Transaction Processing Facility

Thinking About XML and SOAP
An introduction for non-programmers

Richard E. Reynolds

A SOAP Message

```
POST /travel HTTP/1.1
Host: www.rickrey.com:1453
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://www.w3.org/2002/06/soap-envelope">
   <SOAP-ENV:Body>
        <l:temp xmlns:l="http://www.rickrey.org/weather">
           <l:date>current</l:date>
           <l:city>Athens</l:city>
        </l:temp>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note: The indentation and the breaking of the message into lines is for our benefit. In reality this message may be transmitted and processed as a continuous stream of data.

The three principal parts of the message

1) This HTTP part is used to route the message from one processor to another. Any system which supports HTTP can process this message.

```
POST /travel HTTP/1.1
Host: www.rickrey.com:1453
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
```

2) The SOAP part contains the protocol information used to route the message to an application in the system that can handle messages of this particular format

Most systems have SOAP handlers.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://www.w3.org/2002/06/soap-envelope">
    <SOAP-ENV:Body>
```

3) This is the application content of the message. An application has to be written to process it, but most systems contain XML handling code to convert the message into a form that the application can more easily use.

```
<l:temp xmlns:l="http://www.rickrey.org/weather">
    <l:date>current</l:date>
    <l:city>Athens</l:city>
</l:temp>
```

```
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

A weather report from Chicago

Weather for Chicago, Illinois, USA

Date: Tuesday, May 17th, 2004

Time: 15:36 GMT

Temperature: 68 F

Sky Condition: Partly Cloudy

Visibility: 10.0 Miles

Barometric Pressure: 29.96 Inches

Winds: 10.0 MPH

Wind Direction: SSW

An XML encoding

```
<?xml version='1.1' encoding='UTF-8' ?>
<weatherReports>
    <report>
        <name>
            <city>Chicago</city>
            <state>Illinois</state>
            <country>USA</country>
        </name>
        <date>2004-05-17</date>
        <time>09:36:00+07:00</time>
        <temperature>20</temperature>
        <skyCondition>Partly Cloudy</skyCondition>
        <visibility>16.0</visibility>
        <windSpeed>16.0</windSpeed>
        <windDirection>SSW</windDirection>
        <barometricPressure>1030.1</barometricPressure>
    </report>
</weatherReports>
```

A SAX Processor - A message view

```
<?xml version='1.1' encoding='UTF-8' ?>
<weatherReports>
    <report>
        <name>
            <city>Chicago</city>
            <state>Illinois</state>
            <country>USA</country>
        </name>
        <date>2004-05-17</date>
        .
        .
        <barometricPressure>1030.1</barometricPressure>
    </report>

</weatherReports>
```

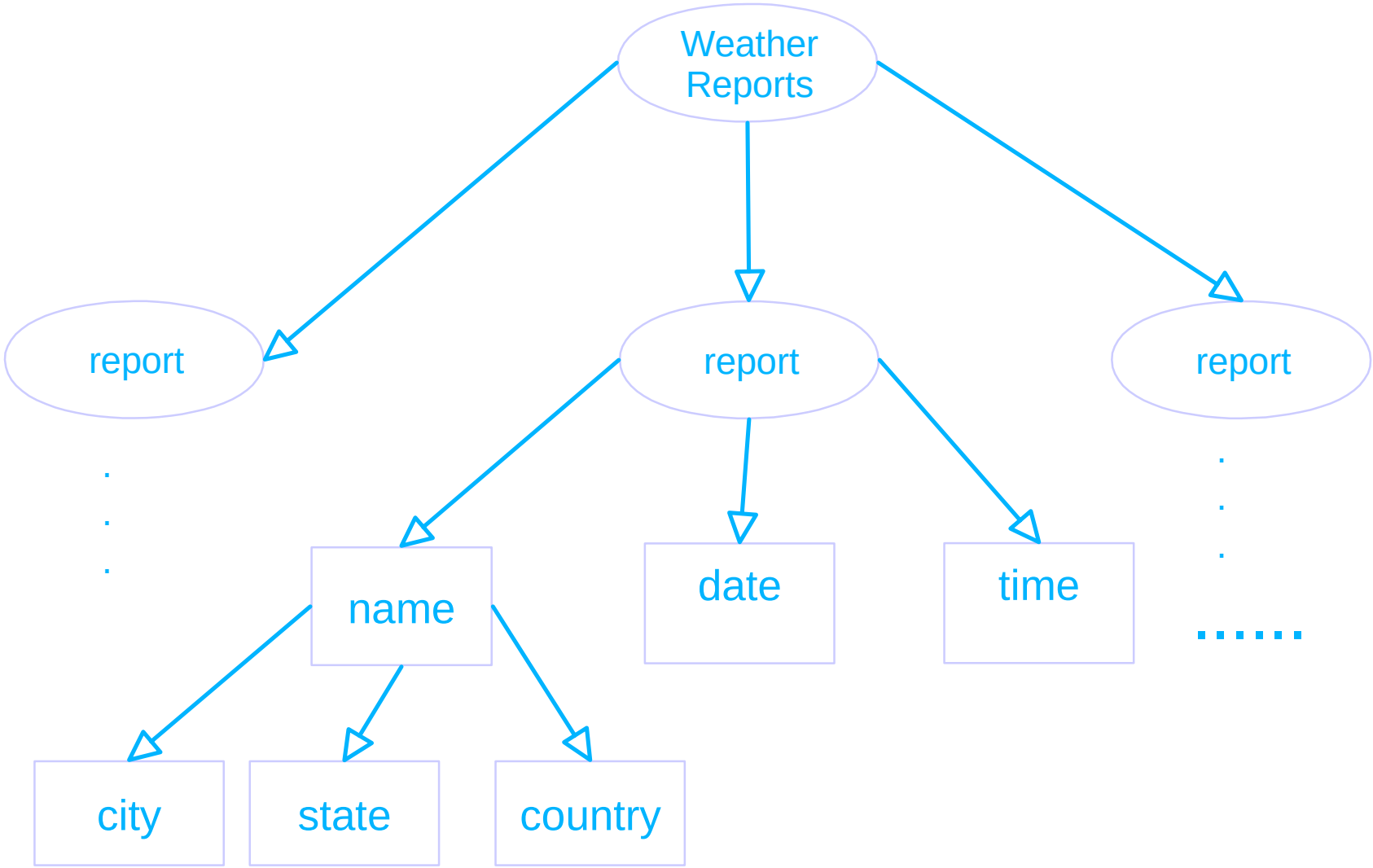
Using the SAX Model, a programmer

- registers routine names to be given control when certain events are reached, such as the beginning or end of elements
- processes the message in a single pass
- must retain the current "state" of the document
- cannot use this model to build a document

An XML encoding for three locations

```
<?xml version='1.1' encoding='UTF-8' ?>
<weatherReports>
    <report>
        ....
    </report
    <report>
        ...
    </report>
    <report>
        ...
    </report>
</weatherReports>
```

A Document View



A DOM model

Using DOM, a programmer:

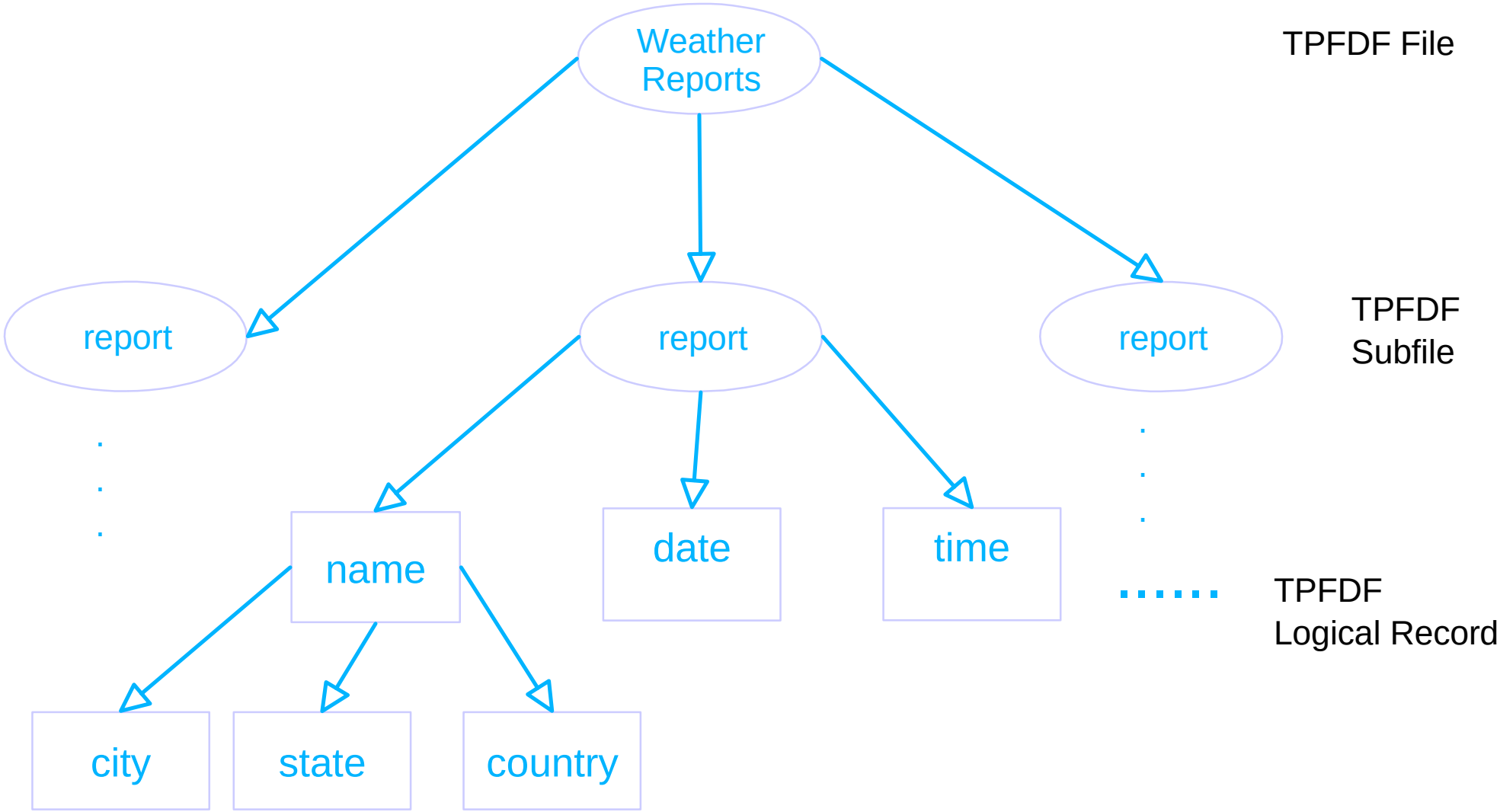
- calls the DOM parser to build the memory structure
- uses DOM API to step through the "nodes" of the document
- can add, delete or change nodes
- can use DOM to build a new document
- expensive to use - API call for each node and for extracting information from the node.

The TPF SOAP model is built for speed

Using TPF SOAP, a programmer:

- accesses the already built infonode structures
- can add, delete or changes nodes
- can use the infonode structures to change an existing document or to build a new document in memory
- a familiar programming interface (structures) for a C programmer

Another look at the document view



My local weather application DSECT

```
   .
WE0NAM   DS      0CL72
WE0CTY   DS      CL24      CITY NAME
WE0STA   DS      CL24      STATE NAME
WE0COU   DS      CL24      COUNTRY NAME
WE0DAT   DS      XL2       PARS DATE
WE0TIM   DS      XL2       MINUTES PAST MIDNIGHT GMT
WE0TMP   DS      H         TEMPERATURE
WE0VIS   DS      H         VISIBILITY
WE0WNS   DS      H         WIND SPEED
WE0BAR   DS      H         BAROMETRIC PRESSURE
WE0DIR   DS      CL3       WIND DIRECTION
WE0SKY   DS      CL25      SKY CONDITION
   .
   .
```

Do you feel more comfortable with this?

My local weather application C structure

```
struct report {
    struct location *name;
    short int parsDay;
    short int minutesPastMidnight;
    short int temperature;
    short int visibility;
    short int windSpeed;
    short int barometricPressure;
    char windDirection[3];
    char skyCondition[25];
}

struct location{
    char city[24];
    char state[24];
    char country[24];
}
```

Still feel more comfortable with this?

Part of my local weather application C++ class

```
class Report {
public:
    Report(...                          //Class Constructor
    short int getWindSpeed();
    void setWindSpeed(short int);
    .
    .
    .
private:
 short int windSpeed;
    .
    .
    .
}
```

Still comfortable?

How should we define our data?

- Programmers should use programming language constructs, regardless of which languages are being used.
  - familiarity and ease of use
  - tools support
- XML provides many advantages across like and unlike platforms
  - platform independence
  - programming language independence
  - relative ease of modifying and versioning
  - schema support
  - tools support

How can we get the benefits of both techniques?

- Use both.
- On your platform, use the best programming language that fits your needs
- To connect between platforms, use all of the capabilities of XML
- Provide a mapping layer between the XML message and the application.

Data Binding is:

- the mapping of data from an external source to a internal program memory structure
- usually associated with XML and Java®, however:
  - can be used for any message or file format
  - can be used with any programming language
- tooling exists for XML or relational data and Java
- principals lend themselves to use by other programming languages
- mapping descriptions are themselves often in XML

A couple of IBM DeveloperWorks references to data binding

- http://www.ibm.com/developerworks/xml/library/x-pracdb1.html
- http://www.ibm.com/developerworks/xml/library/x-bindcastor/

Legal information

Any references to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.