XML in the Development of Component Systems

XML Protocols: XML-RPC
Protocols

Distributed computing
- Components are deployed on different network nodes
- Object implementations do not share memory

Communication Protocol
- Means of exchanging data over a network
- (typically) stream-oriented or message-oriented

RPC: Remote Procedure Protocol
- not supported directly in the network
- (typically) expressed through request/response/error messages
RPC Protocols

- Open Systems Interconnet (OSI) Reference Model (ITU-T Rec. X.200)
  - layered model (physical, data link, network, transport, session, presentation, application layers)
- RPC protocols application specific => defined for application layer
- Representation of request/response message requires specification of presentation layer
- Usage of session layer is optional
RPC protocols are typically defined through interfaces

- informal specification: natural-language description of available operations
- formal specification language: abstract type system to denote operation signatures
- wire-level: detailed specification of messages

Interfaces are offered at access points

- object-oriented RPC: objects offer a service, implementing an interface
RPC Protocol Examples

 OSI Remote Operations (ROSE) (ITU Rec. X.880)
- type system based on ASN.1 (Abstract Syntax Notation) (X.680)
  - interfaces defined through ASN.1 objects
- presentation layer one of multiple encodings (BER: Basic Encoding Rules)
- Used e.g. in GSM (MAP: Mobile Access Part)
- One of multiple session layer protocols (e.g. TCAP: Transaction Capabilities Application Part)

 Open Network Computing (ONC) RPC (IETF RFC 1831)
- interfaces defined in ONC IDL
- presentation layer XDR (External Data Representation, RFC 1832)
- Used e.g. in NFS (Network File System, RFC 3010)
**RPC Protocol Examples (2)**

- **Distributed Computing Environment (DCE) RPC (Open Group C706)**
  - Interfaces defined in DCE IDL (Microsoft IDL)
  - Presentation layer NDR (Network Data Representation)
  - Used e.g. in DCOM (Distributed Component Object Model)

- **Component Object Request Broker Architecture (CORBA) Internet Inter-ORB Protocol (IIOP) (OMG formal/2002-12-06)**
  - Interfaces defined in OMG IDL
  - Presentation layer CDR (Common Data Representation)
  - Used e.g. in J2EE
XML Protocols

General Idea: Use XML as the presentation layer
- in RPC protocols, request and response become XML documents
- often combined with a lower-layer RPC protocol, e.g. HTTP

Rationale
- Interoperable: Specification of presentation layer just needs to define the XML vocabulary
- Easy to implement: Can use existing XML tools

Possible Drawbacks
- lack of interface definition language causes loss of interoperability
- lack of language mappings causes loss of portability
XML-RPC

- Invented by Dave Winer
  - Spec located at http://www.xml-rpc.com/
- Fixed vocabulary for remote procedure calls
  - deliberately very limited type system
- Exchanged over HTTP
- Implementations for various languages
  - multiple implementations for C, C++, C#, Java, Lisp, Perl, PHP, Python, Rebol, Tcl
  - Also available for AppleScript, Delphi, Eiffel, Frontier, Objective C, ...
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: betty.userland.com
Content-Type: text/xml
Content-length: 181

<?xml version="1.0"?>
<methodCall>
    <methodName>examples.getStateName</methodName>
    <params>
        <param> <value><i4>41</i4></value> </param>
    </params>
</methodCall>
Request Requirements

- User-Agent and Host must be specified
- Content-Type is text/xml
- Content-Length must be specified and correct
- Payload is a single `methodCall` element
- `methodCall` must contain `methodName` ([0-9A-Za-z_.:/]+)
- may contain `params` element
  - multiple `param` elements, each containing a single `value` element
Data Types

Scalar types
- four-byte integer: element `<int>` or `<i4>`
- boolean: element `<boolean>`, values 0/1
- double: element `<double>`
- string: element `<string>`
- time stamps: element `<dateTime.iso8601>`, e.g. 19980717T14:08:55
- binary data: element `<base64>`

Structures: Element `<struct>`
- Sequence of `member` elements, each with `name` and `value` element

Arrays: Element `<array>`
- Single `data` element, with a sequence of `value` elements
- No need for homogenous data types in an array
HTTP/1.1 200 OK
Connection: close
Content-Length: 158
Content-Type: text/xml
Date: Fri, 17 Jul 1998 19:55:08 GMT
Server: UserLand Frontier/5.1.2-WinNT

<?xml version="1.0"?>
<methodResponse>
<params>
<param>
<value><string>South Dakota</string></value>
</param>
</params>
</methodResponse>
Response Requirements

- HTTP Status code is always 200 (unless there is a lower-level error)
- Content-Type is text/xml; Content-Length must be present and correct
- Payload is a single `methodResponse` element containing a single `params` element containing a single `param` element containing a single `value` element
  - Alternatively, could contain a `fault` element
    - contains a `value` which contains a `struct` of two elements, named faultCode (type int) and faultString (type string)
Fault Example

HTTP/1.1 200 OK
Connection: close
Content-Length: 426
Content-Type: text/xml
Date: Fri, 17 Jul 1998 19:55:02 GMT
Server: UserLand Frontier/5.1.2-WinNT

<?xml version="1.0"?>
<methodResponse>
  <fault>
    <value>
      <struct>
        <member><name>faultCode</name>
          <value><int>4</int></value>
        </member>
        <member><name>faultString</name>
          <value><string>Too many parameters.</string></value>
        </member>
      </struct>
    </value>
  </fault>
</methodResponse>