



Enabling Grids for E-science

ISSGC'05

XML documents

*Richard Hopkins,
National e-Science Centre, Edinburgh
June 2005*

www.eu-egee.org



Goals –

- General appreciation of XML
- Sufficient detail to understand WSDLs

• Structure –

- Philosophy
 - Detailed XML Format
 - Namespaces
- 

XML = eXtensible Markup Language

- “Markup” means document is an intermixing of
 - Content – the actual information to be conveyed - payload
 - Markup – information about the content - **MetaData**
 - `<date>22/10/1946</date>`
 - `<date> ... </date>` is markup – says that the content is a date
 - Self-describing document
 - **date** is part of a markup vocabulary –
 - a collection of keywords used to identify syntax and semantics of constructs in an XML document

XML = eXtensible Markup Language

- “Extensible” means the markup vocabulary is not fixed
- Compare with similar NON-extensible language
 - HTML (Hypertext Markup Language)
 - Fixed markup vocabulary e.g
 - `<p> This is a paragraph. I like it. </p><p> This is another paragraph </p>`
 - A presentation language for describing how a document should be presented for human consumption –
 - *This is a paragraph. I like it.*
 - *This is **another** paragraph*
 - For HTML the language is fixed and implicit in the fact that this is an HTML document – single-language document
- XML requires explicit definition of the language
- One document can combine multiple languages

```

<businessForms:Invoice>
  <date>
    <USnotations:date>      10/22/2004    </..>
  </..>
  <product>
    <businessForms:barCode>123-768-252    </..>
  </..>
  <quantity>
    <metricMeasures:kilos>  17.53        </..>
  </..>
</..>

```

- **businessForms:Invoice**
 - An Invoice construct within the businessForms language
- **BusinessForms (mythical)**
 - A language defining structure of business documents
 - For business interoperability
 - Doesn't prescribe the language of individual items such as dates
 - Taken from separate languages - USnotations:date
- Language names are actually universally unique URIs – www.DesperatelyTryingToStandardise.org/BusinessForms - see later

<businessForms:Invoice>

<date> <USnotations:date> 10/22/2004 </..> </..>

<product> <businessForms:barCode>123-768-252 </..> </..>

<quantity> <metricMeasures:kilos> 17.53 </..> </..>

</..>

- Separation of concerns – Design Factoring
 - Design of **purchase order structure** and date format etc are independent concerns
 - Re-use of language definitions, e.g. date formats in many languages
 - Extensibility – Purchase order accommodates new product identification schemes (e.g. ISBN for book stores)
- Of course, only works if both ends “understand” all languages used
- Makes things more complex –
 - Creating and identifying the languages

- **Fundamental Standards**
 - **E.g. SOAP - the language for soap messages**
 - **soap-envelope:header** **soap-envelope:body**
 - **A soap message is an XML document and its parts are identified using this vocabulary**
 - **Goal is a factoring that gives pick-and-mix of combinable standards**
 - **Associated with any WS standard will be a Schema definition of its XML language**
-
- **Community conventions**
 - **Perhaps, our BusinessForms language**
- **Specialised Data Structure**
 - **Java configuration tables**
-
- **Specific Application Language**
 - **myProgram:parameter1**
 - **The language used in invoking particular operations of a web service**

How it really looks

```

<businessForms:Invoice>
  <date>
    <USnotations:date>
      10/22/2004
    </USnotations:date>
  </date>
  <product>
    <businessForms:barCode>
      123-768-252
    </businessForms:barCode>
  </product>
  <quantity>
    <metricMeasures:kilos>
      17.53
    </metricMeasures:kilos>
  </quantity>
</ businessForms:Invoice >

```

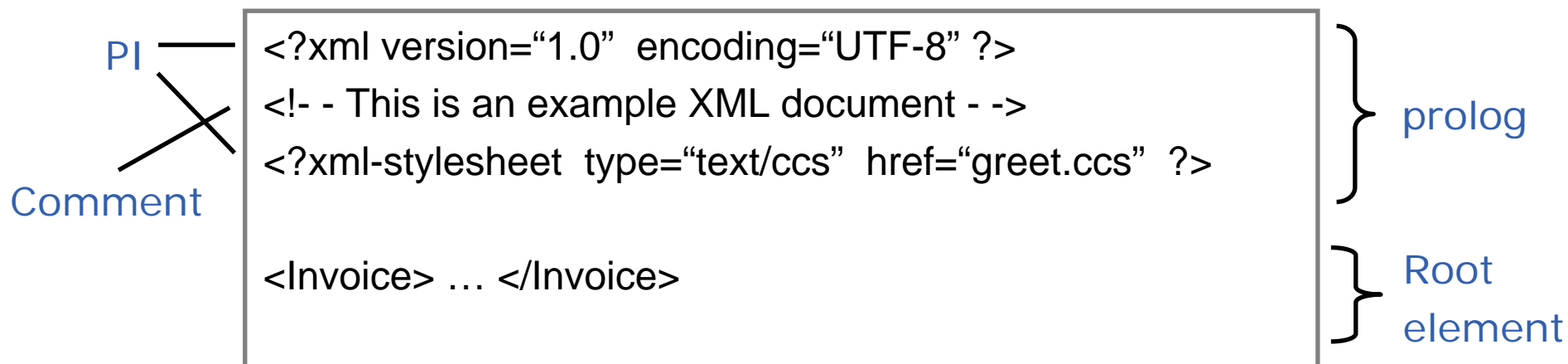
- **Human readable**
 - Sort of - OK with decent editor
 - Is de-buggable
 - Important for meta-data documents,
 - E.g. WSDL
- **Machine processable**
 - Self description enables
 - General tools for producing and consuming XML documents
- **Verbose**
 - OK except for large data
 - Messages may have attachments not in XML

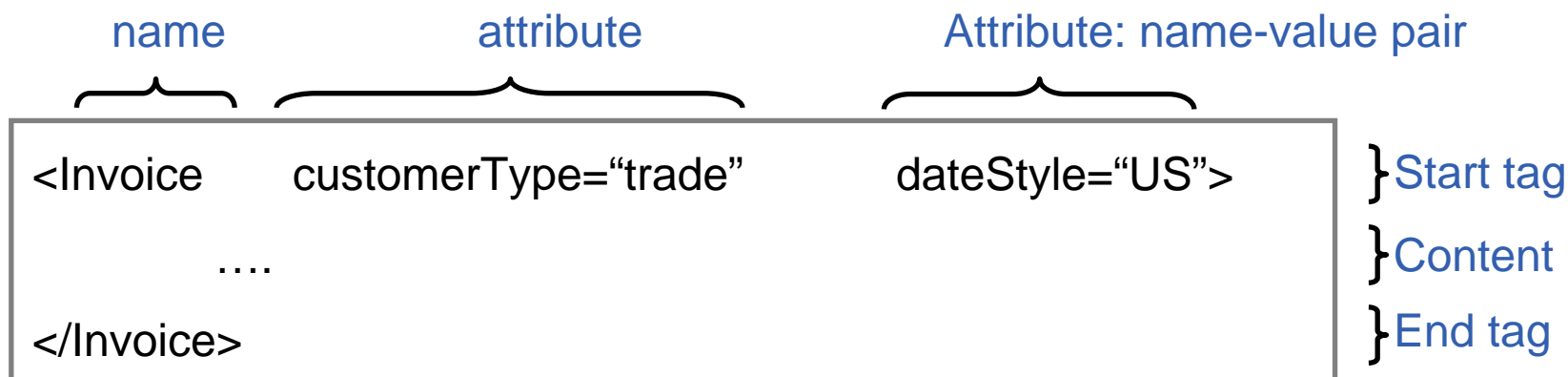
- **Structure**
 - **Philosophy**
 - **Detailed XML Format**
 - **Namespaces**



Main structure of document is

- Prolog – like headers; usually standard and un-interesting
- Element – the actual document – recursively has nested elements
 - Immediately following the prolog
- Miscellaneous –
 - white space and “supplementals” allowed throughout with some restrictions
 - Supplemental – a Comment
a Processing Instruction (PI)



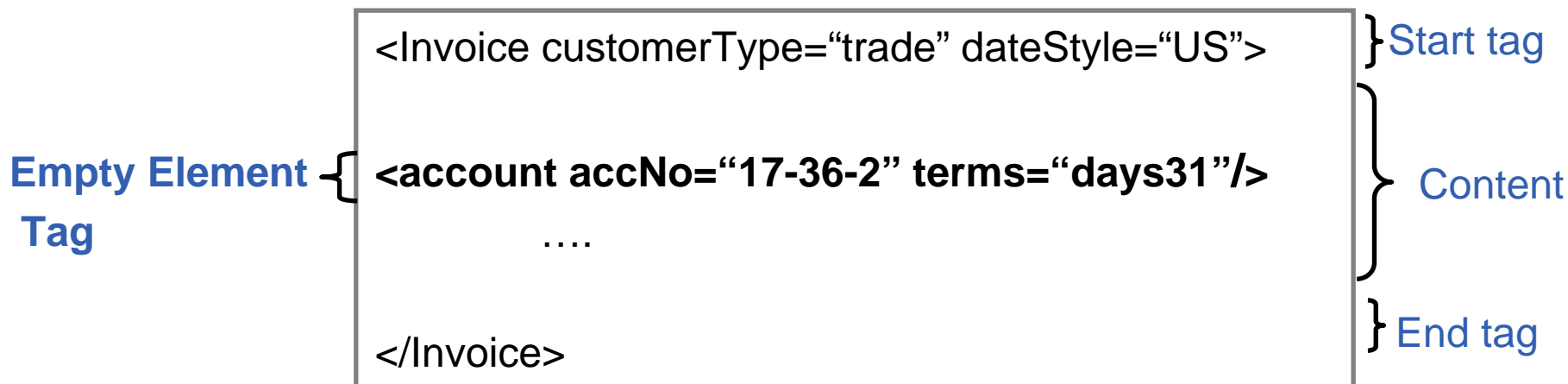


- **Principal element structure –**
 - Start Tag – `<...>`
 - Name of element
 - Zero or more attributes
 - Each a name-value pair
 - Uniquely named;
 - Order insignificant
 - Content – possibly nested elements, and other things
 - End Tag - `</ ... >`
 - Name – MUST be same name as in matching Start Tag
- **Like HTML – but stricter – must have end tag**

```
<Invoice customerType="trade" dateStyle="US"> .... </Invoice>
```

- A name-value pair that is included in the start tag of an element
- Name is part of specific language
- Value may also be part of a specific language – QName – qualified name
- More properly the above might be


```
< BusinessForms:Invoice
  BusinessForms:customerType ="BusinessForms:trade"
  BusinessForms:dateStyle="USnotations:date">
...
</BusinessForms:Invoice>
```
- This starts to get convoluted –
 - necessary for designing for multi-lingual documents



Empty Element Tags –

<account accNo="17-36-2" terms="days31"/>

- Shorthand for element with no content
- just attributes perhaps
- indicated by **/>** not **>**

Same as

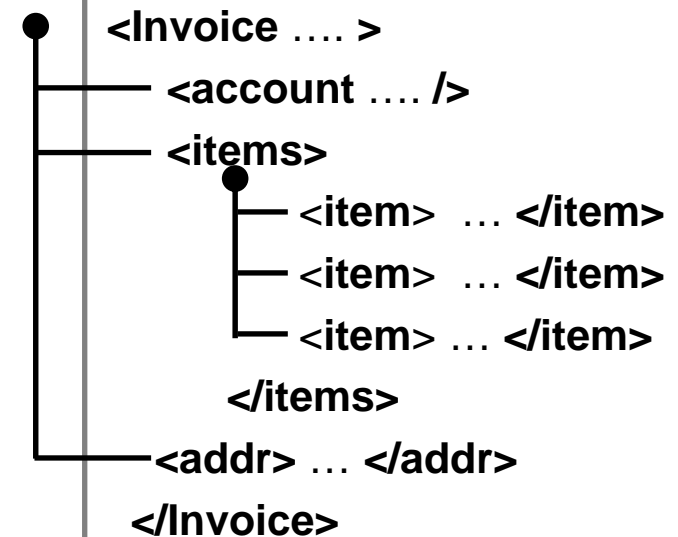
<account accNo="17-36-2" terms="days31">

</account >

```

<Invoice customerType="trade" dateStyle="US">
  <account accNo="17-36-2" terms="days31"/>
  <item> ... </item>
  <item> ... </item>
  <item> ... </item>
  <addr> ... </addr>
</Invoice>

```



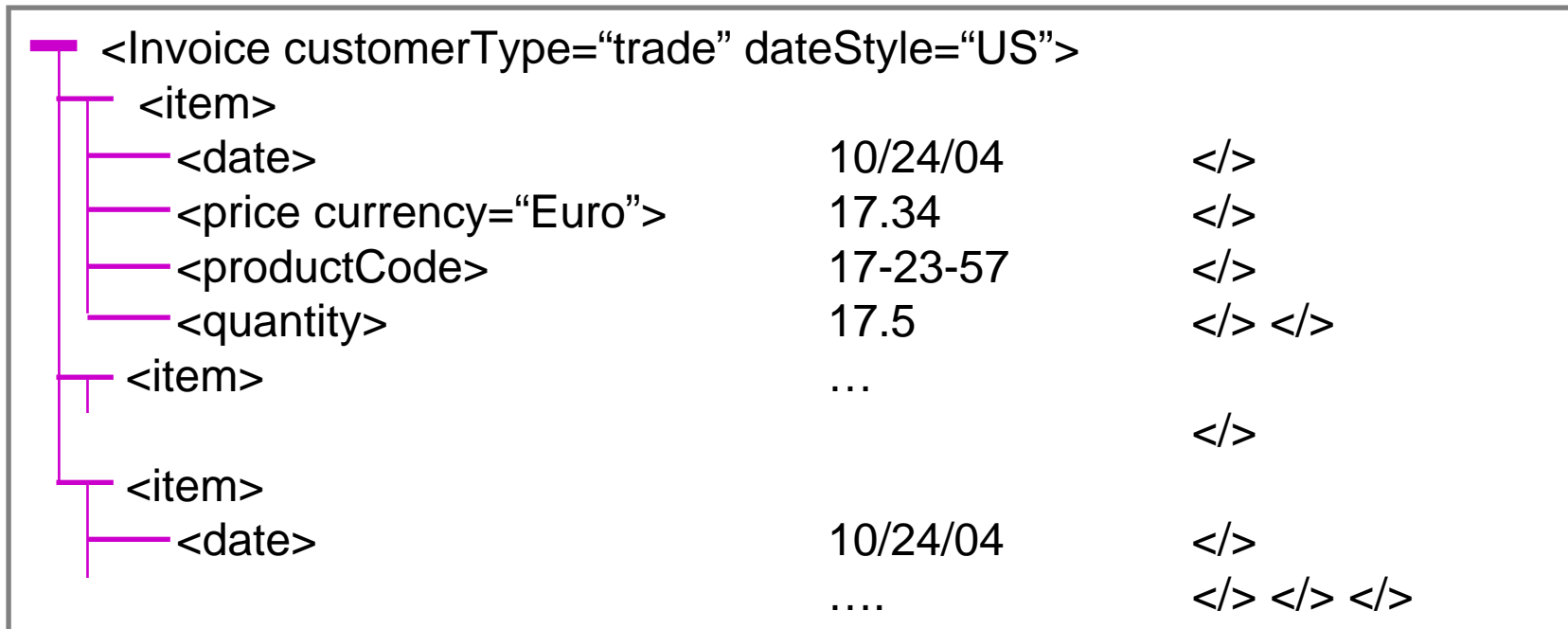
- **Account and Items are child elements –**
 - Non-unique names
 - Usually order is significant
- **This is not a usual programming language configuration**
- **Best if each compound item is either**
 - Structure (Struct) – uniquely named components
 - Array – multiple same-named components

```

<Invoice customerType="trade" dateStyle="US">
  <account accNo="17-36-2" terms="days31"/>
  <item> <date>10/24/04</date> <price> 17.35 </price> </item>
  <item> <date> 10/29/04 </date> <price> 2173.35 </price> </item>
</Invoice>

```

- **<price> 17.35 </price>** is an element with just character data
 - A simple value
- All simple values are text strings, but may have particular syntax and interpretation as decimal, integer, date,



- **Will use XML a lot - Schemas, Soap messages, WSDLs –**
 - so use clearer/briefer notations
- **Textual – direct translation to actual XML**
 - Generally will use indentation to indicate structure**
 - Abbreviate End Tags to just </>**
 - Always have to actually put name in end tag !!!!**
- **Tree diagram – to emphasise structure**

- **Structure**
 - **Philosophy**
 - **Detailed XML Format**
 - **Namespaces**



```

<invoice>                                <!-- INT = International -->
  <deliveryAddress>
    <UK:address> ...<INT:street>...</> ...<UK:county>...</> <UK:postCode>...</></>
  <billingAddress>
    <US:address> ...<INT:street>...</> ...<US:state>...</>   <US:zip>...</> </>
    .... </>

```

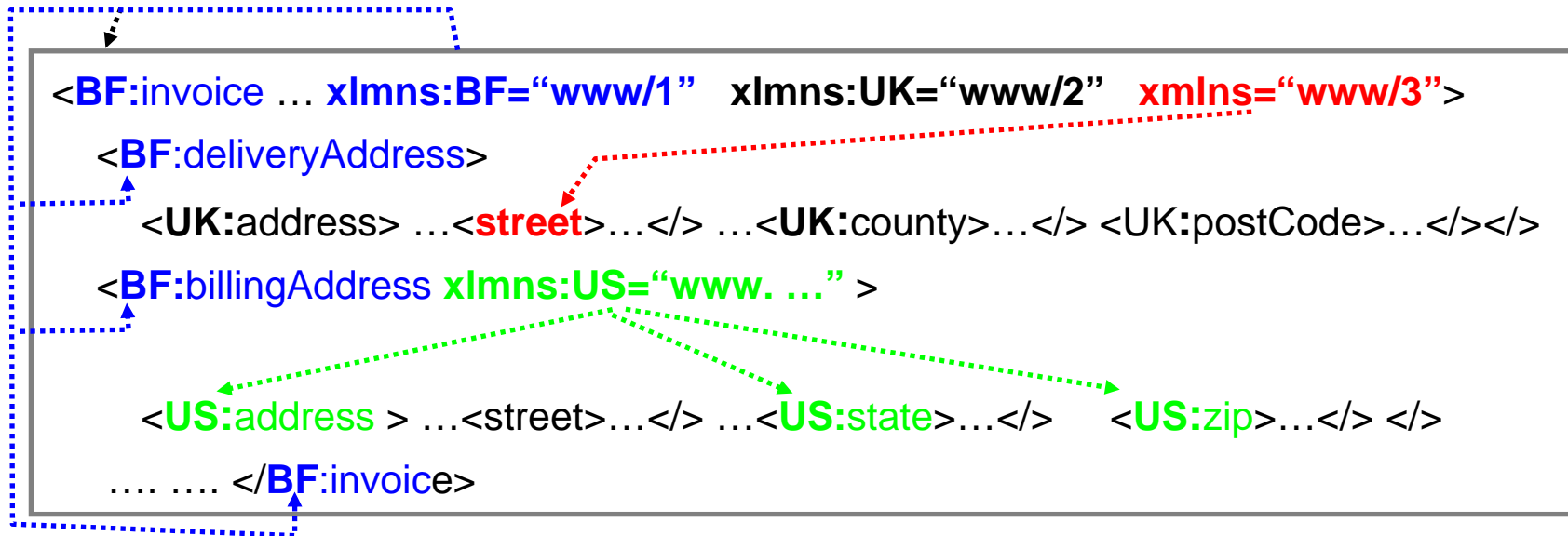
- **A namespace (= “language”)**
 - Defines a collection of names (a vocabulary)
 - For UK : {address, county, postCode, }
 - Usually has an associated syntax (e.g. Schema definition)
 - address = ... county, postCode, ...
 - Syntax may be available to S/W processing it
 - Implies a semantics – the (programmer writing) S/W processing a UK:address knows what it means
 - Provides a unique prefix for disambiguating names from different originators
 - UK vs. US vs. INT

- To get uniqueness of namespace name, use a URI
 - UK:postCode is really
HTTP://www.UKstandards.org/Web/XMLForms:postCode
(mythical)
 - The URI might be a real URL, for accessing the syntax definition, documentation,
 - But it may be just an identifier within the internet domain owned by the namespace owner

UK:postCode is really www.UKstandards.org/Web/XMLForms:postCode

- **But [HTTP://www.UKstandards.org/Web/XMLForms:postCode](http://www.UKstandards.org/Web/XMLForms:postCode) is**
 - Tediously long to use throughout the document
 - Outside XML name syntax
 - Namespaces are not part of XML
 - A supplementary standard <http://www.w3.org/TR/REC-xml-names>
A W3C recommendation

- **In an XML document**
 - declare a namespace prefix, as an attribute of an element
 - `xmlns:UK="HTTP://www.UKstandards.org/Web/XMLForms"`
 - then use that for names in that namespace - UK:postCode
 - **UK:post** code is called a **QName** (qualified name)



- **Namespace declaration occurs as an attribute of an element**
 - i.e. within a start tag
- **Scope is from beginning of that start tag to matching end tag**
 - Excluding scope of nested re-declarations of same prefix
- **Can declare a default namespace**
 - xmlns="www/3" – this is the name space for all un-qualified names in the scope of this declaration, eg. Street
 - But no defaulting for attributes – if no prefix, no namespace

- **Well-formed means it conforms to the XML syntax, e.g.**
 - Start and end tags nest properly with matching names
- **Valid means it conforms to the syntax defined by the namespaces used**
 - Can't check this without a definition of that syntax –
 - Normally a Schema
 - DTD (document Type Definitions) – deprecated
 - Others type definition system
 - – *some more sophisticated than Schemas*

- **THE END**