

IndySoap Architectural Overview

Presentation Resources

Contents:

1. Conceptual model for Application – Application communication
2. SOAP definitions
3. XML namespaces
4. Sample WSDL
5. Sample SOAP exchange, Interface model
6. Indy Soap architecture
7. Live Object Tracking
8. Assertions
9. DUnit testing logic

IndySoap – charter

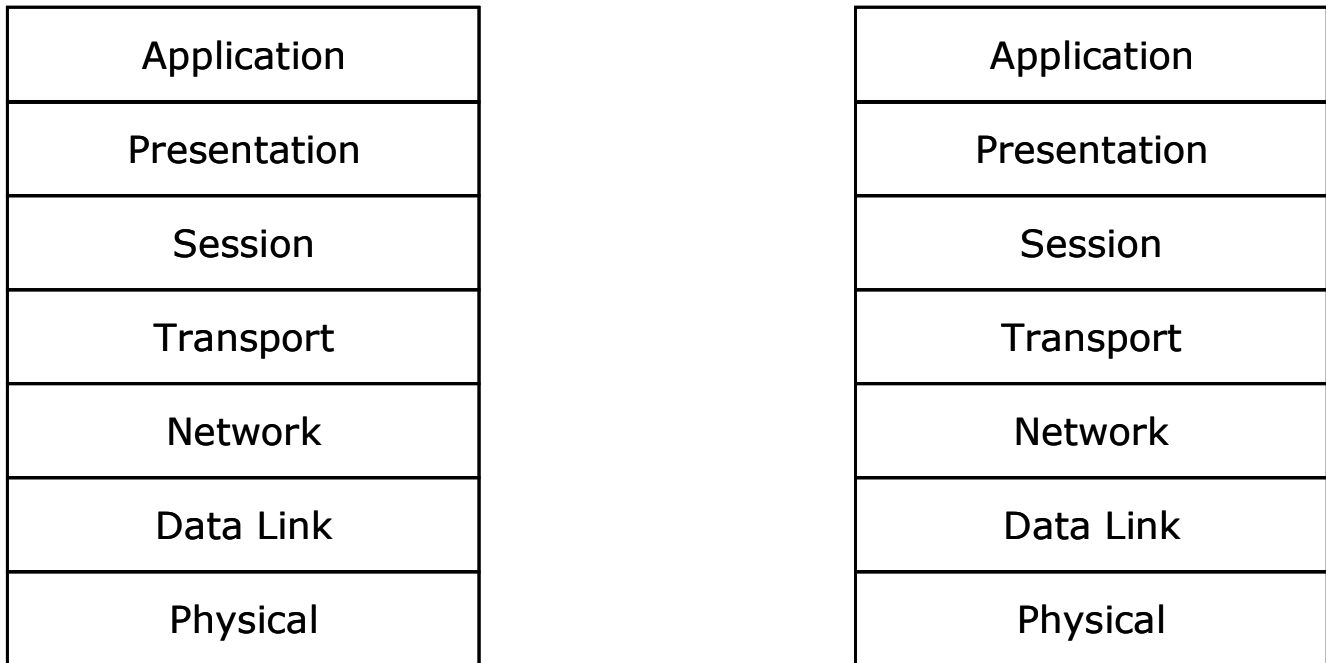
- IndySoap is a Open Source framework for developing web services using Borland Compilers (Delphi, Kylix, CBuilder)
- IndySoap is primarily intended for use as an RPC framework using interfaces as the RPC metaphor. Other uses for Web services and SOAP may be supported at a later date
- IndySoap is intended to inter-operate with other SOAP/Web services implementations
- IndySoap will use Indy for it's transport layers but will also offer transport layers for other network libraries as well

IndySoap – where to find it

Project Home: <http://groups.yahoo.com/group/indy-soap-public/>

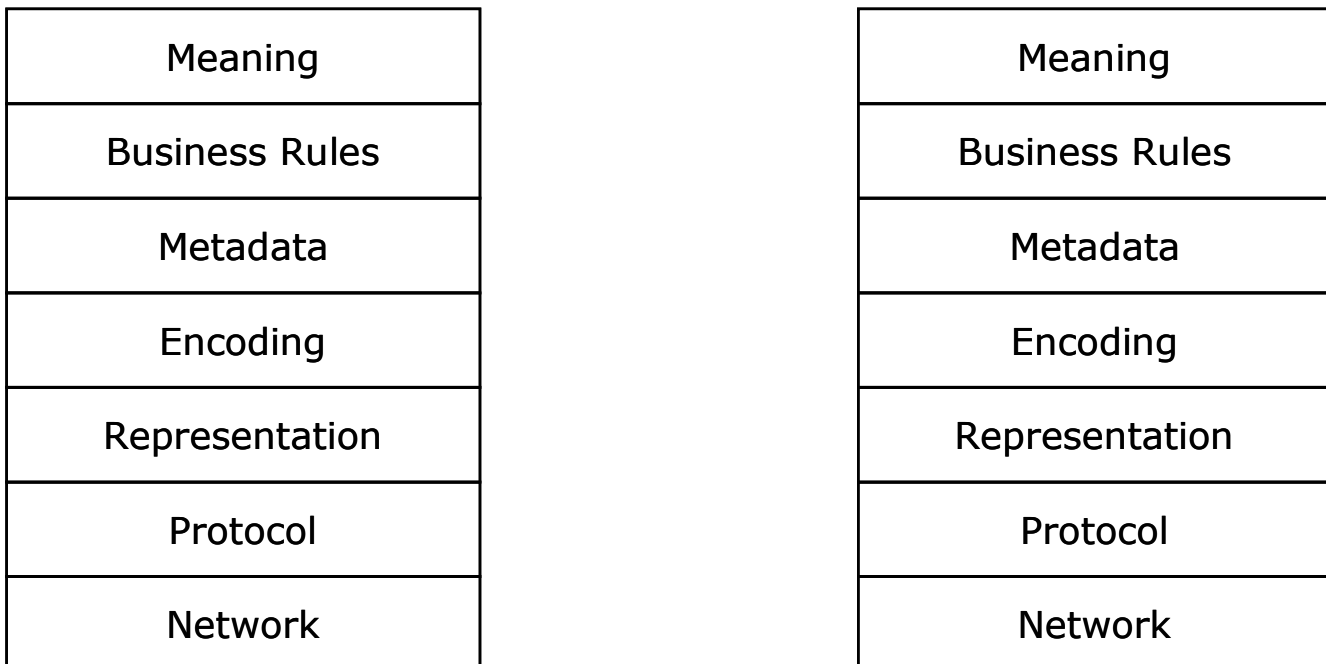
Source Location: <ftp://indy90:indy90@ftp.nevrona.com/soap>

Open System Interconnection Model



Application Development Model

This diagram shows a more useful model for conceptually understanding an application's tasks when interfacing with another application.



“Network” equates to OSI layers 1 – 4, while the other layers are an interpretation of layers 5 - 7

Definitions:

- Normative Official Jargon for a standard that has been all the way through the standard development process and can (*err, might*) be relied on as an unchanging constant
- XML eXtended Markup Language
A very widely supported SGML variant that is useful for supporting structured documents
- Namespace Namespaces are used within an XML document to qualify element and attribute names so that name clashes can be managed
- RPC Remote Procedure Call
A label for any method that can be used to execute code in a different process/computer from the initiator
- SOAP Simple Object Access Protocol
A *not very* simple way to exchange messages between different “objects” in a distributed computing environment using XML based messages.
- WSDL Web Services Description Language
A common format for a structured document describing the way that a particular SOAP service functions

Relevant Standards and Specifications:

- XML <http://www.w3.org/TR/2000/REC-xml-20001006>
A normative standard that lays the groundwork for XML
- Namespaces <http://www.w3.org/TR/1999/REC-xml-names-19990114/>
A normative standard for defining namespaces within XML documents.
- Schema <http://www.w3.org/TR/xmlschema-0/>
<http://www.w3.org/TR/xmlschema-1/>
<http://www.w3.org/TR/xmlschema-2/>
A normative standard for describing types and value restrictions of elements and attributes in XML documents
- SOAP <http://www.w3.org/TR/soap12-part0/>
<http://www.w3.org/TR/soap12-part1/>
<http://www.w3.org/TR/soap12-part2/>
A working draft describing “a lightweight protocol for exchange of information in a decentralized, distributed environment”
- WSDL <http://www.w3.org/TR/wsdl>
A Note from Microsoft/IBM proposing “an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information”

Note the steady deterioration in confidence in the SOAP area.

Soap Message Namespaces

Colour chart

XML	Black
XML namespace	Red
Schema	Yellow
Soap Envelope	Grey
Soap Encoding	Blue
Source Definitions	Purple
Source Service	Teal
Actual Message Content	Green

```

<?xml version="1.0" encoding='UTF-8'?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <NS1:TestParam
      xmlns:NS1="urn:Defn-ITestInterface"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:NS2="urn:Defn">
      <AInt xsi:type="xsd:int">2</AInt>
      <AString xsi:type="xsd:string">AString&lt;something&gt;</AString>
      <AArray xsi:type="SOAP-ENC:Array"
        SOAP-ENC:arrayType="xsd:string[2]">
        <item>ArrayString 0</item>
        <item>ArrayString 1</item>
      </AArray>
      <NS1:AObject href="#1"/>
      <NS2:AObject id="1" xsi:type="NS2:TTestObject">
        <TestInt xsi:type="xsd:int">3</TestInt>
        <TestString xsi:type="xsd:string">String</TestString>
        <NS2:TestObject xsi:NULL="True"/>
      </NS2:AObject>
    </NS1:TestParam>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

WSDL (<http://www.tankebolaget.se/scripts/NumToWords.dll/wsd/I/NumToWords>)

```

<definitions name = "INumToWordsservice" targetNamespace = "http://tempuri.org/"
  xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns = "http://tempuri.org/"
  xmlns:xs = "http://www.w3.org/2001/XMLSchema"
  xmlns:soap = "http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soapenc = "http://schemas.xmlsoap.org/soap/encoding/">
  <message name="NumToWords_SwedishRequest">
    <part name="aNumber" type="xs:int" />
    <part name="bStripSpaces" type="xs:boolean" /> </message>
  <message name="NumToWords_SwedishResponse">
    <part name="return" type="xs:string" /> </message>
  <message name="NumToWords_EnglishRequest">
    <part name="aNumber" type="xs:int" /> </message>
  <message name="NumToWords_EnglishResponse">
    <part name="return" type="xs:string" /> </message>
  <portType name="INumToWords">
    <operation name="NumToWords_Swedish">
      <input message="tns:NumToWords_SwedishRequest" />
      <output message="tns:NumToWords_SwedishResponse" />
    </operation>
    <operation name="NumToWords_English">
      <input message="tns:NumToWords_EnglishRequest" />
      <output message="tns:NumToWords_EnglishResponse" />
    </operation>
  </portType>
  <binding name="INumToWordsbinding" type="tns:INumToWords">
    <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
    <operation name="NumToWords_Swedish">
      <soap:operation soapAction="urn:NumToWordsIntf-INumToWords#NumToWords_Swedish"
        style="rpc" />
      <input>
        <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="urn:NumToWordsIntf-INumToWords" />
      </input>
      <output>
        <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="urn:NumToWordsIntf-INumToWords" />
      </output>
    </operation>
    <operation name="NumToWords_English">
      <soap:operation soapAction="urn:NumToWordsIntf-INumToWords#NumToWords_English"
        style="rpc" />
      <input>
        <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="urn:NumToWordsIntf-INumToWords" />
      </input>
      <output>
        <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="urn:NumToWordsIntf-INumToWords" />
      </output>
    </operation>
  </binding>
  <service name="INumToWordsservice">
    <port name="INumToWordsPort" binding="tns:INumToWordsbinding">
      <soap:address
        location="http://www.tankebolaget.se/scripts/NumToWords.dll/soap/I/NumToWords" />
    </port>
  </service>
</definitions>

```

Soap Message

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ns="urn://nevrona.com/indy/soap/v1/tankebolaget-ITankebolaget">
  <SOAP-ENV:Body>
    <ns:NumToWords_EnglishRequest
      SOAP-
        ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <aNumber xsi:type="xsd:int">20</aNumber>
    </ns:NumToWords_English>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response

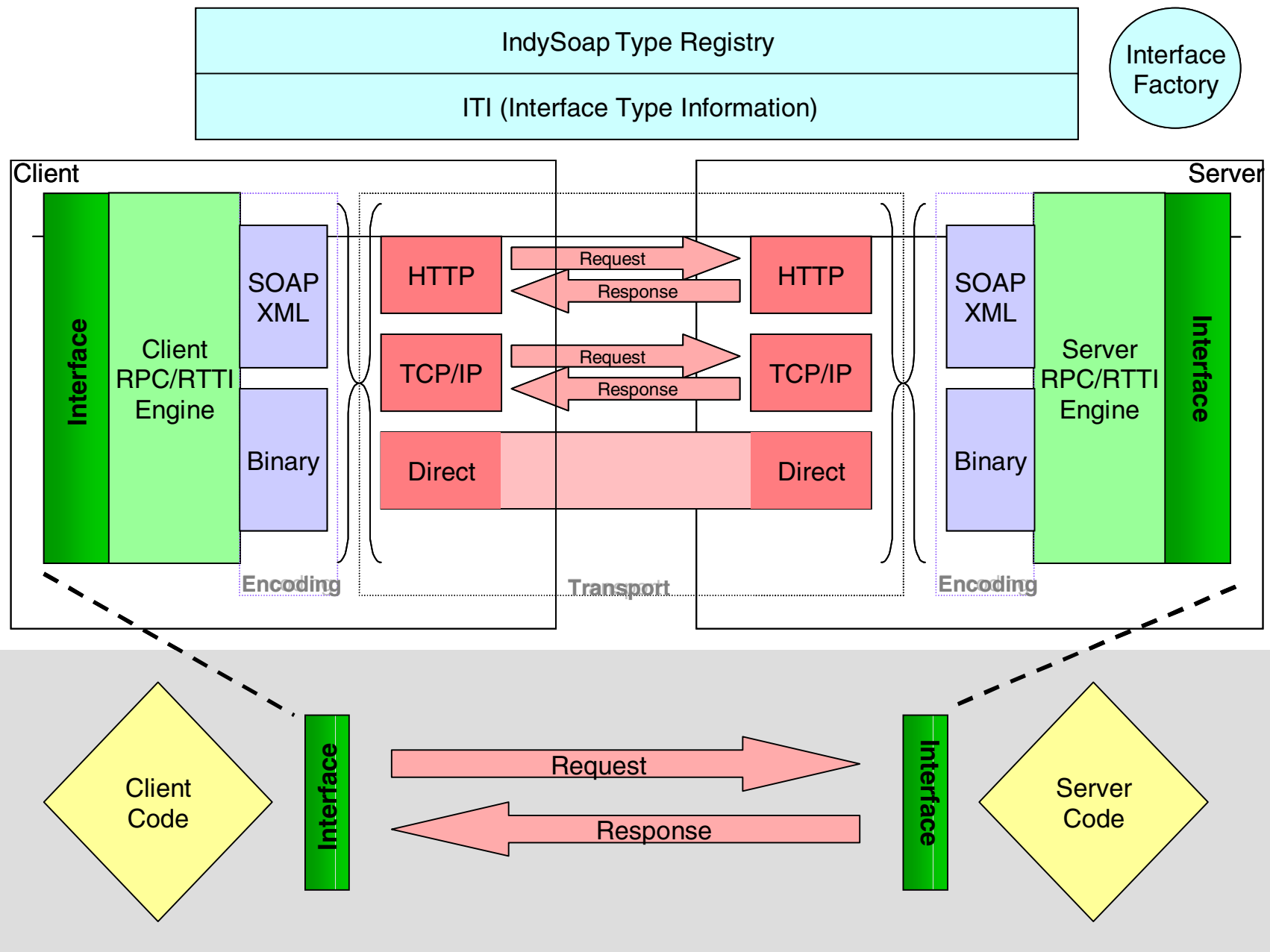
```
<?xml version="1.0" ?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body
    SOAP-
      ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <NS1:NumToWords_EnglishResponse
      xmlns:NS1="urn:NumToWordsIntf-INumToWords">
      <return xsi:type="xsd:string">twenty</return>
    </NS1:NumToWords_EnglishResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Interface

type

```
ITankebolaget = interface (IIdSoapInterface)
  [{D9D7681B-4E3A-4CF2-87C9-9401E1298AEA}]
  function NumToWords_Swedish(aNumber:integer;
    bStripSpaces : boolean):string;
  function NumToWords_English(aNumber:integer):string;
end;
```

IndySoap Architectural Diagram



Live Object Tracking

IndySoap maintains a list of all live objects. This list performs several important services

- Object lifetime tracking
- Object validity checking
- Object leak detection

Base implementation:

type

```
TIdBaseObject = class (TObject)
public
    constructor Create;
    destructor Destroy; override;
    class function GetLiveObjectCount: Cardinal;

    // These routines are only fully functional when OBJECT_TRACKING is
    // defined

    // the client can use the SerialNumber as an independent confirmation that the
    // object is the actual object. This is protection against the object being freed
    // and recreated as a different valid object between usages
    property SerialNumber: Integer;

    // if this is called, then when the object is freed, a debugger breakpoint will be
    // called. This will be raised even if the object is erroneously freed as another
    // object (though not if it is erroneously freed using freemem or similar)
    procedure AskForBreakPointOnFree;

    // what this checks depends on whether FULL_OBJECT_TRACKING is defined
    // If so, checks that the object is valid and in the list of current objects.
    // If not, checks that self <> nil (which is not always very useful but still worth
    // checking)
    function TestValid(AClassType: TClass = NIL): Boolean;
end;
```

Object Leaks:



This summary will appear after the program terminates. It will list all live objects

Assertions:

```

procedure TIdSoapReaderXML.ReadBase( ARootElement: TdomElement;
                                       AsoapNode: TIdSoapNode);
var
  LNode: TdomNode;
  i: Integer;
begin
  assert(Self.TestValid(TIdSoapReaderXML),
    'IdSoapRpcXml.TIdSoapReaderXML.ReadNode: self is not valid');
  assert(ARootElement.TestValid(TdomElement),
    'IdSoapRpcXml.TIdSoapReaderXML.ReadNode: XmlNode is not valid');
  assert(ASoapNode.TestValid(TIdSoapNode),
    'IdSoapRpcXml.TIdSoapReaderXML.ReadNode: SoapNode is not valid');
  for i := 0 to ARootElement.ChildNodes.Length - 1 do
    begin
      LNode := ARootElement.ChildNodes.Item(i);
      assert(LNode.TestValid(TdomNode),
        'IdSoapRpcXml.TIdSoapReaderXML.ReadNode: Base Node '+
        intostr(i)+' is not valid');
      if (LNode is TdomElement) and not
        (LNode as TDomElement).hasAttribute('id') then
        begin
          ReadElement(ARootElement, LNode as TdomElement, ASoapNode);
        end;
    end;
  end;

procedure TIdSoapReaderXML.ReadElement( ARootElement, AXmlElement : TdomElement;
                                          ASoapnode : TIdSoapNode;
                                          ATypeNS : string = "";
                                          AType:String = "";
                                          AName : string = "");
var
  LId : string;
  LElement : TdomElement;
  LType : string;
  LTypeNS : string;
begin
  assert(Self.TestValid(TIdSoapReaderXML),
    'IdSoapRpcXml.TIdSoapReaderXML.ReadElement: self is not valid');
  assert(ARootElement.TestValid(TdomElement),
    'IdSoapRpcXml.TIdSoapReaderXML.ReadElement: RootElement is not valid');
  assert(AXmlElement.TestValid(TdomElement),
    'IdSoapRpcXml.TIdSoapReaderXML.ReadElement: Element is not valid');
  assert(ASoapNode.TestValid(TIdSoapNode),
    'IdSoapRpcXml.TIdSoapReaderXML.ReadElement: SoapNode is not valid');

  // first question, what kind of node is this. There's a little bit of guess work
  // here, and to a degree the only way to really know is to consult the WSDL
  // for the service. This is being avoided at this time.
  if AXmlElement.hasAttribute('href') then
    begin
      // this is a complex type by reference. Lookup the reference, and read it
      LId := AXmlElement.getAttribute('href');
      LElement := GetElementById(ARootElement, LId);
      IdRequire(LElement.TestValid(TdomElement),
        'IdSoapRpcXml.TIdSoapReaderXML.ReadElement: No Element '+
        'with the reference "'+LId+'" could be found (referenced by '+
        AXmlElement.nodeName+'");
      ReadElement(ARootElement, LElement, ASoapNode, ATypeNS, AType, AName);
    end
  else ...

```

DUnit Testing Structure

1. Infrastructure
 - 1.1. Debugging Test Object Tracking functionality
 - 1.2. TidStringList TStringList derivative that owns objects
 - 1.3. TidCriticalSection TCriticalSection derivative that knows whether it is locked or not
 - 1.4. Utilities Various conversion utilities used in IndySoap
 - 1.5. IndyTests Tests for some Indy issues that caused trouble during development
2. Type Registration
 - 2.1. Type Registry
 - 2.2. Exception Registry
 - 2.3. Interface Registry
 - 2.4. Server Interface Registry
 - 2.5. TIdBaseSoapableClass Tests Test object lifetime management issues
3. ITI + Management
 - 3.1. Base ITI Tests Various tests for base ITI behaviour
 - 3.1.1 Parameter
 - 3.1.2 Method
 - 3.1.3 Interface
 - 3.1.4 ITI
 - 3.2. ITI Streaming Check can save and load ITI without loss of information
 - 3.3. ITI Parsing Make sure parser is solid
 - 3.4. ITI Provider
4. Encoding / Decoding
 - 4.1. Self Tests As soon as packet encoding is tested, always test all 3 types of packet encoding
 - 4.1.1 xml8
 - 4.1.2 xml16
 - 4.1.3 Bin
 - 4.2. Compatibility Tests
 - 4.2.1 General General Soap reading issues that have arisen
 - 4.2.2 Standard Check can read various packets from Soap Standard. Packets from Soap standard have to be modified to meet Soap standard?
 - 4.2.3 Borland Check that packet reading layer will read Borland Soap packets OK
5. Communications
 - 5.1. HTTP
 - 5.2. TCP/IP
6. WSDL functionality
 - 6.1. <nothing>
7. Functional Tests
 - 7.1. Interface Tests Tests that test every variation of parameter usage that we can think of
 - 7.1.1 xml8
 - 7.1.2 xml16
 - 7.1.3 bin
 - 7.2. Original Tests The original tests defined before any development started as a development yardstick
 - 7.2.1 HTTP
 - 7.2.1.1 xml8
 - 7.2.1.2 xml16
 - 7.2.1.3 bin
 - 7.2.2 TCPIP
 - 7.2.2.1 xml8
 - 7.2.2.2 xml16
 - 7.2.2.3 bin