unifAce APS 9.2

cbg Workshop

CompuwareBenutzerGruppe

Web Services und XMLReader / -Writer in unifAce 9

Michael Zille
Compuware GmbH

unifAce APS 9.2

cbg Workshop

UXMLREADER & UXMLWRITER

unifAce und XML

Einsatzbereiche

- Im- und Export von Repository-Daten
- Verarbeiten und austauschen von XML streams (z.B. als „disconnected record sets“)

Features

- Generieren und definieren von DTDs
- Lesen und schreiben von XML streams und Dateien
- Validieren und parsen von XML
- Transformation von XML mittels XSLT-Style Sheets

Ansätze für die XML Verarbeitung

- Deklarative XML Verarbeitung
- Prozedurale XML Verarbeitung

unifAce und XML

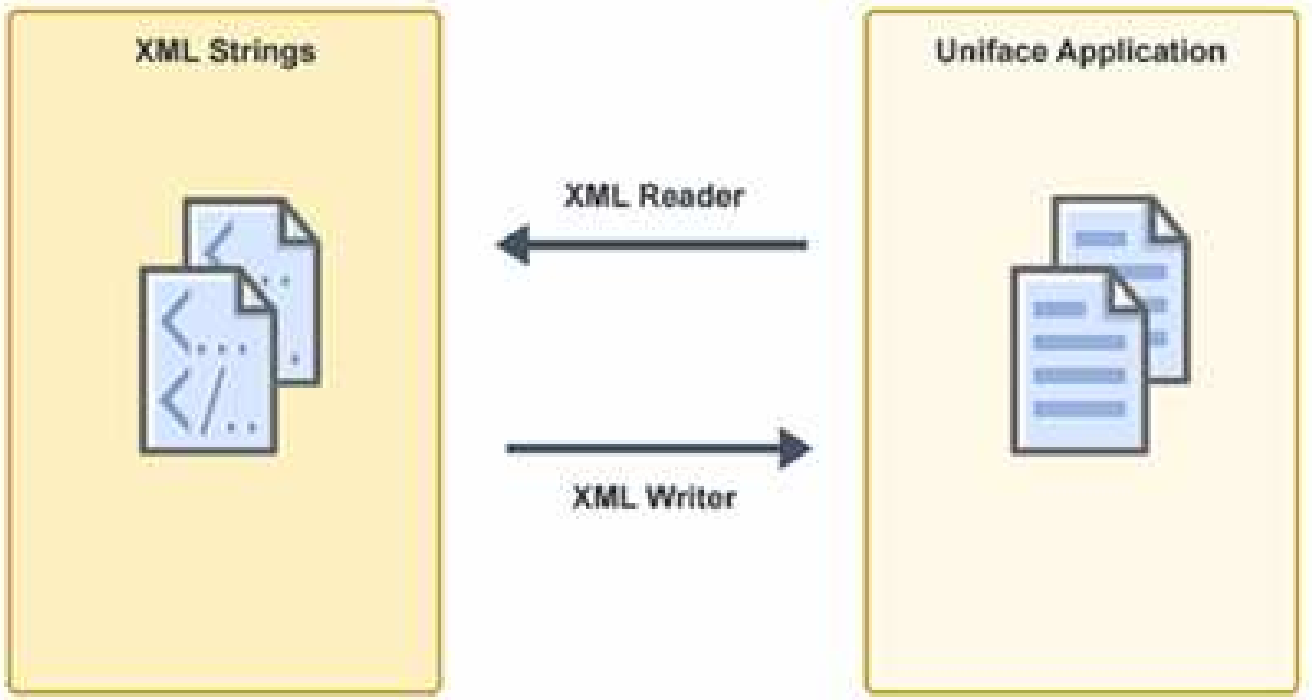
Ansätze für die XML Verarbeitung

- **Deklarative XML Verarbeitung**
  - unifAce-konformes DTD
  - Built-in parsing
  - XML-Datentyp: *xmlstream*
  - Default Mapping: unifAce-Struktur und XML (=>DTD)
  - unifAce-Anweisungen:
    xmlsave, xmlload, xmlvalidate, retrieve/reconnect
  - unifAce-Trigger:
    pre save, post save, pre load, post load occurrence

- **Prozedurale XML Verarbeitung**
  - UXMLWRITER (API Komponente)
  - UXMLREADER (API Komponente)
    - ⇨ SAX Callback Operations

- **XML Transformation (XSLT)**
  - USYSXSLT (API Komponente)
  - XSLT Style Sheet Datei

XML -Reader / -Writer



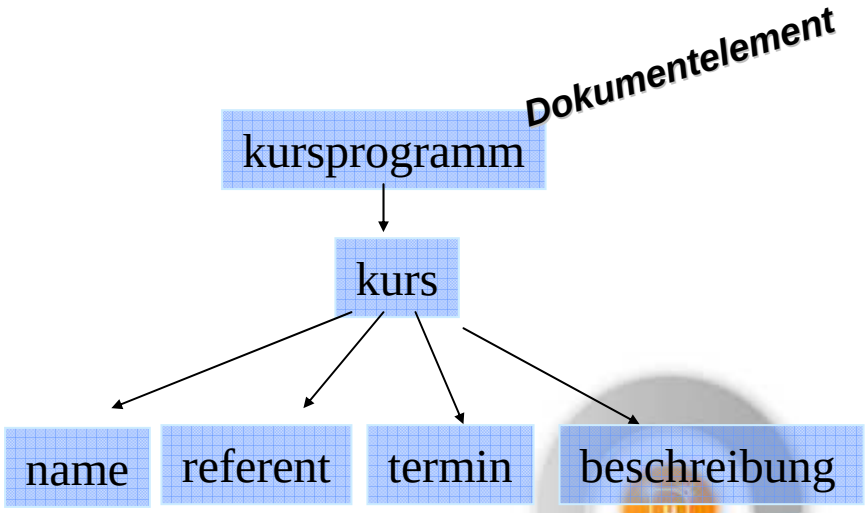
- Ermöglicht das prozedurale „parsen“ und erstellen von XML-Dokumenten (XML strings)
- Vergleichbar mit einem SAX-Parser
- UXMLWRITER dient zur Erstellung eines XML-Dokuments (XML-Stream)
- UXMLREADER dient zum „Parsen“ eines existierenden XML-Dokumentes (XML-Stream oder -Datei)

XML-Elemente

- **werden in einer Baumstruktur dargestellt, die im Dokumentelement ihre Wurzel hat**
- **sind vom Dokumentelement umschlossen**
- **erhalten ihre Bedeutung durch die Tags**
- **Ein Element wird i. d. R. begrenzt durch ein Start-Tag und ein End-Tag**

  <elementtypname> </elementtypname>
- **Ein Element besteht i. d. R. aus Tags und Nutzdaten**

```
<kursprogramm>
   <kurs>
      <name>XML-Grundlagen</name>
      <referent>Hans Fromm</referent>
      <termin>12.11.2006</termin>
      <beschreibung>
          Einführungsseminar für Programmierer
      </beschreibung>
   </kurs>
</kursprogramm>
```



UXMLWRITER

- Ist eine interne Uniface Komponente
- Ermöglicht das Erstellen von XML-Dokumenten (XMLStreams)

UXMLWRITER hierfür stellt folgende Operations bereit:

STARTDOCUMENT
STARTELEMENT
ADDATTRIBUTE
CHARACTERS
ENDELEMENT
ENDDOCUMENT

GETDOCUMENT

DOCCOMMENT

Die Operations müssen in einer feste Reihenfolge aufgerufen werden.

UXMLWRITER
Operations I

Auf der Dokument-Ebene:

➢ *StartDocument (Boolean AddXMLDecl :IN)*

** markiert den Beginn eines Dokumentes (mit / ohne XML Deklaration)*

➢ *EndDocument ()*

** markiert das Ende eines Dokumentes*

Spezielle Operations:

➢ *GetDocument (str XMLDocument :OUT, bool keep :IN)*

** liest den erstellten XML Stream in eine Variable / ein Feld*

** liest nur max. 10240 Characters pro Aufruf*

** gibt im ReturnValue die Anzahl noch nicht ausgelesener Bytes aus*

➢ *DocComment (str Comment :IN)*

** fügt dem XML Stream einen XML Kommentar hinzu*

UXMLWRITER
Operations II

Auf der Element-Ebene:

➢ *StartElement (str Prefix :IN, str LocalName :IN, str URI :IN)*

* *markiert den Beginn eines Elementes des Namens: LocalName*
* *kann den Elementnamen einen Prefix voranstellen*
* *kann eine Namespace-URI mitgeben (max. 512 Bytes)*

➢ *EndElement ()*

* *markiert das Ende eines Elementes*

Unterhalb der Element-Ebene:

➢ *AddAtribute (str Name :IN, str Value :IN, bool Indent :IN)*

* *fügt den Startelement ein Attribut hinzu*
* *kann angeben, ob das Attribut in der folgenden Zeile stehen soll*

➢ *Characters (str Chars :IN)*

* *fügt innerhalb des Elementes einen Text ein*

Arbeit mit UXMLWRITER

Erstellen von XML-Dokumenten:

1. **Erstellen Sie für jedes zu erzeugende Dokument eine Instanz von UXMLWRITER.**

2. **Starten Sie für jede Instanz die Dokumenterstellung**
   Operation: STARTDOCUMENT

3. **Erstellen Sie Elemente mit entsprechendem Inhalt.**
   Operation: STARTELEMENT, CHARACTERS, ENDELEMENT

4. **Ergänzen Sie – je nach Bedarf - Element und Kommentare .**
   Operation: ADDATTRIBUTE, DOCCOMMENT

5. **Lesen Sie das erstellte Dokument in eine Variable/ein Feld aus.**
   Operation: GETDOCUMENT

6. **Löschen Sie die nicht mehr benötigte UXMLWRITER Instanz.**

UXMLWRITER

Demo

Übung

UXMLREADER

- Ist eine interne Uniface Komponente
- Ermöglicht das Parsen von XML-Dokumenten (XMLStreams)
  Die XMLStreams können in Feldern, Variablen oder Dateien vorliegen
- Agiert als Verbindung zwischen den „SAX Parser“ und den „SAX Event Handler“
- Als „SAX Event Handler“ dient eine Uniface Komponente, in der die „SAX Callback Operations“ als OPERATIONS definiert sind.
- In diesen OPERATIONS findet die eigentliche Verarbeitung statt.

Simple API for XML (SAX)

- **Die Simple API for XML (SAX) ist ein Pseudo-Standard,**
- **der ein Application Programming Interface (API) zum Parsen von XML-Daten beschreibt.**
- **Die aktuelle Hauptversion SAX 2.0 wurde 2000 von David Meggionson veröffentlicht und**
- **ist Public Domain.**
- **Ein SAX-Parser liest XML-Daten als sequentiellen Datenstrom und ruft für im Standard definierte Ereignisse vorgegebene Rückruffunktionen (*callback function*) auf.**
- **Eine Anwendung, die SAX nutzt, kann eigene Unterprogramme als Rückruffunktionen registrieren und auf diese Weise die XML-Daten auswerten.**

Simple API for XML (SAX)

<?xml version="1.0"?><doc><para> Hello, world!</para> </doc>

```
<?xml version="1.0"?>
    <doc>
        <para>
            Hello, world!
        </para>
    </doc>
```

```
start document
        start element: doc
                start element: para
                        characters: Hello, world!
                end element: para
        end element: doc
end document
```

UXMLREADER
Operations

➢ *SetHandler (str/Handle Instance : IN)*

** spezifiziert die Komponenten-Instance, die als SAX Handler dient (CallBack-Verarbeitung)*

** muß vor der PARSE / PARSEFILE Operation ausgeführt werden*

➢ *Parse (str XMLDocument : IN , boolean Validation : IN)*

** startet den SAX Parser (built in uniface) und die Callback-Operations*

** gibt die Quelle des XML streams an (Feld oder Variable)*

** max. 10240 Bytes, Encoding muss UTF-8 sein*

** gibt an, ob Validiert werden soll*

➢ *ParseFile (str Filename : IN, boolean : IN)*

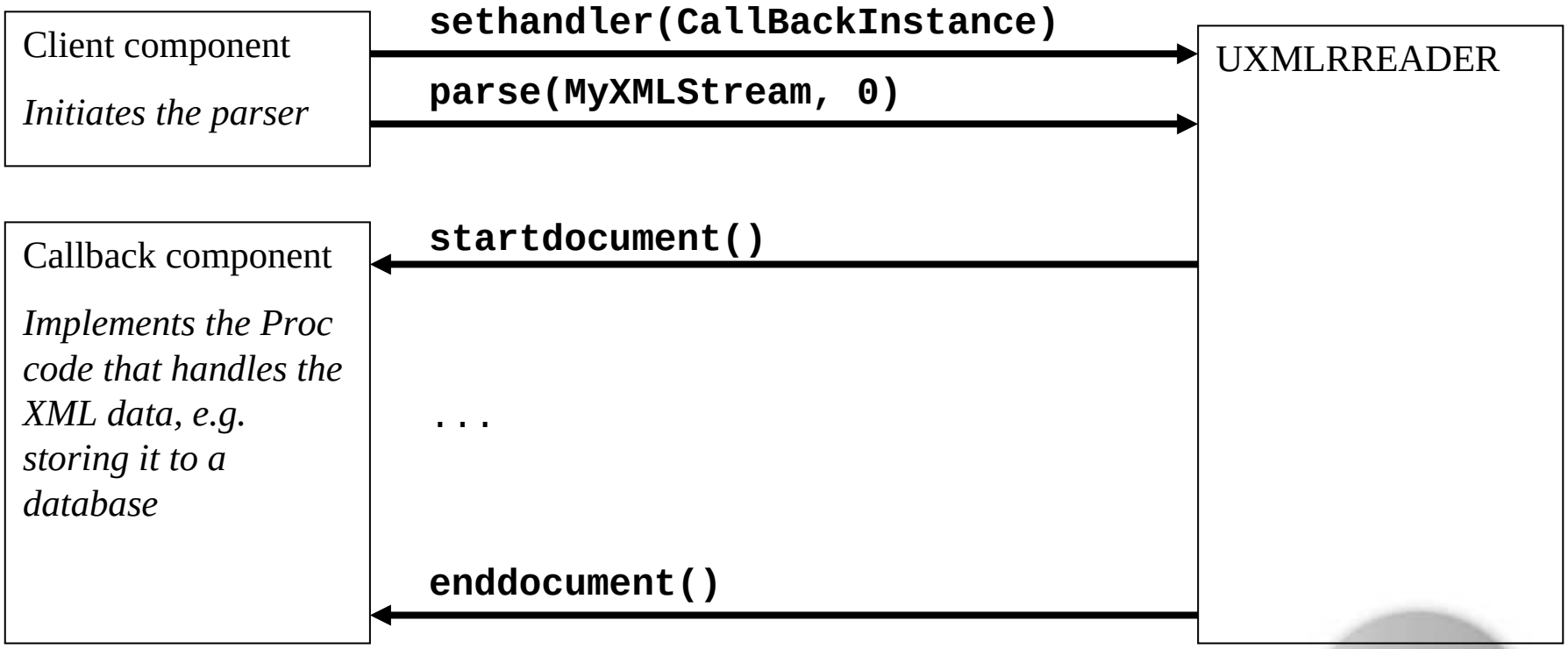
** startet den SAX Parser (built in uniface) und die Callback-Operations*

** bearbeitet die angegebene Datei (Dateiname: max. 512 Bytes)*

** gibt an, ob Validiert werden soll*

UXMLREADER

Call-back Mechanismus



SAX handler Instanz
Callback Operations

- **STARTDOCUMENT**
  aktiviert, wenn der Parser das Lesen eines Dokuments startet.
- **STARTELEMENT**
  aktiviert, wenn der Parser ein Start-Tag findet
  z.B. <doc> oder <para>
- **CHARACTER**
  aktiviert, wenn vom Parser Zeichen (character data) gefunden werden (zwischen start- und end-tag).
- **ENDELEMENT**
  aktiviert, wenn der Parser ein Ende-Tag findet
  z.B. </doc> oder </para>.
- **ENDDOCUMENT**
  aktiviert, wenn das Ende des Dokuments erreicht wird.

Callback Operations

```
operation startdocument
; your code
end
;---------------------->
operation enddocument
; your code
End
```

```
operation startelement
params
    string name: in
    string attr: in
endparams
; your code
end
;---------------------->
operation endelement
params
    string name: in
endparams
; your code
end
```

```
operation warning | error | fatalerror
params
        string list: in
endparams
; your code
end
```

```
operation characters
params
    string chars: in
endparams
; your code
end
```

Arbeit mit UXMLREADER

Erstellen einer "SAX Event Handler"-Komponente

- **Erstellen Sie eine Uniface-Komponente, in der mindestens eine "SAX Callback Operation" implementiert ist.**

  Der Aufruf erfolgt durch die PARSE bzw. PARSEFILE Operations der UXMLREADER Komponente.

Lesen und Parsen eines XML-Dokuments:

1. **Erstellen Sie eine Instanze der UXMLREADER Komponente.**
2. **Erstellen Sie eine Instanz der "Event Handler" Komponente, oder holen Sie die Referenz (Name, Handle) auf eine bereits existierende Komponenteninstanz.**
3. **Machen Sie die "Event Handler" Komponente mit der SETHANDLER Operation von UXMLREADER bekannt.**
4. **Starten Sie die Verarbeitung (parsing) mit der Operation PARSE bzw. PARSEFILE von UXMLREADER.**

UXMLREADER

Demo

Übung

unifAce APS 9.2

cbg Workshop

Einsatz von Web Services

Web Services

Was ist eigentlich ein Web Service?

Gartner Group

„Web services are software technologies making it possible to build bridges between IT systems that otherwise would require extensive development efforts“

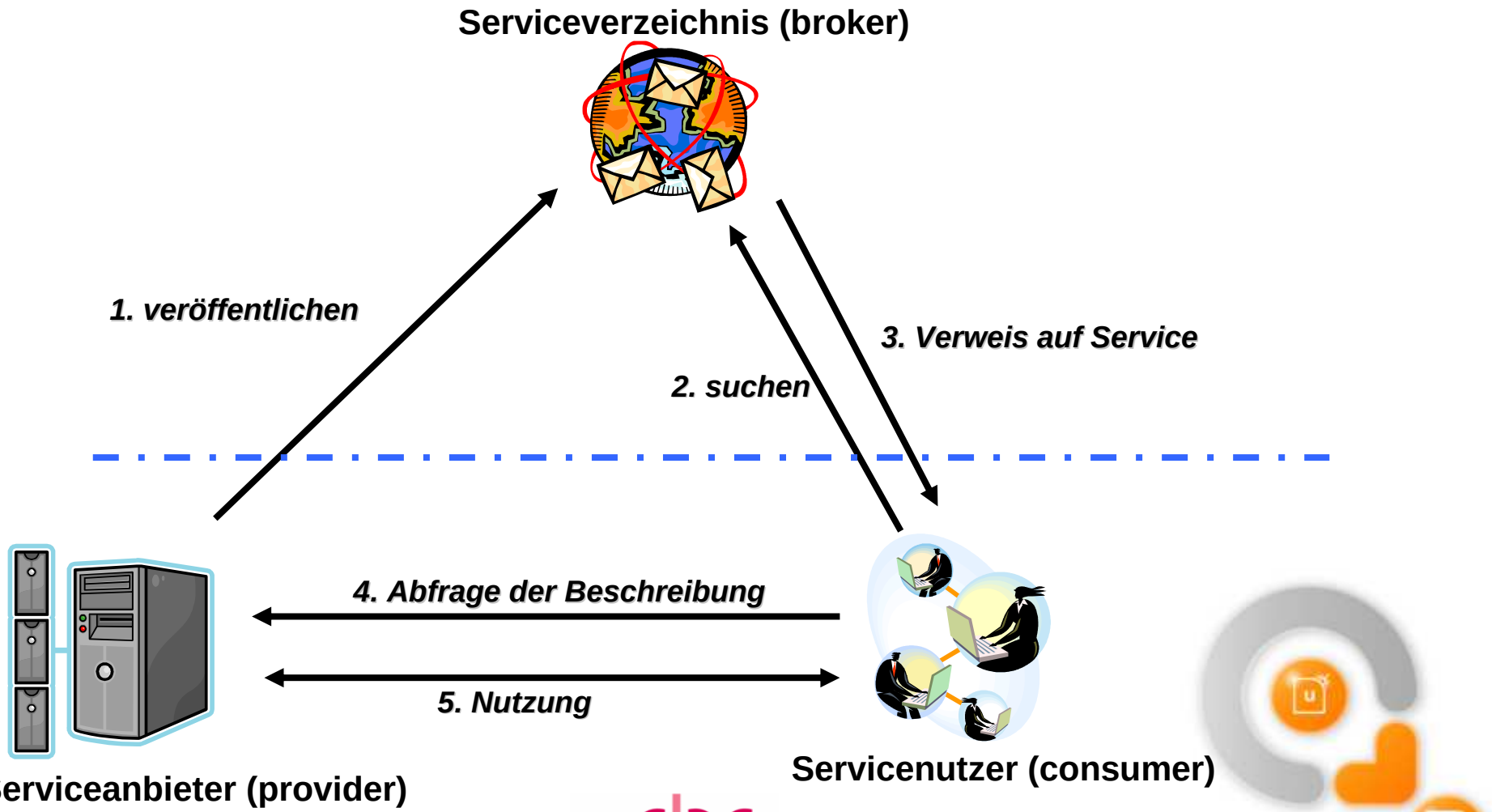
W3C

„A Web service is a software application identified by a URI, whose interface and bindings are capable of being defined, described, and discovered as XML artifacts. A Web service supports direct interaction with other software agents using XML-based Messages exchanged via internet-based protocols. (October 2002)“

„A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by ist description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. (August 2003)“

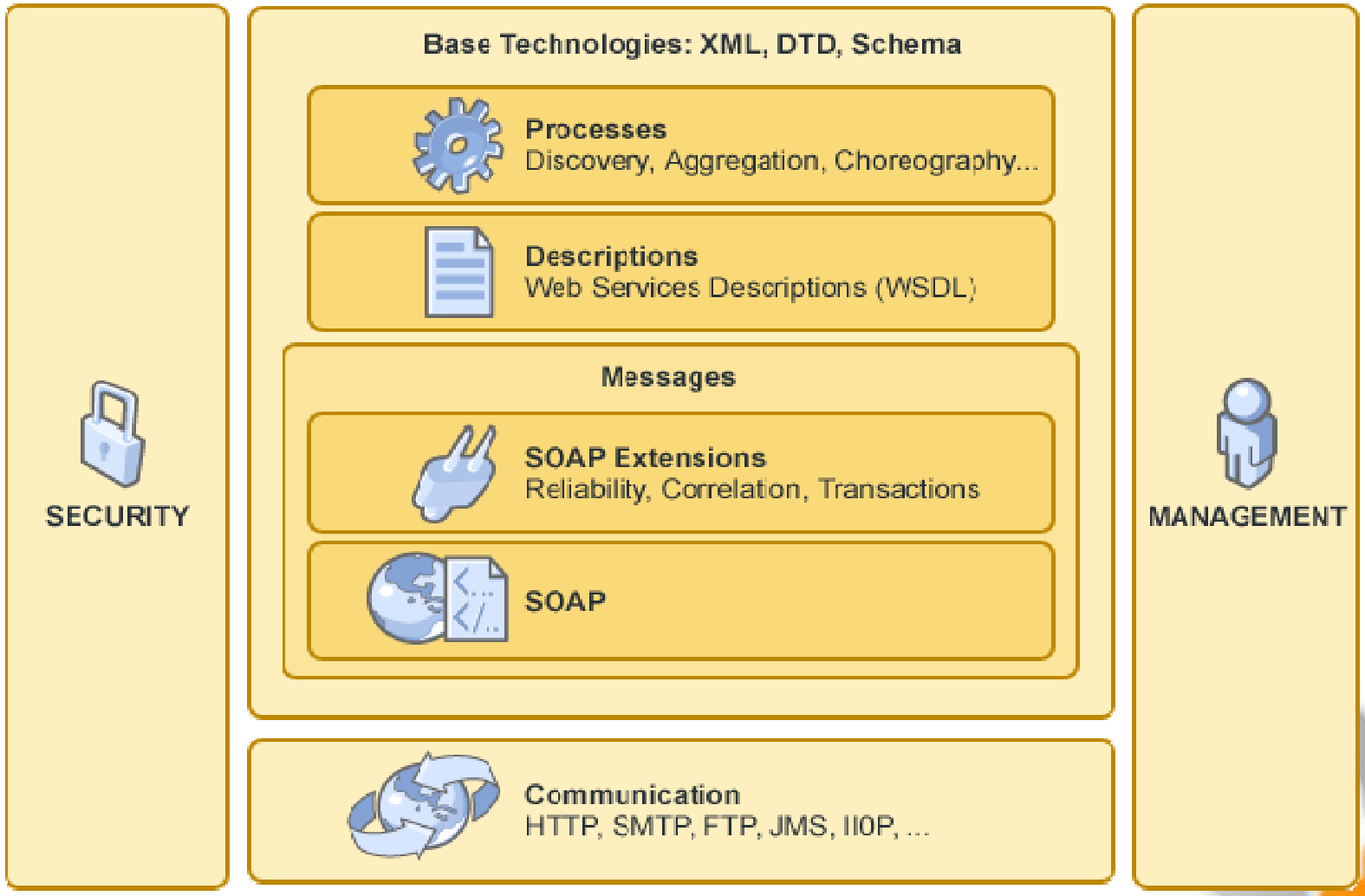
Web Services

Service-orientierte Architektur (SOA)



Web Services

Standards und Spezifikationen



Web Services

Standards und Spezifikationen

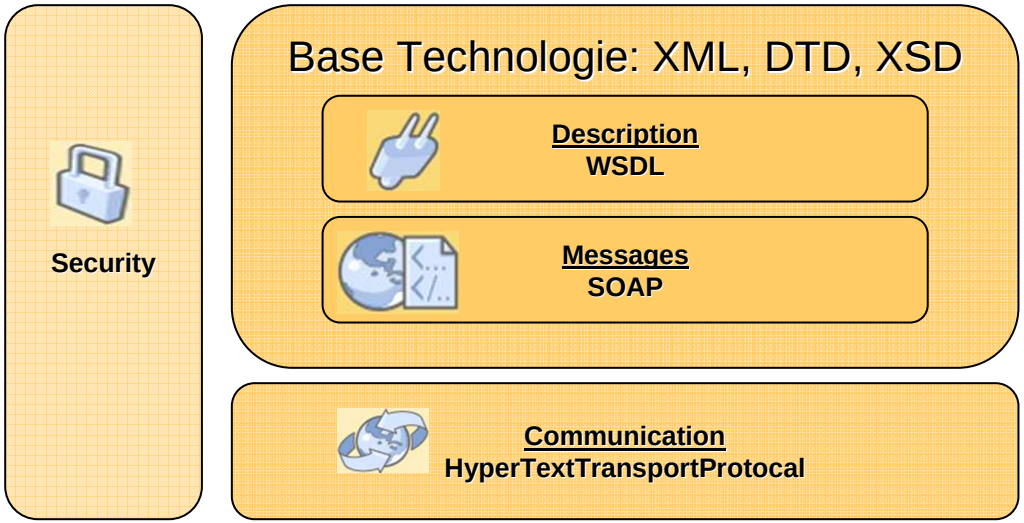
- **SOAP**
  beschreibt das XML-basierte Nachrichtenformat und dessen Einbettung in ein Transportprotokoll
- **WSDL**
  (Web Service Description Language)
  ist eine – ebenfalls XML-basierte – Beschreibungssprache um Web Services zu beschreiben
- **UDDI**
  (Universal Description, Discovery and Integration)
  beschreibt einen Verzeichnisdienst für Web Services.
  UDDI (Universal Description, Discovery and Integration protocol)
- **XSD**
  (XML Schema Definition)
- **HTTP**
  (HyperText Transport Protocol)
- **…**

Web Services

Begriffliche Abgrenzung

Uniface Dokumentation:

- **Software that makes its functionality available over the Internet**
- **Its public interfaces and bindings are defined and described using XML**
- **This definition is discovered by other software systems**
- **Other systems con interact using XML-Based messages conveyed by Internet Protocol**



Web Services mit Uniface

Call In: *Uniface Services als Web Services zur Verfügung stellen*

Call Out: *Web Services in einer Uniface Anwendung verwenden*

Web Services

WSDL und Uniface

- ***Die Web Services Description Language (WSDL) ist eine XML-basierte Sprache. Sie dient der Beschreibung einer Web Service-Schnittstelle und der Art und Weise des Zugriffs auf den Web Service***
- ***Bevor eine Uniface Anwendung einen Web Service aufrufen kann (call-out), muß die entsprechende WSDL-Datei importiert werden.***
- ***Wenn ein Uniface Service als Web Service bereitgestellt werden soll (call-in), muß dessen Schnittstellenbeschreibung in eine WSDL-Datei exportiert werden.***
- ***Ein WSDL-Dokument kann verschiedene „binding styles“ verwenden***
  - ***Der „binding style“ legt das „message format“ und die Art und Weise der Verbindung mit dem Protokoll fest***
  - ***WSDL kennt 4 „binding styles“ von denen einige den Anforderungen des WS-I Basic Profile 1.1 und des Simple SOAP Binding Profile entsprechen.***

WSDL Binding Styles und Uniface

Binding Style	Vereinbar mit: WS-I Basic Profile 1.1 and Simple SOAP Binding Profile 1.0	Unterstützt durch: Uniface SOAP Call-out Connector	Unterstützt beim: Uniface Export (=>WSDL) SOAP Call-in
RPC/Encoded	NEIN	U1.0 (*1)	Mit dem Schalter: /rpcencoded
RPC/literal	JA	U2.0	NEIN
Document/literal Wrapped	JA	U2.0	JA
Document/literal Bare	JA	U2.0	NEIN

(*1) Diese Variante ist aus Gründen der Abwärtskompatibilität verfügbar.
Für neue Anwendungen empfiehlt sich die Verwendung von U2.0

Web Services mit Uniface

Call-Out

Call Out: *Web Services in einer Uniface Anwendung verwenden*

Hierzu muß die Uniface-Anwendung:

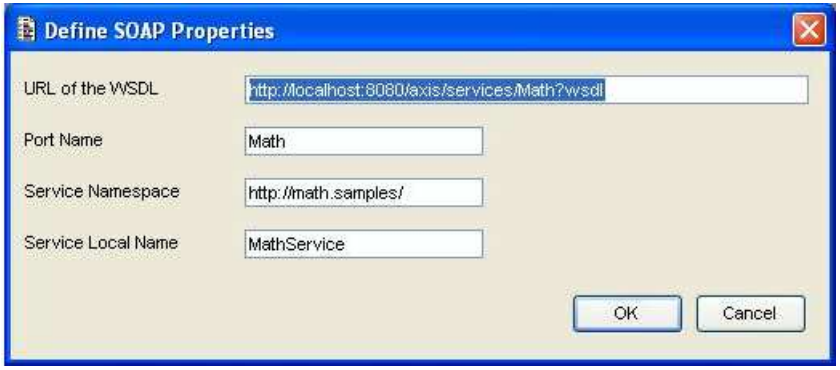
- **die Signature kennen** — WSDL Import
- **mit dem „Format“ umgeben können** — SOAP Connector

Web Services mit Uniface

Call Out – Development (Signature)

Voraussetzung:
SOAP-Connector (SOP U2.0) in der ASN-Datei [DRIVER_SETTINGS] eintragen

1. **Web Service Beschreibung importieren**
   /sti /mwr=ws http://TheWebServicesServerSide/math?WSDL
   => Signature



2. **Signature (bei Bedarf bearbeiten und) kompilieren**
3. **Web Service Funktionalität in der Uniface Komponente aufrufen**
   activate „MATHSERVICE“.add(TheResult,2.5,100)

Web Services mit Uniface

Call Out – WSDL Import

- **Der WSDL-Import wird durch den IDF-Switch /sti durchgeführt**
- **konvertiert die WSDL-Daten in eine Uniface-Signature**
- **Wenn eine korrespondierende XSD-Datei existiert, wird ebenfalls eine Modellstruktur erstellt**

/sti /mwr = Middleware {OptionalMiddlewareSubswitchesAndParameters} Importfile

/sti /mwr=ws {/bare} {/dataonly} {/nosig} {/verbose} {/mod = ModelName} ImportFile

Web Services mit Uniface

Call Out – Development (Signature)

Return Type

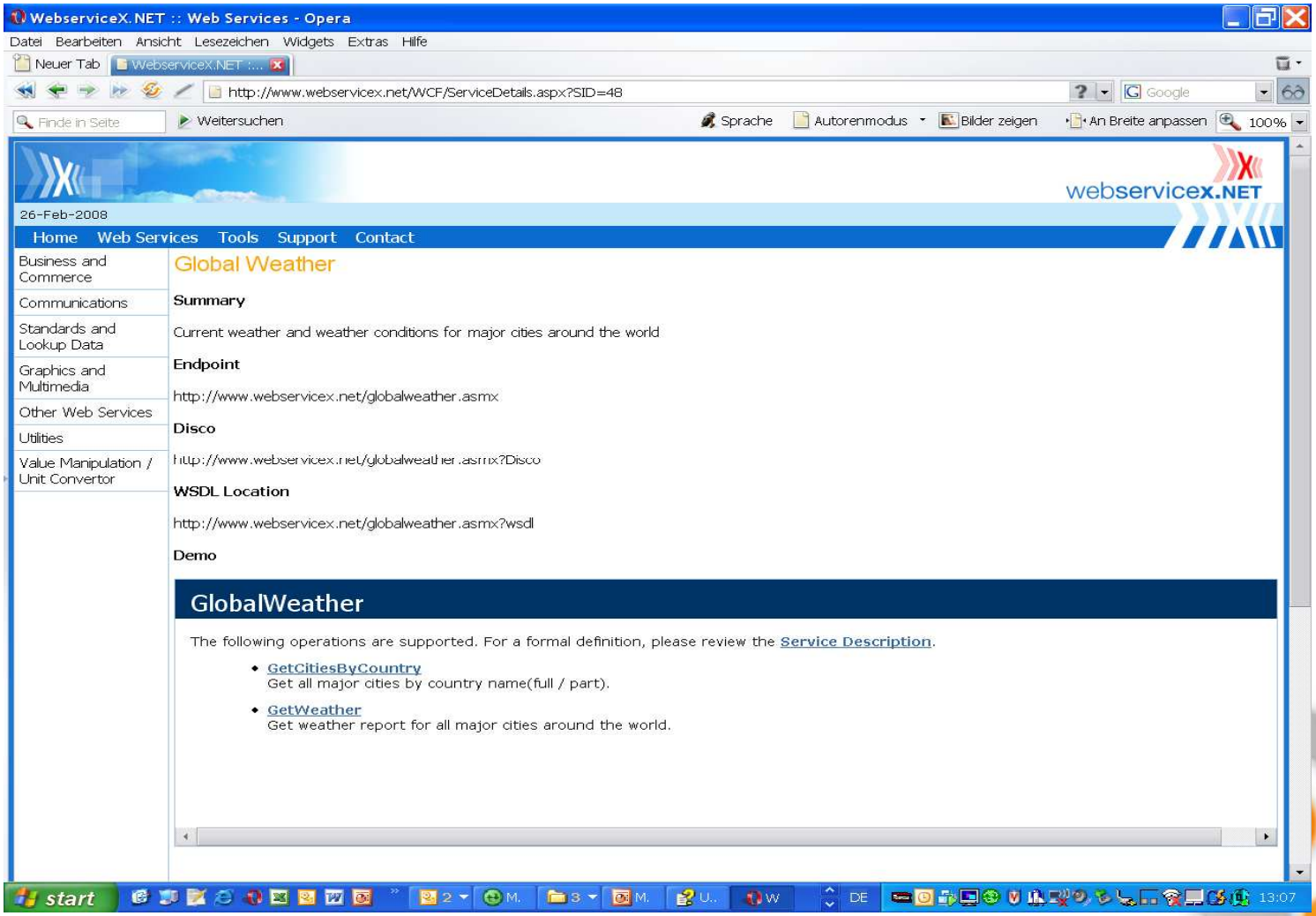
The following defaults apply:

- ***If a return is of data type numeric (XSD data types int, short, or long), it is mapped to $status.***

  This can result in $status containing either a Uniface status or a return value from the Web service, which could be confusing. If you do not like this behavior, you can modify the signature, as follows:
  - ***Add an additional parameter as the first parameter of the operation.***
  - ***In the operation definition, set the Return Value to In First Parameter.***
- ***If a return is of any other data type, it is mapped to the first parameter and Uniface returns a value to $status indicating success or failure.***

Web Services mit Uniface

Beispiel: Wetterinformationen



Web Services mit Uniface

Beispiel: Wetterinformationen

```
- <wsdl:definitions targetNamespace="http://www.webserviceX.NET">
  + <wsdl:types></wsdl:types>
  + <wsdl:message name="GetWeatherSoapIn"></wsdl:message>
  + <wsdl:message name="GetWeatherSoapOut"></wsdl:message>
  + <wsdl:message name="GetCitiesByCountrySoapIn"></wsdl:message>
  + <wsdl:message name="GetCitiesByCountrySoapOut"></wsdl:message>
  + <wsdl:message name="GetWeatherHttpGetIn"></wsdl:message>
  + <wsdl:message name="GetWeatherHttpGetOut"></wsdl:message>
  + <wsdl:message name="GetCitiesByCountryHttpGetIn"></wsdl:message>
  + <wsdl:message name="GetCitiesByCountryHttpGetOut"></wsdl:message>
  + <wsdl:message name="GetWeatherHttpPostIn"></wsdl:message>
  + <wsdl:message name="GetWeatherHttpPostOut"></wsdl:message>
  + <wsdl:message name="GetCitiesByCountryHttpPostIn"></wsdl:message>
  + <wsdl:message name="GetCitiesByCountryHttpPostOut"></wsdl:message>
  + <wsdl:portType name="GlobalWeatherSoap"></wsdl:portType>
  + <wsdl:portType name="GlobalWeatherHttpGet"></wsdl:portType>
  + <wsdl:portType name="GlobalWeatherHttpPost"></wsdl:portType>
  + <wsdl:binding name="GlobalWeatherSoap" type="tns:GlobalWeatherSoap"></wsdl:binding>
  + <wsdl:binding name="GlobalWeatherHttpGet" type="tns:GlobalWeatherHttpGet"></wsdl:binding>
  + <wsdl:binding name="GlobalWeatherHttpPost" type="tns:GlobalWeatherHttpPost"></wsdl:binding>
  + <wsdl:service name="GlobalWeather"></wsdl:service>
  </wsdl:definitions>
```

Web Services mit Uniface

Beispiel: Wetterinformationen

```
- <wsdl:service name="GlobalWeather">
  - <wsdl:port name="GlobalWeatherSoap" binding="tns:GlobalWeatherSoap">
      <soap:address location="http://www.webservicex.net/globalweather.asmx"/>
    </wsdl:port>
  - <wsdl:port name="GlobalWeatherHttpGet" binding="tns:GlobalWeatherHttpGet">
      <http:address location="http://www.webservicex.net/globalweather.asmx"/>
    </wsdl:port>
  - <wsdl:port name="GlobalWeatherHttpPost" binding="tns:GlobalWeatherHttpPost">
      <http:address location="http://www.webservicex.net/globalweather.asmx"/>
    </wsdl:port>
  </wsdl:service>
```

Web Services mit Uniface

Beispiel: Wetterinformationen

```
- <wsdl:binding name="GlobalWeatherSoap" type="tns:GlobalWeatherSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
  - <wsdl:operation name="GetWeather">
      <soap:operation soapAction="http://www.webserviceX.NET/GetWeather" style="document"/>
    - <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
    - <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  - <wsdl:operation name="GetCitiesByCountry">
      <soap:operation soapAction="http://www.webserviceX.NET/GetCitiesByCountry" style="document"/>
    - <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
    - <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
```

Web Services mit Uniface

Beispiel: Wetterinformationen

```
- <wsdl:portType name="GlobalWeatherSoap">
  - <wsdl:operation name="GetWeather">
    - <documentation>
        Get weather report for all major cities around the world.
      </documentation>
      <wsdl:input message="tns:GetWeatherSoapIn"/>
      <wsdl:output message="tns:GetWeatherSoapOut"/>
    </wsdl:operation>
  - <wsdl:operation name="GetCitiesByCountry">
      <documentation>Get all major cities by country name(full / part).</documentation>
      <wsdl:input message="tns:GetCitiesByCountrySoapIn"/>
      <wsdl:output message="tns:GetCitiesByCountrySoapOut"/>
    </wsdl:operation>
  </wsdl:portType>
```

Web Services mit Uniface

Beispiel: Wetterinformationen

```
- <wsdl:message name="GetWeatherSoapIn">
    <wsdl:part name="parameters" element="tns:GetWeather"/>
  </wsdl:message>
- <wsdl:message name="GetWeatherSoapOut">
    <wsdl:part name="parameters" element="tns:GetWeatherResponse"/>
  </wsdl:message>
- <wsdl:message name="GetCitiesByCountrySoapIn">
    <wsdl:part name="parameters" element="tns:GetCitiesByCountry"/>
  </wsdl:message>
- <wsdl:message name="GetCitiesByCountrySoapOut">
    <wsdl:part name="parameters" element="tns:GetCitiesByCountryResponse"/>
  </wsdl:message>
```

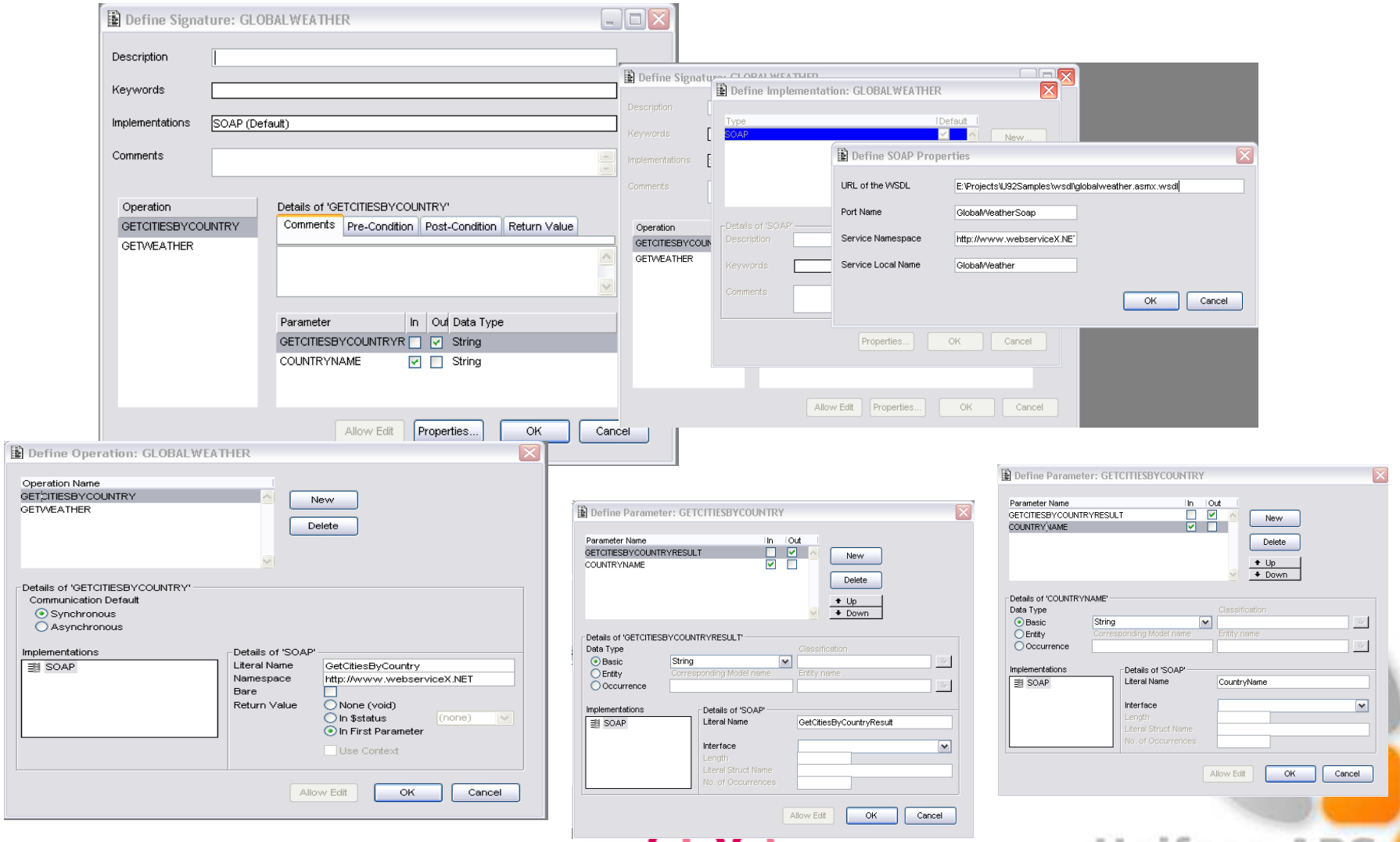
Web Services mit Uniface

Beispiel: Wetterinformationen

```
- <wsdl:types>
  - <s:schema elementFormDefault="qualified" targetNamespace="http://www.webserviceX.NET">
    - <s:element name="GetWeather">
      - <s:complexType>
        - <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="CityName" type="s:string"/>
            <s:element minOccurs="0" maxOccurs="1" name="CountryName" type="s:string"/>
          </s:sequence>
        </s:complexType>
      </s:element>
    - <s:element name="GetWeatherResponse">
      - <s:complexType>
        - <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="GetWeatherResult" type="s:string"/>
          </s:sequence>
        </s:complexType>
      </s:element>
    - <s:element name="GetCitiesByCountry">
      - <s:complexType>
        - <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="CountryName" type="s:string"/>
          </s:sequence>
        </s:complexType>
      </s:element>
    - <s:element name="GetCitiesByCountryResponse">
      - <s:complexType>
        - <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="GetCitiesByCountryResult" type="s:string"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="string" nillable="true" type="s:string"/>
    </s:schema>
  </wsdl:types>
```

Web Services mit Uniface

Beispiel: Wetterinformationen



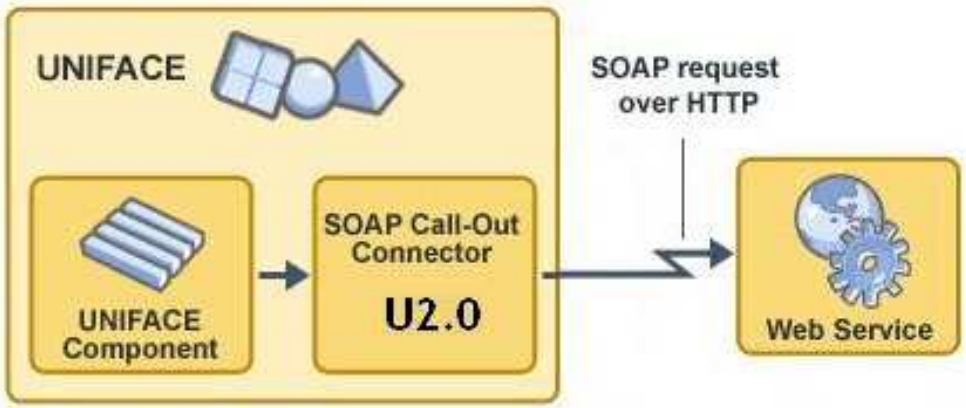
Web Services mit Uniface

Call Out – Deployment (SOAP-Connector)

Voraussetzung:

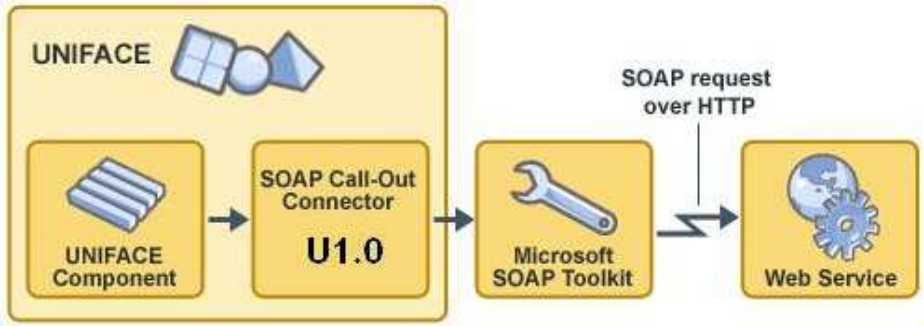
SOAP-Connector (SOP U2.0) in der ASN-Datei [DRIVER_SETTINGS] eintragen

SOP U2.0



SOP U1.0 (veraltet)

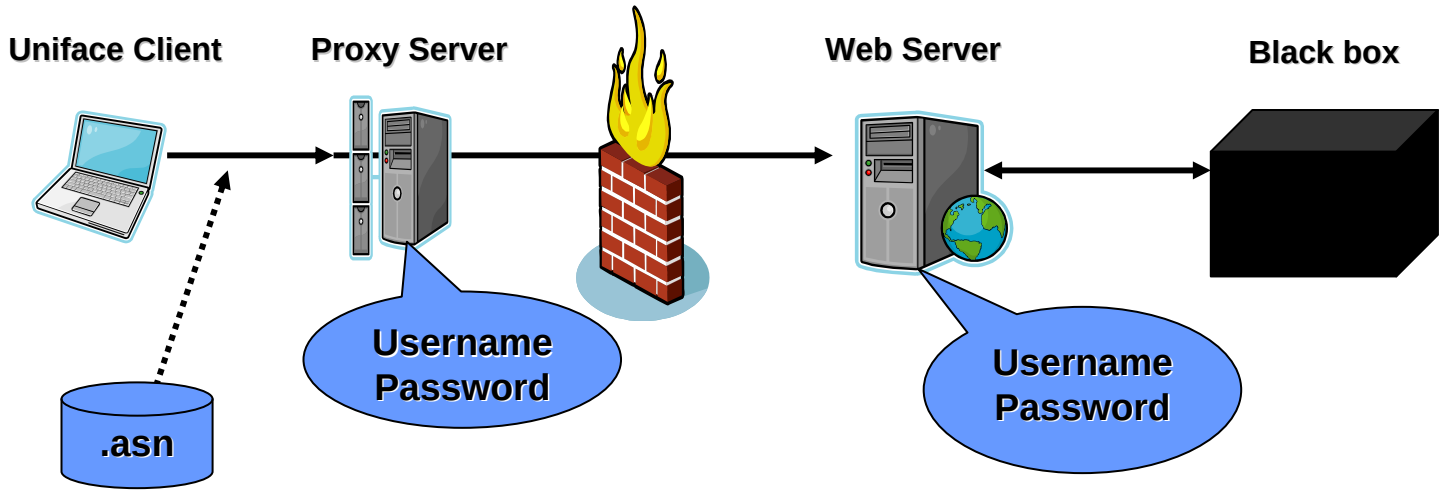
- nur Binding Style 'RPC/Encoded'
- Microsoft SOAP Toolkit nötig



Web Services mit Uniface

Call Out – Deployment (SOAP-Connector)

Authentication



SOAP Connector Options

[DRIVER_SETTINGS]
SOP U2.0
USYS$SOP_PARAMS = puser=user1 ppass=password1 scheme=B

[SERVICES_EXEC]
COMP1 $SOP:COMP1 proxy=proxyhost euser=user2 epass=password2

Web Services mit Uniface

SOAP Connector Optionen

- **Die Optionen können in der Assignment-Datei definiert werden**
  1. **Konnektor-bezogen [DRIVER_SETTINGS]**
  2. **Service-bezogen [SERVICES_EXEC]**

- ***USYS$SOP_PARAMS***
  - ***puser= ProxyServerUserID***
  - ***ppass= ProxyServerPassword***
  - ***euser= WebServiceUserID***
  - ***epass= WebServerPassword***
  - ***proxy= ProxySpec***
  - ***...***

Web Service – Call Out

Demo

Übung

Web Services mit Uniface

Call-In

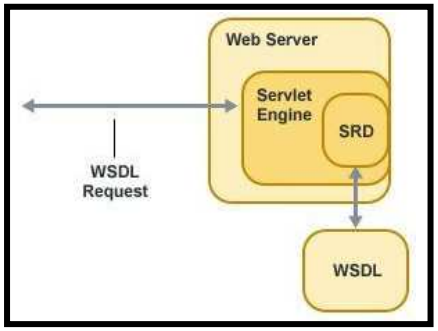
Call In: *Uniface Services als Web Services zur Verfügung stellen*

Hierzu muß die Fremd-Anwendung:

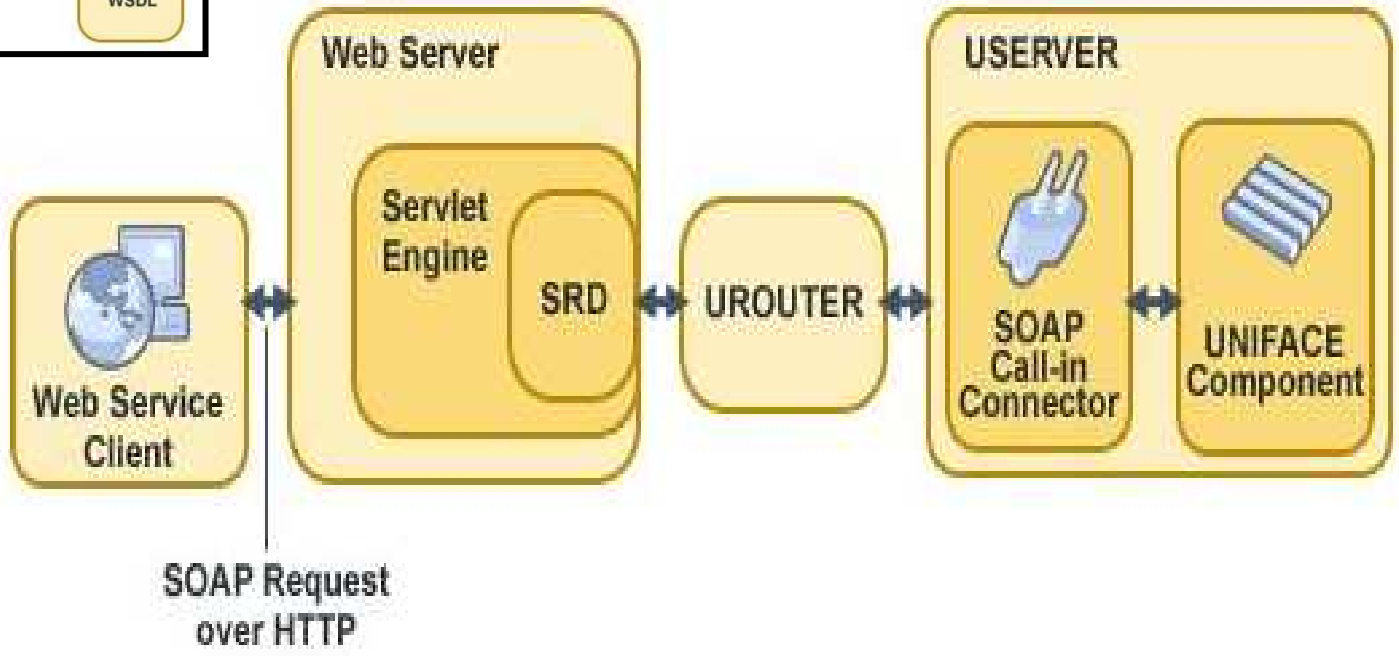
- **die Schnittstelle kennen**
- **der Uniface-Service bereitgestellt sein**

Web Services mit Uniface

Call In – Nutzung des Web Services

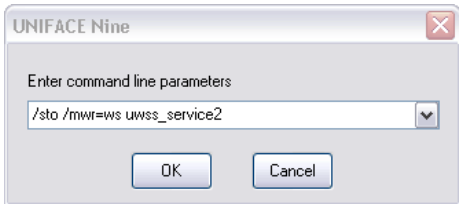


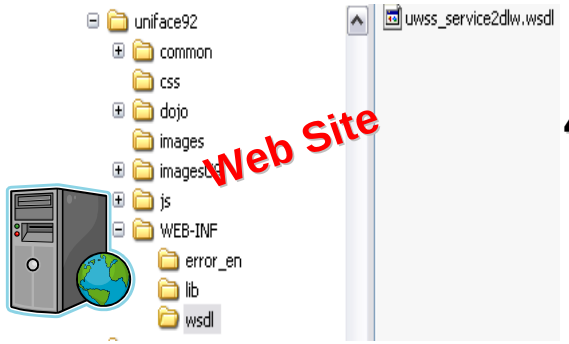
1. *Aufruf der Web Service Beschreibung (WSDL / XSD)*
2. *Aufruf des Web Services*



Web Services mit Uniface

Call In – Entwicklung / Bereitstellung





1. **Erstellen einer Uniface Service Komponente**
2. **Exportieren der Komponenten Signature**
3. **Bereitstellen des Web Services**
   - **Veröffentlichen der WSDL (Web Server)**
   - **Prüfen / ändern der SRD-Konfiguration (Servlet engine)**
   - **Prüfen / ändern der URouter-Konfiguration (UST)**
   - **Prüfen / ändern der Uniface-Laufzeitumgebung**
   - **Starten URouter, Servlet Engine, ...**
4. **Testen des Web Services**

cbg
CompuwareBenutzerGruppe

Compuware Corporation

Uniface APS
The Next Generation

Web Services mit Uniface

Call In – WSDL Export

- **Der WSDL-Export wird durch den IDF-Switch /sto durchgeführt**
- **Erzeugt im Projektverzeichnis eine Datei *signatureNamedlw.wsdl***
- **Bei Bedarf wird außer der WSDL-Datei auch eine XSD-Datei erstellt**
- **Der Binding Style ist „Document/Literal wrapped“**
  (kann per Subswitch auf „RPC/Encoded“ geändert werden)
- **Uniface Datentypen werden auf XML Schema Datentypen abgebildet**

/sto /mwr = Middleware {OptionalMiddlewareSubswitchesAndParameters}

/sto /mwr =ws {/rpcencoded} SignatureList

Web Service – Call In

Demo

Übung

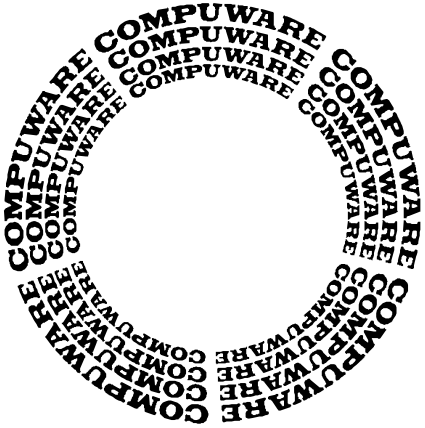
Web Services mit Uniface

Call In – API

- **Mit Call-In API bietet die Möglichkeit ergänzend eigene 3GL-Sicherheitsroutinen zu implementieren.**

- **Die Routinen laufen:**
  - **Vor der Ausführung des SOAP Envelope und**
  - **Direkt nachdem Uniface den SOAP Request verarbeitet hat**

- **API – Funktionen**
  - **soap_pre_request**
  - **soap_post_request**
  - **soap_rel_buf**



COMPUWARE®

People and software for business applications℠

Compuware Corporation

CompuwareBenutzerGruppe

Uniface APS
The Next Generation