with unquoted value (I can't get rid of the space because of context):

```
__value = __quoted | __unquoted, __WS.
```

I would like to have some kind of signal keyword, imagining "¤" as syntax:

```
__value = __quoted | __unquoted, ( __WS | [¤EOF] ).
```

3b) Or maybe something like:

```
--trailing-newline
```

Force a newline at the end of the output?

but what I need is: "Force a newline at the end of the input"

Questions for the experts, please (cont.)

4) Am I blind to iXML tool features that can annotate the output XML (in a namespace) with line/column/name information of the input text rule triggers?

Such would help compose very useful error/information messages for the lay users about their text files.