IBM eServer zSeries

XML Datenaustausch mit VSE

Ingo Franzki (ifranzki@de.ibm.com) 17.11.2004

Trademarks

The following are Trademarks of the International Business Machines Corporation in the United States and / or other counties.

CICS*
DB2*
DB2 Connect
DB2 Universal Database
e-business logo*
Enterprise Storage Server
HiperSockets
IBM*
IBM logo*
IMS
Intelligent Miner
Multiprise*
MQSeries*
OS/390*
S/390*
SNAP/SHOT*
Virtual Image Facility
VM/ESA*
VSE/ESA
VisualAge*
VTAM*
WebSphere*
xSeries
z/Architecture
z/VM
zSeries

* Registered Trademarks of IBM Corporation

The following are Trademarks or registered Trademarks of other companies.

LINUX is a registered Trademark of Linus Torvalds
Tivoli is a Trademark of Tivoli Systems Inc.
Java and all Java-related Trademarks and Logos are Trademarks of Sun Microsystems, Inc., in the United States and other countries
UNIX is a registered Trademark of The Open Group in the United States and other countries.
Microsoft, Windows and Windows NT are registered Trademarks of Microsoft Corporation.
SET and Secure Electronic Transaction are Trademarks owned by SET Secure Electronic Transaction LLC.
Intel is a registered Trademark of Intel Corporation.

genda

- **XML Grundlagen**
  - XML DTDs
  - XML Schemas
- **Möglichkeiten zu XML Datenübertragung mit VSE**
  - XML Daten erzeugen im VSE
  - XML Daten einlesen/parsen im VSE

ML Grundlagen

XML Eigenschaften:

§ Standardisiert

§ Plain Text (von Menschen lesbar)

§ Strukturiert

§ Selbstbeschreibend (meistens)

ML Grundlagen

Beispiel: VSAM Record:

Lastname	Firstname	Location	ZIP	EmpNo	Salery	…
string	string	string	string	integer	float	….

Einfache XML Version:

```
<?xml version="1.0" encoding="UTF-8"?>
<record>
  <lastname>Mueller</lastname>
  <firstname>Klaus</firstname>
  <location>Munich</location>
  <zip>12345</zip>
  <empno>1234567</empno>
  <salary>1234.56</salary>
</record>
```

ML Grundlagen

Beispiel: VSAM Record:

Lastname	Firstname	Location	ZIP	EmpNo	Salery	…

Strukturierte XML Version, selbe Daten:

```
<?xml version="1.0" encoding="UTF-8"?>
<record>
  <address>
      <lastname>Mueller</lastname>
      <firstname>Klaus</firstname>
      <location zip="12345">Munich</location>
  </address>
  <employee>
      <empno>1234567</empno>
      <salary>1234.56</salary>
  </employee>
</record>
```

Aber:
- Datenstruktur ist unbekannt
- Datentypen sind unbekannt

ML Grundlagen - DTD

Eine DTD im XML File definiert die Datenstruktur:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MYRECORD [
   <!ELEMENT record (address, employee)>
     <!ELEMENT address (lastname, firstname, location)>
      <!ELEMENT lastname (#PCDATA)>
      <!ELEMENT firstname (#PCDATA)>
      <!ELEMENT location (#PCDATA)>
       <!ATTLIST location zip CDATA #REQUIRED>
     <!ELEMENT employee (empno, salary?)>
      <!ELEMENT empno (#PCDATA)>
      <!ELEMENT salary (#PCDATA)>
]>
<record>
  <address>
      <lastname>Mueller</lastname>
      <firstname>Klaus</firstname>
      ...
```

ML Grundlagen - DTD

Die DTD kann in einer externen Datei ausgelagert werden:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MYRECORD SYSTEM "myrecord.dtd">
<record>
  <address>
      <lastname>Mueller</lastname>
      <firstname>Klaus</firstname>
      <location zip="12345">Munich</location>
  </address>
  <employee>
      <empno>1234567</empno>
      <salary>1234.56</salary>
  </employee>
</record>
```

ML Grundlagen - Attribute

Datentypen können über Attribute festgelegt werden:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MYRECORD SYSTEM "myrecord.dtd">
<record>
  <address>
      <lastname type="string">Mueller</lastname>
      <firstname type="string">Klaus</firstname>
      <location type="string" zip="12345">Munich</location>
  </address>
  <employee>
      <empno type="int">1234567</empno>
      <salary type="float">1234.56</salary>
  </employee>
</record>
```

Aber:

- was heist „string“, „int“
- Was bedeutet das „type“ Attribut?

ML Grundlagen - Schemas

Ein Schema definiert einen Namespace verschiedener Datentypen,
m XML File kann man einfach die definierten Typen verwenden:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MYRECORD SYSTEM "myrecord.dtd">
<record
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <address>
      <lastname xsi:type="xsd:string">Mueller</lastname>
      <firstname xsi:type="xsd:string">Klaus</firstname>
      <location xsi:type="xsd:string" zip="12345">Munich</location>
  </address>
  <employee>
      <empno xsi:type="xsd:int">1234567</empno>
      <salary xsi:type="xsd:float">1234.56</salary>
  </employee>
</record>
```

ML Grundlagen - Schemas

- `xmlns:xsd="http://www.w3.org/2001/XMLSchema"`
  - Definiert die eigentlichen Datentypen, z.B. "xsd:string"
- `xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"`
  - Definiert den Namespace für "type", z.B. "xsi:type"
  - Wenn nur "type" geschrieben wird, weiß das Programm nicht das ein Schema type gemeint ist, nur bei exakter Angabe des Schema-Instance Namespaces für das "type" Attribute kann ermittelt werden, das ein Schema type gemeint ist.

ML Grundlagen - Schemas

Schema Definition für STRING:

```
<xs:simpleType name="string" id="string">
 <xs:annotation>
  <xs:appinfo>
   <hfp:hasFacet name="length" />
   <hfp:hasFacet name="minLength" />
   <hfp:hasFacet name="maxLength" />
   <hfp:hasFacet name="pattern" />
   <hfp:hasFacet name="enumeration" />
   <hfp:hasFacet name="whiteSpace" />
   <hfp:hasProperty name="ordered" value="false" />
   <hfp:hasProperty name="bounded" value="false" />
   <hfp:hasProperty name="cardinality" value="countably infinite" />
   <hfp:hasProperty name="numeric" value="false" />
  </xs:appinfo>
  <xs:documentation source="http://www.w3.org/TR/xmlschema-2/#string" />
 </xs:annotation>
 <xs:restriction base="xs:anySimpleType">
  <xs:whiteSpace value="preserve" id="string.preserve" />
 </xs:restriction>
</xs:simpleType>
```

ML Grundlagen - Schemas

- **Dies ist nur der 'Aufhänger' für die Definition, in den Tiefen des Schemas stehen dann die Details, z.B. wie lang/kurz ein Typ sein darf, welche Zeichen er enthalten darf usw.**

- **Ein Programm kann über das Schema den Datentyp ermitteln, dies ist jedoch sehr aufwendig. Daher wird in der Praxis meist eine bestimmte Menge von Namespaces im Programm unterstützt, und das Programm 'weiß' einfach daß**
  - http://www.w3.org/1999/XMLSchema:string
  - http://www.w3.org/2001/XMLSchema:string

  alles Verweise auf den Typ String sind.

```
      xmlns:xsd="http://www.w3.org/2001/XMLSchema“
+     xsd:string
=     http://www.w3.org/2001/XMLSchema:string
```

ML Grundlagen

- **Erst mit einer DTD und dem Schema ist ein XML Dokument „selbstbeschreibend“**
- **Ohne DTD und Schema muss ein Programm „wissen“**
  - welche Struktur das XML Dokument hat
  - was die Tags bedeuten
  - welche Datentypen verwendet werden

ML Daten erzeugen

- **Da XML plain Text ist, kann ein XML Dokument relative einfach erzeugt werden**
  - XML Tags und Daten in einen LIBR Member, POWER Entry oder in eine VSAM Datei schreiben
  - Die Struktur der XML Daten ergibt sich meist aus der Struktur (Copybook) der Daten im VSAM Record, oder durch Vorgaben der empfangenden Seite
  - Nutzdaten müssen in eine textuelle Darstellung konvertiert werden (serialisiert).

ML Daten erzeugen – mit COBOL

```
01     XML-DOC.
  10   XML-TITEL           PIC X(21) VALUE "<?xml version=""1.0""?>".
  10   XML-RECORD-A        PIC X(8)  VALUE "<record>".
  10   XML-ADDRESS-A       PIC X(9)  VALUE "<address>".
  10   XML-LASTNAME-A      PIC X(10) VALUE "<lastname>".
  10   XML-LASTNAME-VAL    PIC X(50).
  10   XML-LASTNAME-A      PIC X(11) VALUE "</lastname>".
  10   XML-FIRSTNAME-A     PIC X(11) VALUE "<firstname>".
  10   XML-FIRSTNAME-VAL   PIC X(50).
  10   XML-FIRSTNAME-A     PIC X(12) VALUE "</firstname>".
  10   XML-LOCATION-A      PIC X(15) VALUE "<location zip=""".
  10   XML_LOCATION-ZIP    PIC X(5).
  10   XML-LOCATION-A2     PIC X(3)  VALUE """>".
  10   XML-LOCATION-VAL    PIC X(50).
  10   XML-LOCATION-A      PIC X(11) VALUE "</location>".
  10   XML-ADDRESS-B       PIC X(10) VALUE "</address>".
  10   XML-EMPLOYEE-A      PIC X(10) VALUE "<employee>".
  10   XML-EMPNO-A         PIC X(7)  VALUE "<empno>".
  10   XML-EMPNO-VAL       PIC X(10).
  10   XML-EMPNO-A         PIC X(8)  VALUE "</empno>".
  10   XML-SALERY-A        PIC X(8)  VALUE "<salery>".
  10   XML-SALERY-VAL      PIC X(10).
  10   XML-SALERY-A        PIC X(9)  VALUE "</salery>".
  10   XML-EMPLOYEE-B      PIC X(11) VALUE "</employee>".
  10   XML-RECORD-B        PIC X(9)  VALUE "</record>".
```

```
<record>
  <address>
      <lastname>Mueller</lastname>
      <firstname>Klaus</firstname>
      <location zip="12345">Munich</locatio
  </address>
  <employee>
      <empno>1234567</empno>
      <salary>1234.56</salary>
  </employee>
</record>
```

```
MOVE "Mueller" TO XML-LASTNAME-VAL.
MOVE "Klaus"   TO XML-FIRSTNAME-VAL.
MOVE "12345"   TO XML-LOCATION-ZIP.
MOVE "Munich"  TO XML-LOCATION-VAL.
MOVE "1234567" TO XML-EMPNO-VAL.
MOVE "1234.56" TO XML-SALERY-VAL.
```

ML Daten erzeugen – Beispiel 1

VSAM ESDS Cluster (variable Record Länge):

```
<?xml version="1.0" encoding="UTF-8"?>
<record><lastname>Mueller</lastname>…</empno><salary>1234.56</salary></record>
<record><lastname>Maier</lastname>…</empno><salary>1234.56</salary></record>
<record><lastname>Schwarz</lastname>…</empno><salary>1234.56</salary></record>
```

Danach kann die ESDS Datei z.B. mit FTP zur Weiterverarbeitung übertragen werden

Problem:

- „händische“ Erzeugung des XML Dokuments
- Struktur ist im Programm hart codiert
- 2 Schritte nötig: Erzeugen + Übertragen der Datei

ML Daten erzeugen – Beispiel 2

VSAM ESDS Cluster + VSAM Redirektor

```
<?xml version="1.0" encoding="UTF-8"?>
<record><lastname>Mueller</lastname>…</empno><salary>1234.56</salary></record>
<record><lastname>Maier</lastname>…</empno><salary>1234.56</salary></record>
<record><lastname>Schwarz</lastname>…</empno><salary>1234.56</salary></record>
```

ESDS Datei wir mit VSAM Redirektor ‚redirected‘, d.h. die Records werden direkt zu einem anderen System geschickt.

- Nur noch 1 Schritt: Erzeugen (Übertragen per FTP entfällt)
- Selbes Programm wie im vorigen Beispiel
- XML Struktur immer noch im Programm hart codiert

ML Daten erzeugen – Beispiel 3

VSAM ESDS Cluster + VSAM Redirektor

Mueller Klaus Munich 12345 12345678 1234,567
Maier Dieter Stuttgart 12345 12345678 1234,567
Schwarz Hans Dortmund 12345 12345678 1234,567

Daten werden **binär** in ESDS Cluster geschrieben.
ESDS Datei wir mit VSAM Redirektor ‚redirected', d.h. die Records werden direkt zu einem anderen System geschickt.
Erst der Redirector Handler (Java) erzeugt das XML Dokument.

- Nur noch 1 Schritt: Erzeugen (Übertragen per FTP entfällt)
- XML Struktur wird erst im Handler bestimmt
- Java bietet komfortable Möglichkeiten um mit XML Daten umzugehen

ML Daten erzeugen – Beispiel 4

POWER LST Entry erzeugen + AutoFTP/AutoEMAIL

```
<?xml version="1.0" encoding="UTF-8"?>
<record>
 <lastname>Mueller</lastname>
 …
 <salary>1234.56</salary>
</record>
```

XML Dokument wird in POWER Entry erzeugt (z.B EXEC CICS SPOOL ... oder mit Batch Programm).
Der POWER Entry wird dann per AutoFTP oder AutoEMAIL automatisch zur Weiterverarbeitung verschickt

ML Daten einlesen/parsen

- **Da XML plain Text ist, muss es zuerst geparsed werden, bevor es weiterverarbeitet werden kann.**
- **Die Struktur muss validiert werden (DTD)**
- **Die Datentypen müssen erkannt werden (Schema)**
- **Die Nutzdaten müssen in das vom Programm erwartete Datenformat konvertiert werden (de-serialisiert)**

ML Daten einlesen/parsen - Beispiel 1

- **Aufbau + Struktur des einzulesenden XML Dokuments ist bekannt und „fest“**
- **Nutzdaten können mit etwas Aufwand aus dem XML Datenstrom extrahiert werden**
  - Feste Zeichen-Offsets vom Anfang des Dokuments
  - Vereinfachtes Parsen des XML Dokuments
- **Problem:**
  - Unflexibel bei XML Struktur Änderungen
  - Struktur muss fest und bekannt sein

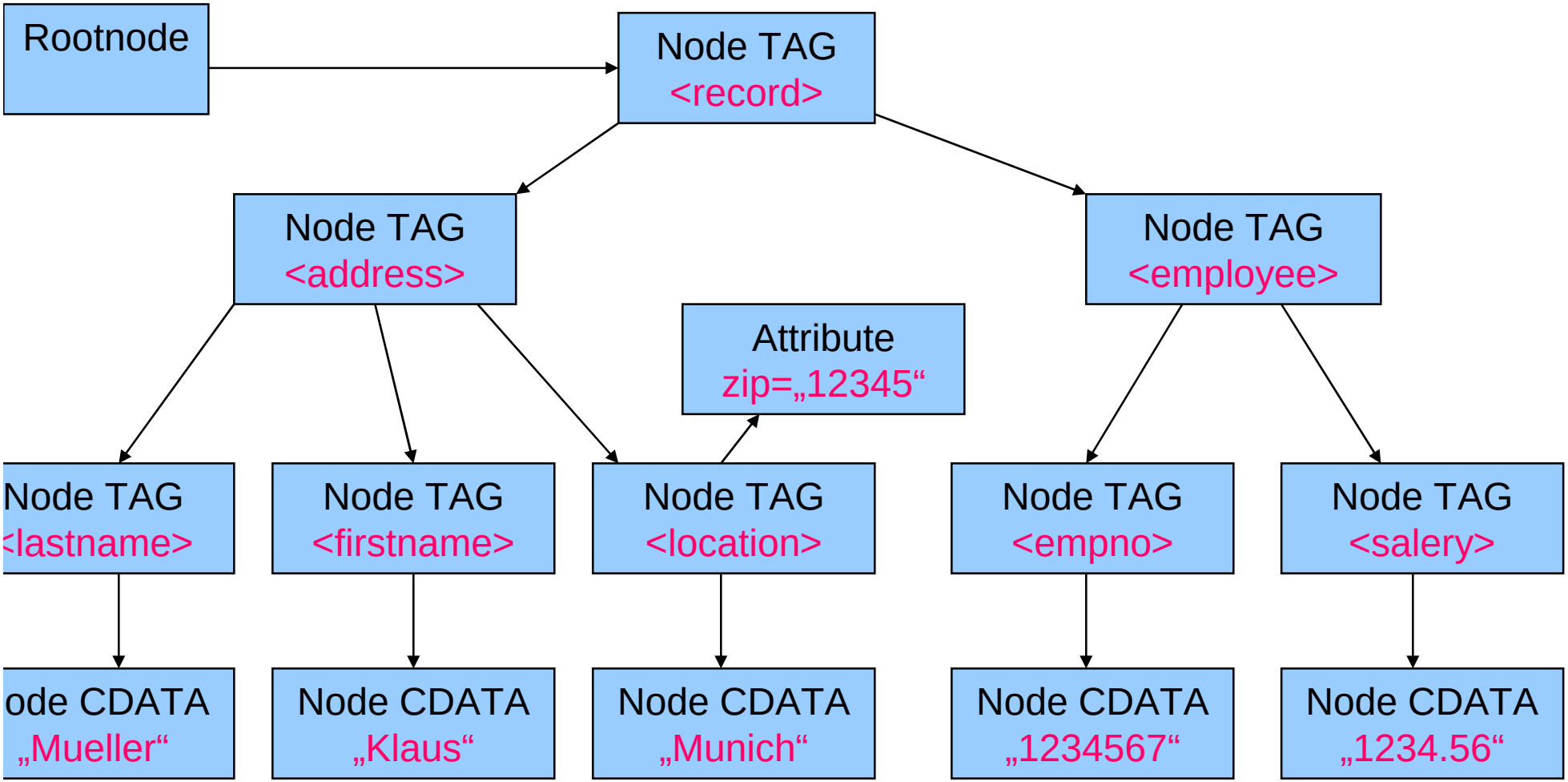
ML Daten einlesen/parsen - Beispiel 2

- **Verwenden eines XML Parsers (Teil der VSE Connectors/SOAP)**
  - SAX-Parser (eventbasiert)
    - Ruft eine Callback-Funktion für jedes Tag, Attribut, Daten, …
  - DOM-Parser (XML Baum im Speicher)
    - Baut einen Baum aus verpointerten Kontrollblöcken im Speicher
    - Der Baum kann hinterher bequem durchsucht werden
- **Nur ganze XML Dokumente lassen sich parsen**

SE XML Parser (DOM)

- **Aufruf**
  - EXEC CICS LINK PROGRAM(IESXMLCA)
  - COMMAREA enthält Parameter (Buffers, Funktions-Code, Return-Codes, …)
- **Funktionen**
  - **BUFFER2TREE**: Parsed ein XML Dokument und baut einen verpointerten XML-Baum im Speicher
  - **TREE2BUFFER**: Erzeugt ein XML Dokument aus einem verpointerten XML-Baum im Speicher
  - **FREETREE**: Gibt einen XML-Baum wieder frei, der mit BUFFER2TREE erzeugt wurde
- **ftp://ftp.software.ibm.com/eserver/zseries/zos/vse/download/vsecon/XMLParser.pdf**

SE XML Parser (DOM) – XML Baum



ML Anwendungen im VSE

- **WebServices (SOAP) Support**
  - SOAP = Simple Object Access Protocol
  - XML basierendes Protokoll
  - Vergleichbar mit Remote Procedure Call (RPC)
  - CICS Programm kann von außen als WebService aufgerufen werden
  - CICS Programm kann externen WebService aufrufen
  - Anwendungen (Programme) sehen nichts mehr von XML

imple Object Access Protocol (SOAP)

Aufruf:

```
<soap:Envelope>
  <soap:Body xmlns:m="http://www.stock.org/stock" />
    <m:GetStockPrice>
        <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

Antwort:

```
<soap:Envelope>
  <soap:Body xmlns:m="http://www.stock.org/stock"
    <m:GetStockPriceResponse>
      <m:Price>34.5</m:Price>
    </m:GetStockPriceResponse>
  </soap:Body>
</soap:Envelope>
```

ML Datenaustausch über HTTP

- **CICS Web Programm nimmt HTTP Requests mit XML Daten entgegen**
  - Externes System schickt HTTP POST Request an VSE
  - CICS Programm benutzt EXEC CICS WEB … um die XML Daten zu empfangen
  - Kann XML Parser aufrufen, um die XML Daten zu parsen
  - Erstellt Antwort-XML-Dokument per EXEC CICS DOCUMENT …

ML Datenaustausch über HTTP

- **CICS Programm erzeugt HTTP Requests mit XML Daten**
  - Programm erzeugt XML Daten im Speicher
  - Setzt HTTP POST Request ab und überträgt die XML Daten (z.B. mit HTTP Client, Teil der VSE Connectors)
  - Empfängt HTTP Response

equirement

- **Provide text transformation under VSE/ESA from VSE based formats to XML formats**
  - Customer requirement is to get assistance in transforming VSE based data to XML format
    - VSAM data transformation based on a COBOL copybook
- **Fragen:**
  - Welche Struktur soll das erzeugte XML haben?
  - Sollen Datentypen per Schema ausgewiesen werden?
  - Rückweg: XML einlesen und in Record konvertieren?
- **Mögliche Lösung:**
  - API, die anhand eines einmal aus dem Copybook erzeugten Mappings die Record-Daten in XML überführt
    - Wo wird die Mapping-Information gespeichert?
  - Codegenerator der den benötigten Code automatisch erzeugt

Dokumentation + Links

- **XML Parser Interface:**
  ftp://ftp.software.ibm.com/eserver/zseries/zos/vse/download/vsecon/XMLParser.pdf
- **HTTP Client Interface:**
  ftp://ftp.software.ibm.com/eserver/zseries/zos/vse/download/vsecon/HTTPClient.pdf
- **WebServices by IBM: Overview:**
  http://www.ibm.com/software/solutions/webservices/
- **Simple Object Access Protocol (SOAP)**
  http://www.w3.org/TR/SOAP/
- **Extensible Markup Language (XML)**
  http://www.w3.org/XML/
- **WebService im VSE**
  http://www.ibm.com/servers/eserver/zseries/os/vse/software/vsesoap.html

ragen ?