
E-Business mit XML4cobol Web Services

Maas High Tech Software GmbH

Filderstadt

Autor: Thomas Funke

Stand vom: 01.03.03

Dokument: X4CWS_whitepaper_technologyinfo.doc

Alle in dieser Publikation verwendeten Soft- und Hardware-Bezeichnungen, sowie Markennamen und Produktbezeichnungen der jeweiligen Firmen unterliegen im allgemeinen warenzeichen-, marken- oder patentrechtlichem Schutz.

© 2003 Maas High Tech Software GmbH
Alle Rechte vorbehalten.

Maas High Tech Software GmbH,
Hornbergstr. 49, D-70794 Filderstadt
Telefon: +49 (0) 711 – 77917 – 0
Internet: <http://www.maas.de>

Inhaltsverzeichnis

Zusammenfassung	1
1 Designziele.....	2
2 Die Eigenschaften von XML4cobol Web Services.....	3
3 Die Architektur von XML4cobol Web Services.....	4
SOAP und die möglichen Transportprotokolle	4
Zum Thema Sicherheit	5
Die Verarbeitung eingehender Dienste-Anforderungen	6
Context Handler, Name Service, Error Handler	8
Das systemübergreifende Dienstverzeichnis UDDI.....	10
XML4cobol / XML4pl1 Enterprise	10
4 Vorgehensweise zur Integration bestehender Anwendungen.....	12
Voraussetzung: Was macht einen Web Service aus?	12
Schritt 1: Analyse des bestehenden Systems und der Infrastruktur.....	14
Schritt 2: Analyse der bestehenden Programme	14
Schritt 3: SOAP Server und Infrastruktur an zusätzliche Anforderungen anpassen	16
Schritt 4: Datenstrukturen und Serviceschnittstelle festlegen.....	16
Schritt 5: Rahmen-Programme für COBOL erstellen.....	16
Schritt 6: Testtreiber erstellen	17
5 Schlußbemerkung.....	18
6 Weitere Informationen.....	19

Zusammenfassung

Wenn es darum geht, bestehende Anwendungen für ein durch E-Business in Bewegung geratenes Geschäftsumfeld aufzurüsten, sind vor allem pragmatische Lösungen gefragt. Die Familie von XML Standards, die unter dem Schlagwort „Web Services“ bekannt werden, öffnen neue Perspektiven über Systemgrenzen hinweg.

Die bewährten Systeme weiterhin nutzen und neue Technologien gezielt nur dort einsetzen, wo sie wirklich Nutzen bringen, ist das Leitmotiv der Integrationsspezialisten bei Maas.

Maas langjährige Erfahrungen bei der Integration von Anwendungen auf dem IBM Mainframe haben zu dem neuen Produkt XML4cobol Web Services geführt. XML4cobol Web Services ist eine Kombination aus Entwicklerwerkzeugen, Laufzeitmodulen und Methodik, die es Firmen ermöglicht, in eigener Regie bewährte Systeme für Web Services zu öffnen. Es sind in der Regel zwei Ziele, die sie verfolgen:

- Bestehende Anwendungen sollen mit einem E-Business Server (J2EE oder .NET) kommunizieren.
- Neue Anwendungen sollen die Vorzüge des Mainframe nutzen (Leistung, Skalierbarkeit, Sicherheit, Kontrolle, Zuverlässigkeit).

Dieses Whitepaper beschreibt XML4cobol Web Services – die Designziele und die zugrundeliegenden Technologien. Es beschreibt auch die Vorgehensweise für die Integration bestehender Anwendungen.

1 Designziele

Bei der Entwicklung von XML4cobol Web Services waren die folgenden Ziele wichtig:

- **Mainframe-Stärke:** Wenn neue E-Business Systeme eingeführt werden, sollten diese die Stärken der Plattform IBM OS/390 nutzen.
- **Nutzung der Infrastruktur:** Die vorhandenen Strukturen sollten weiterhin genutzt werden: Die Infrastruktur für die Kommunikation, der leistungsfähige Rechner, das Transaktionssystem, die Berechtigungssysteme, die Datenrepositories, das Datenarchiv.
- **Koexistenz von neu und alt:** Es sollte prinzipiell möglich sein, neue Services (Dienste) zu entwickeln und bestehende Anwendungen als Services zu integrieren.
- **Transaktions-Orientierung:** Web Services sollten in das Transaktionssystem integriert bleiben. Daraus resultierten hohe Anforderungen an die Performanz für Online-Transaktionssysteme.
- **SOAP Schnittstelle für COBOL:** Web Services kommunizieren über das SOAP-Protokoll. Ein eigenständiger COBOL-basierter SOAP-Server sollte dafür sorgen, dass die Integration direkt erfolgt.
- **Integration der COBOL Entwickler:** Die bisherigen COBOL Entwickler sollten ebenfalls befähigt werden, „Ihre“ Services ohne zusätzliche Client/Server-Programmierung einzuführen. Es entfällt somit der Aufwand einer gegenseitigen Abstimmung mit Java/C++-Entwickler und die Komplexität durch gemischte Teams/Systeme.
- **Einfachheit:** Die Nutzung durch COBOL-Entwickler und der Mainframe Betrieb sollte einfach sein. Die Schnittstelle sollte „erwartungs-konform“ in die Großrechner-Denkwelt der COBOL Entwickler passen.
- **Performance:** Der Einsatz von XML hat viele Vorteile. Diese durften aber nicht zu Lasten der in Mainframeumgebungen typisch hohen Anforderungen an die Verarbeitungsgeschwindigkeit gehen. Der Durchsatz sollte durch Einsatz von XML Webservices-Middleware nicht schlechter sein als beim Einsatz anderer Middleware wie z.Bsp. CTG oder IMS Connect.

Die Implementierung von XML-Schnittstelle in COBOL Programmen führte Schritt-für-Schritt zu den Produkten XML4cobol Base und XML4cobol Enterprise, die bereits seit 2 Jahren produktiv bei verschiedenen Kunden eingesetzt werden.

XML4cobol Web Services basiert auf bewährter XML4cobol-Technologie und bietet zusätzliche Erweiterungen für die wichtigsten Protokolle, die allgemein unter dem Begriff „Web Services“ zusammengefasst werden. Auch dieses Produkt ist bereits produktiv bei Großrechneranwendern im Finanzbereich im Einsatz.

2 Die Eigenschaften von XML4cobol Web Services

Die zugrundeliegende XML4cobol Technologie unterstützt mit einfachen, wenig technischen Schnittstellen für die Anwendungsprogrammierung prinzipiell den XML-formatierten Datenaustausch.

XML4cobol Web Services ist eine SOAP-Server Implementierung. SOAP (das Simple Object Access Protocol) spezifiziert, wie Programme andere Services aufrufen, wie diese Parameter übergeben und wie sie Nachrichten im Fehlerfall austauschen. Der SOAP Server läuft bereits unter Kontrolle des Transaktionssystems CICS/IMS. Als SOAP-Server bietet XML4cobol Web Services COBOL-basierte Anwendungen als XML-basierte Dienste nach außen an.

Für den Zugriff aus Großrechner-Programmen auf externe Webservices, zum Beispiel auf einen Java Server, bietet XML4cobol Web Services einen SOAP-Client. Mit diesem können Dienste von externen Systemen angefordert werden.

Intern kennt XML4cobol Web Services alle XML-verarbeitenden COBOL Programme und ruft diese auf. Die COBOL Anwendungen laufen als Transaktionen unter den IBM Transaktionssystemen CICS oder IMS. Eine CICS oder IMS Transaktion wird extern über einen XML Request angestoßen, die Ergebnisse werden als XML Response zurückgegeben.

XML4cobol Web Services hat bereits in konkreten Installationen bewiesen, dass es den Performanceanforderungen einer CICS/IMS Umgebung gerecht wird.

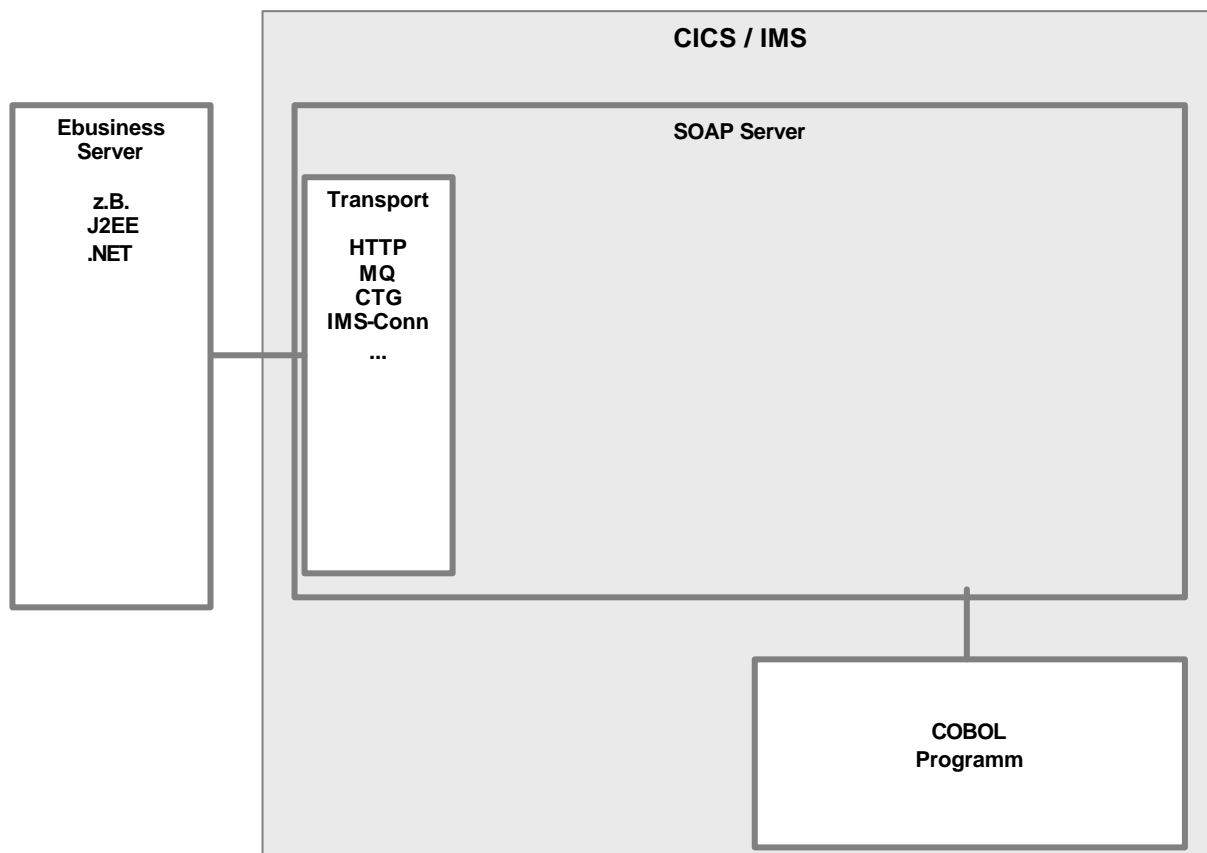
XML4cobol Web Services unterstützt:

- Heute CICS und IMS auf OS/390 und z/OS. Die Unterstützung von AS/400 und BS2000 ist derzeit in Planung.
- Verschiedene gängige Transportprotokolle (HTTP, MQ, TCP/IP, APPC).
- Sessioncontext und Errorhandling.

3 Die Architektur von XML4cobol Web Services

SOAP und die möglichen Transportprotokolle

Für unsere Darstellung nehmen wir als Ausgangssituation: Ein E-Business Server (meist J2EE od. .NET) möchte einen Service, der als COBOL-Programm auf dem Mainframe implementiert ist, aufrufen. Die Middleware hierfür heißt Web Services mit einem zentralen SOAP Server.



SOAP (das Simple Object Access Protocol) definiert ein XML Format für die Kommunikation zwischen Programmen. Als Kommunikations-Mechanismus ist ein RPC Remote Procedure Call definiert. Der Aufruf erfolgt über den SOAP Request, bei Bedarf werden Parameter mit übergeben. Die Ergebnisse werden als SOAP Response zurückgegeben. Im Fehlerfall ist stattdessen als Ergebnis der SOAP Fault vorgesehen.

Der E-Business Server agiert als SOAP Client und fordert vom SOAP Server auf dem Mainframe den Dienst an. Der SOAP Server läuft bereits im Transaktionssystem CICS bzw. IMS. Der Dienst ist in COBOL von Mainframe-Experten implementiert.

Das Transportprotokoll überträgt den XML-formatierten SOAP-Request/Response. Als Standardprotokoll für den Transport ist HTTP. Andere Protokolle sind zulässig. Je nach der zum Host bevorzugten Connectivity kann frei gewählt werden:

- HTTP
- MQSeries
- CTG
- APPC
- IMS Connector

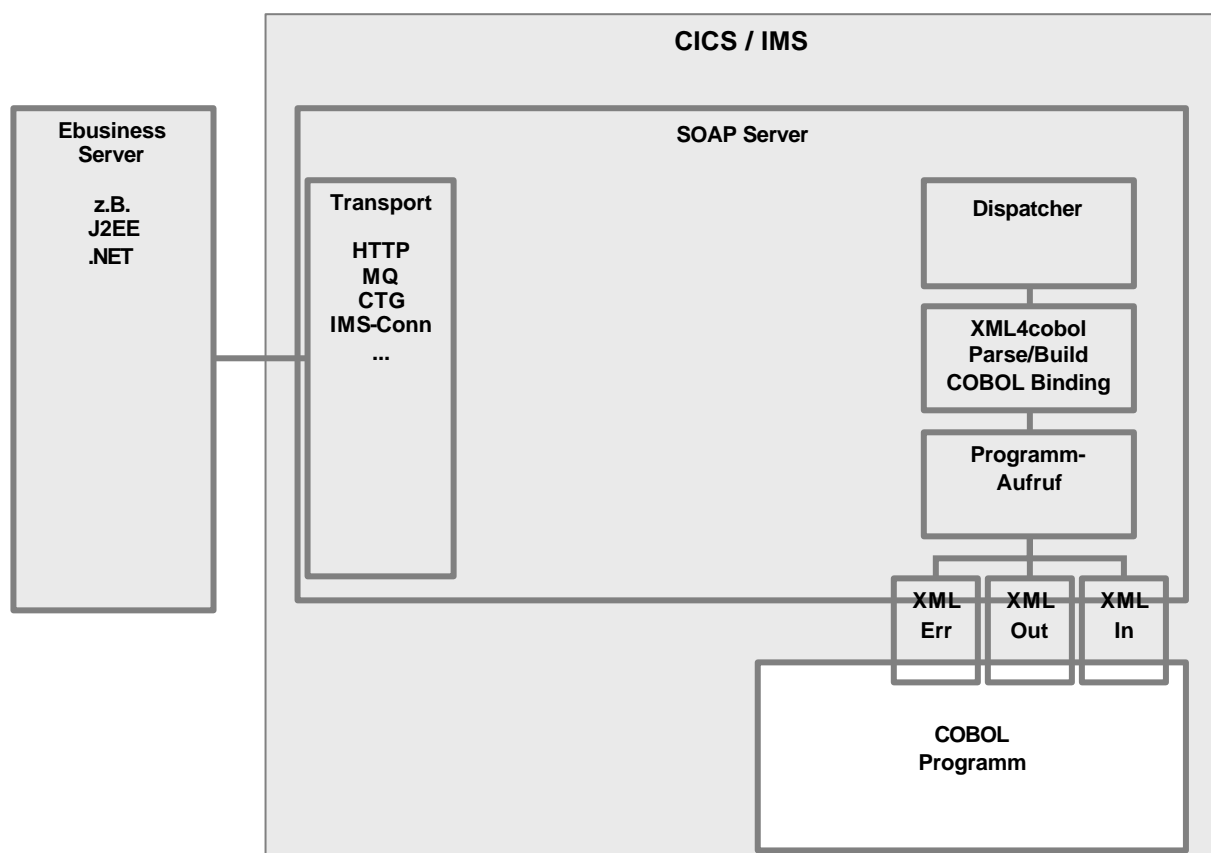
Da nur http als Protokoll in den Ebusiness-Entwicklungswerkzeugen für Webservices unterstützt wird, erhöht sich bei Einsatz eines anderen Protokolls der Aufwand auf der SOAP-Client Seite.

Zum Thema Sicherheit

Standards für die Sicherheit in einem Web Services Umfeld sind noch nicht ausdiskutiert. Dieses Thema wird deshalb systemspezifisch gelöst.

Die Verarbeitung eingehender Dienste-Anforderungen

Innerhalb des SOAP Servers analysiert ein Dispatcher die SOAP-Protokollinformationen. Der Dispatcher startet eine fachliche Anwendung, die auch als Transaktion implementiert sein kann. Die Vorteile hierbei: Transaktions-Accounting und vorhandene RACF-Mechanismen können genutzt werden.



Bevor XML Dokumente an die COBOL Programme weitergereicht werden, werden diese von generierten XML4cobol Parsermodulen aufbereitet. Umgekehrt werden Ergebnisse aus COBOL von XML4cobol Builder-Modulen wieder zu XML Dokumenten zusammengefügt. XML Parser und XML Builder lassen sich mit Hilfe von XML4cobol leicht in COBOL bzw. PL1 generieren.

XML Parser und XML Builder sind modular deklariert und können einzeln und direkt aufgerufen werden. Diese Modularität gestattet es, effizient XML Daten als Fragmente anzusprechen.

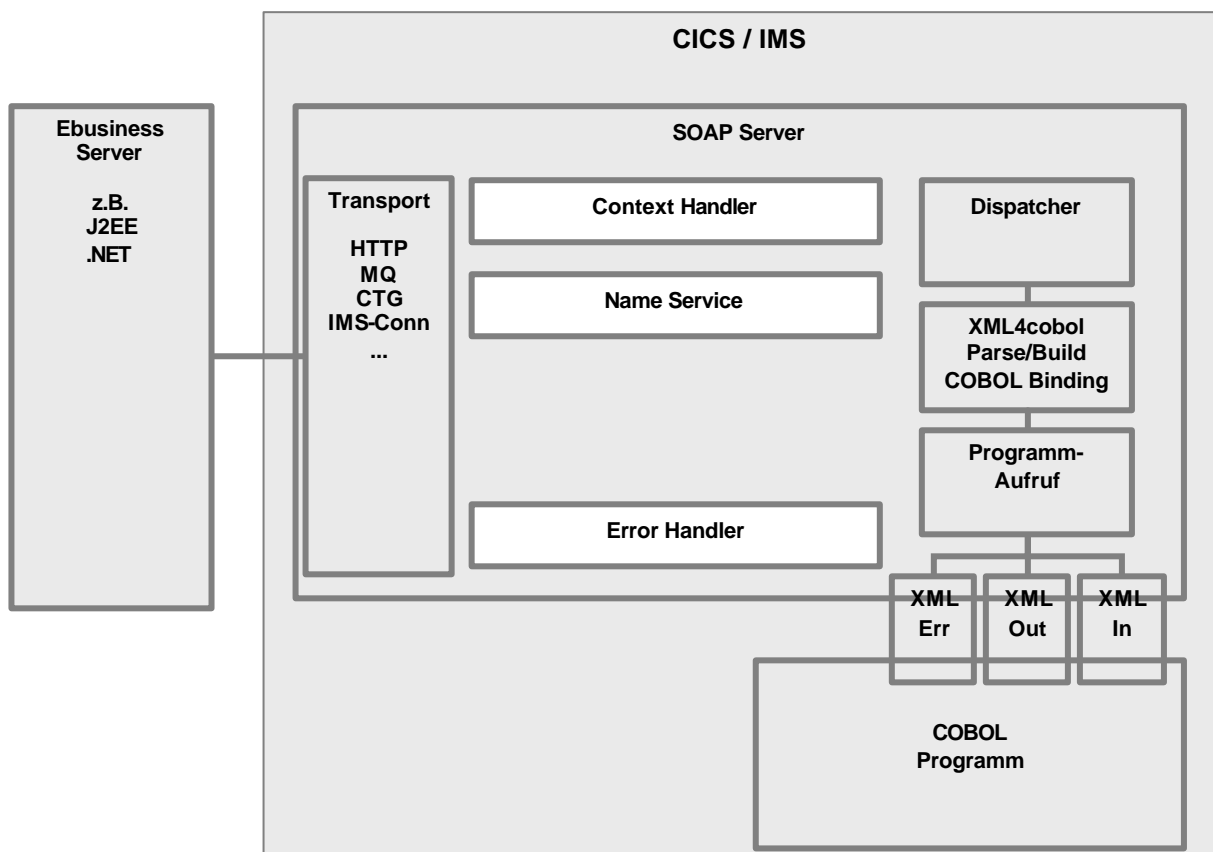
Nach der Analyse der eingehenden Dienstanforderung ruft der Dispatcher das jeweilige COBOL-Programm auf. Als Parameter werden Handle (=Zeiger) auf XML4cobol-Laufzeitstrukturen übergeben:

- Handle auf die im Request gelieferten XML IN-Parameter
- Handle auf die vom Programm zu erzeugenden XML OUT-Parameter
- Handle auf das gegebenenfalls vom Programm erzeugte Error, bzw. FAULT XML Dokument.

Der Dienst – das COBOL Programm - verarbeitet die IN-Parameter und liefert als Ergebnis entweder die OUT- oder die FAULT Parameter. Die leistungsfähige XML4cobol API (CALL Interface) macht die Arbeit mit XML Dokumenten für COBOL-Entwickler einfach. Wenn vorhandene Programme ohne Änderung angebunden werden sollen, hat das gerufene COBOL Programm die Funktion eines Mappers XML->COBOL-pur.

Context Handler, Name Service, Error Handler

XML4cobol Web Services bietet als weitere Komponenten einen Context Handler, einen Name Service und einen Error Handler.



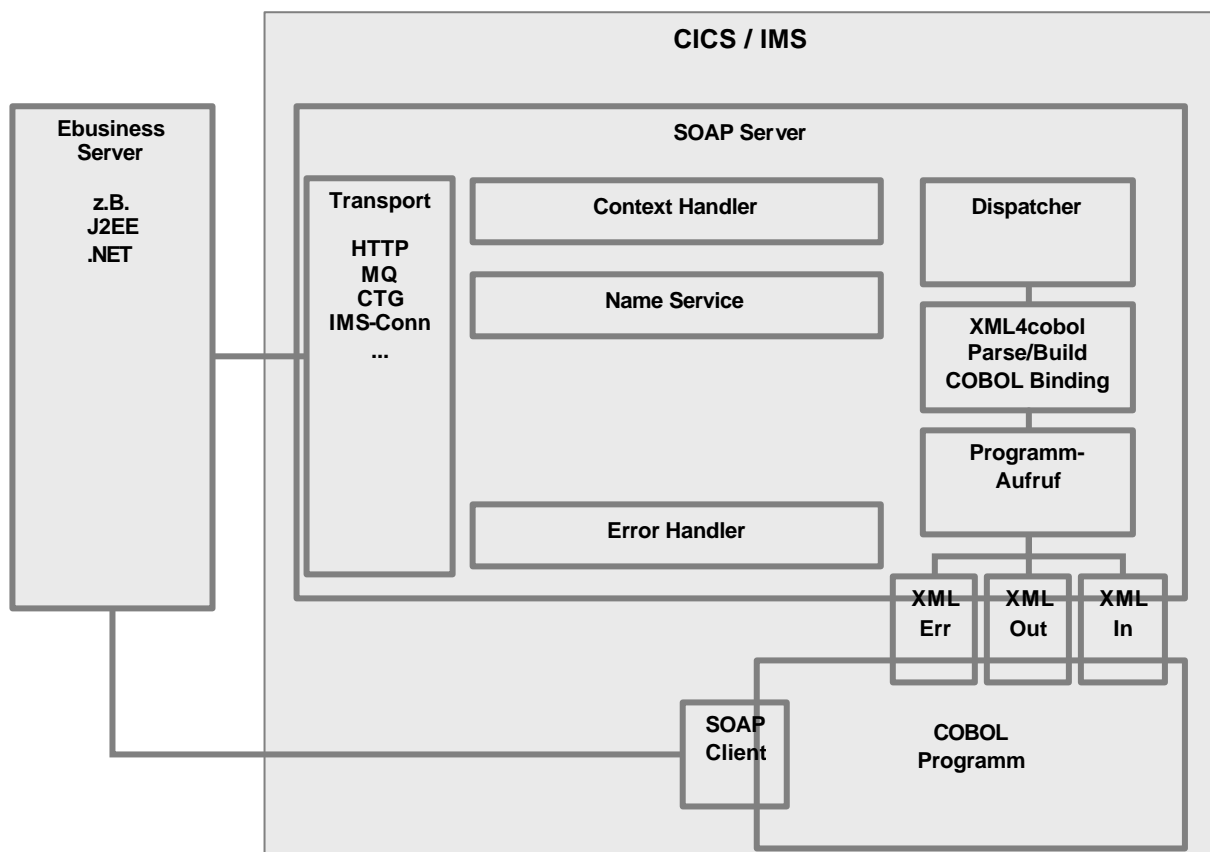
Der Context Handler ermöglicht Sessions und Single Logon. Was noch nicht in den Standards für Web Services spezifiziert ist, kann heute als zulässige Erweiterungen zum SOAP-Protokoll implementiert werden. Ein "Gedächtnis-Ansatz" für den transaktionsübergreifenden Kontext in neuen Anwendungen ist vorhanden. Ein evtl. bereits vorhandenes Session-Gedächtnis für Anwendungen im „Conversational“-Stil kann genutzt werden. Diese Lösung ist besonders notwendig für die Anbindung von bestehenden Anwendungen.

Der Name Service stellt sicher, dass das korrekte Programm gerufen wird. Hier werden als Informationen die XML4cobol Module, das SOAP-Cobol-Serviceprogramm und optional auch Informationen für den Start einer Unter-Transaktion bereitgehalten. Der Name Service kann tabellengesteuert oder über andere Informationsmechanismen z. B. mit Hilfe von Info-Programmen erfolgen. Daneben werden hier die WSDL-Dokumente bereitgehalten.

WSDL Web Services Description Language ist ein weiterer XML Standard für Web Services. Ein WSDL Dokument beschreibt den Dienst mit seinen Operationen, deren Schnittstellen und wo die Operationen aufgerufen werden können.

Der ErrorHandler interpretiert Systemfehler und erzeugt daraus entsprechende FAULT Antworten an den SOAP-Client. Möglicherweise ist eine Anpassung an die vorhandene anwendungsspezifische Fehlerbehandlung sinnvoll und notwendig.

Ein SOAP-Client ermöglicht dem COBOL Entwickler, bei Bedarf auch andere Services aus dem Internet – beispielsweise von einem E-Business-Server – aufzurufen und zu integrieren.



Das systemübergreifende Dienstverzeichnis UDDI

UDDI Universal Description, Discovery and Integration ist als zukünftige Spezifikation für Dienstverzeichnisse heute in Sicht. Ein UDDI Verzeichnis listet Dienste und hält deren WSDL Informationen bereit. WSDL Web Services Description Language beschreibt den Dienste und deren Aufrufchnittstelle. Ein UDDI Verzeichnis gestattet Programmen nach Diensten zu suchen und diese mit Hilfe von WSDL Informationen aufzurufen.

XML4cobol Web Services ist für den Einsatz von UDDI vorbereitet. Der Name Service kennt die WSDL Dokumente der Cobol Services. Der SOAP Server kann alle WSDL Dokumente an einen UDDI Server liefern und somit Dienste registrieren.

Als UDDI Server kann beispielsweise der IBM Websphere UDDI Directory Server dienen. Diese Option ist jedoch nur in einer geschlossenen Umgebung sinnvoll. Im Internet ist eine UDDI Lösung noch nicht wirklich produktiv verfügbar.

XML4cobol / XML4p1 Enterprise

XML4cobol Enterprise ist ein Produkt von Maas, mit dem Anwendungsentwickler XML-Schnittstellen direkt in die sprachspezifische Laufzeitumgebung ihrer COBOL Anwendung implementieren. Es gibt auch eine Variante dieses Produkts mit Unterstützung für PL/1.

XML4cobol Enterprise basiert auf einem einfachen Konzept und einem Sourcecode-Generator, der aus einer DTD (Dokument Typ Definition) oder einem Schema für ein XML Dokument die entsprechenden COBOL, bzw. PL/1 Module generiert. Generiert werden Cobol Copybooks für die Datenstrukturen und strukturspezifische Parser/Builder-Module, die XML Daten auf COBOL Datenfelder abbilden (= mappen) und umgekehrt. Durch den Generatoransatz wird maximale Performance beim Parsen und Mappen erzielt.

Die Funktionsweise der von XML4cobol generierten Module in Kürze:

- Der von XML4cobol erzeugte Parser liest ein eingehendes XML Dokument ein, prüft es auf die Regeln der DTD und bereitet es für den optimierten Zugriff intern und speicherresident als XML Elementenbaum auf.
- Mit Hilfe der generierten Copybooks lassen sich die speicherresidenten XML Daten 1:1 auf COBOL Datenfelder abbilden. Der Zugriff auf die XML Daten erfolgt über die XML4cobol API. Mit der XML4cobol API läßt sich der XML Elementenbaum gezielt bearbeiten und auch dynamisch erweitern.
- Der von XML4cobol erzeugten Builder erzeugt aus den speicherresidenten XML Daten gültige XML Dokumente.

Eine Randbemerkung: Die einfachere Variante von XML4cobol ist das Produkt XML4cobol Base (das Vorgängerprodukt heißt XML4cobol Version 1). XML4cobol Base erzeugt für eine beschränkte Zahl von Datentypen und für einfache Datenstrukturen ebenfalls Parser/Builder-Module und Copybooks. Der Mapping-Mechanismus von XML4cobol Base ist einfacher: Der Parser liest die Daten eines eingehenden XML Dokuments direkt in die Felder der COBOL Copystrecke im Working-Storage ein, der Builder erzeugt direkt aus den Daten der COBOL Copystrecke das XML Dokument. XML4cobol Base bietet keine dynamische Speicherverwaltung und keine Programmierschnittstelle für Zugriff auf Teilstrukturen. Das Bereitstellen einer leistungsfähigen Laufzeitstruktur ist für die Arbeit mit komplexen XML-Dokumenten sehr hilfreich.

XML4cobol Enterprise ist ein Lösungskonzept für Anwender, die selber bestimmen wollen, wie ihre Anwendungsarchitektur auszusehen hat. Mit XML4cobol Enterprise erzeugen sie „Adapter“-Module, für ihre eigenen Schnittstellen. Es bleibt letztendlich frei, wie jeder Modultyp eingesetzt wird. XML4cobol ist daher geeignet für bestehende Systeme mit unterschiedlicher Komplexität.

XML4cobol Enterprise ist der minimal-invasive Ansatz, XML zu implementieren. Es verlangt nicht die komplette Reorganisation bestehender Systeme.

XML4cobol Enterprise zielt auf optimale Performanz. Die Datenstrukturen werden hart-kodiert in in COBOL/PL1 generiert. Für diese Strukturen entstehen spezifische, hoch-effiziente XML Schnittstellen. XML4cobol Enterprise vermeidet bewusst jeden interpretierenden Ansatz und räumt auf diese Weise die bekannten Performanz-Probleme von Vielzweck-XML Parsern aus.

Mit XML4cobol Enterprise verfolgt Maas eine Lösungsstrategie, die auch langfristig trägt. In dieser Strategie erfolgt die Implementierung einer Lösung stets nach dem Baukastenprinzip. So kann Maas jederzeit eigene Lösungen durch Standard-Lösungen ersetzen. Beispiel: Maas beobachtet derzeit die Entwicklung des COBOL Compilers mit XML Unterstützung von IBM. Sobald die Erwartungen an Stabilität und Performanz erfüllt werden, kann Maas die Ergebnisse dieses Parsers einbinden und sich auf das Mapping konzentrieren.

4 Vorgehensweise zur Integration bestehender Anwendungen

Häufig stellt sich die Frage: Wie kann man mit XML4cobol Web Services bestehende Anwendungen in ein Web Services Konzept integrieren und mit welchem Aufwand?

Für die Antwort muss bekannt sein, welche Anforderungen an einen Web Services gestellt werden. Hier genügt ein Verständnis für die Beschreibung der Web Services Schnittstelle (der WSDL Web Services Description Language). Was macht einen Web Service aus?

Die Frage nach dem Aufwand läßt sich beantworten, wenn klar ist, mit welcher Strategie sich die bestehenden Schnittstellen sich auf die WSDL Schnittstelle abbilden lassen. Hierzu werden die bestehenden Programme und Schnittstellen analysiert.

Voraussetzung: Was macht einen Web Service aus?

Jede Anwendung bietet Funktionalität. Die Funktionen einer Anwendung werden als Operationen aufgerufen. Jede Operation benötigt Informationen. Nach Ausführung der Operation werden Ergebnisse zurückgeschickt. Die Kommunikation erfolgt nach einem vorgegebenen Protokoll. Das Protokoll regelt, wie Information verpackt und übertragen wird.

Diese Angaben werden in der WSDL, der Web Services Description Language abgelegt. Als Textdatei im XML Format wird eine WSDL Beschreibung auch als WSDL Dokument bezeichnet.

Die WSDL ist die Grundlage für die vollständige Beschreibung einer Web Services Schnittstelle. Mit WSDL wird ein Dienst genau beschrieben: wer er ist, was er tut, was der Aufrufer tun muss, um den Dienst zu benutzen. Es ist weniger von Belang, auf welchem Wege die WSDL zur Verfügung gestellt wird: ob in einem UDDI-Verzeichnis auffindbar oder ob der Zugriff auf die WSDL über HTTP, Message Queue oder per Email erfolgt.

Die WSDL beschreibt formal:

- **Daten** (<wsdl:types/> for data types). Mit WSDL können Datentypen deklariert werden, z.B. mittels XML Schema. Externe Deklarationen aus einem XML Schema können in ein WSDL Dokument importiert werden.
- **Nachrichten/Parameter** (<wsdl:message/> for messages). Hier sind die Datenstrukturen für Parameter gemeint, die ausgetauscht werden. Definiert werden Parameter für die Eingabe, für die Ausgabe und für den Fehlerfall. In der WSDL besteht ein Message aus einem oder mehreren Parts (Einzel-Daten).
- **Schnittstellen** (<wsdl:portType/> for interfaces). Ports sind Adressen, die den Dienst implementieren. Über die Port-Schnittstelle kann ein Dienst aufgerufen werden. Die Port-Schnittstelle wird einmal abstrakt beschrieben (port type) und konkret (binding). Port Type legt fest, wie die Software-Schnittstelle funktioniert, d.h. welche Operationen (operations), bzw. Funktionsaufrufe möglich sind. Das Binding legt fest, welches Protokoll für die Übertragung und

Verpackung von Daten zu verwenden ist, z.B. SOAP für die Verpackung und HTTP für das Transportprotokoll.

- **Dienste/Services** (<wsdl:service/> for services). Ein Dienst setzt sich aus Port-Schnittstellen zusammen.

Wenn Sie nun Ihre bestehende Anwendungen in eine Web Services Schnittstelle integrieren möchten, sind Fragen zu beantworten wie:

- Wo und mit welchem Protokoll erreiche ich den Service (über welches Binding). Man kann beispielsweise den Zugriff auf einen Service über das HTTP-Protokoll ermöglichen (indem man auf die URL <http://www.versicherung.de/webservices/KFZ-Vertragsauskunft> verweist.) Sie können aber beispielsweise auch MQSeries verwenden.
- Welche Methoden/Funktionsaufrufe (bzw. operations) stellt meine Anwendung (bzw. Service) zur Verfügung?
- Welche XML Parameter kann ich übergeben bzw. bekomme ich zurück? Oft werden in einem WSDL Dokument externe XML Schemas importiert, um Datentypen zu deklarieren. Als Beispiel für eine Referenz auf ein XML Schema könnte sein: <http://www.versicherung.de/schemas/KFZ-Vertrag>.

Das WSDL Dokument liefert alle Informationen, die man benötigt, um einen Client zu programmieren. Auch kann man die SOAP-Server-Steuerung aus der WSDL generieren. Die Unterstützung von WSDL-Clients ist allgegenwärtig. WSDL beschleunigt die Service-Programmierung erheblich. Beispiele hierfür:

- IDE Werkzeuge nutzen die Informationen in einer WSDL. Sie „wissen“ kontext-sensitiv, was die in einer WSDL beschriebenen Dienste anbieten und unterstützen den Programmierer bei der Kodierung. Bei Einsatz von http als Transport können sie bereits aus der WSDL alle notwendigen Programme erzeugen.
- Client-Anwendungen (z. B. in Microsoft Excel XP) nutzen die WSDL, um beispielsweise mathematische Funktionen auf dem HOST aufzurufen.
- XML-Entwicklerumgebungen wie der XMLSpy nutzen die WSDL, um in „Echtzeit“ den Aufruf von Web Services zu testen. Aus der WSDL läßt sich vollautomatisch eine SOAP-Anfrage an einen externen ISP richten, einen Web Service aufrufen und die Ergebnisse des Service Providers auswerten.

Es gibt bereits viele Werkzeuge am Markt, die Java-Zugriffsklassen oder Testtreiber aus einem WSDL Dokument generieren können. Hier werden allerdings bestimmte generatorspezifische Restriktionen auferlegt. Es wird zum Beispiel ein Bind-Mechanismus nur für HTTP vorgesehen. Auch unterscheiden sich die Werkzeuge in den Zielplattformen, die sie unterstützen: Microsoft, IBM, Sun.

Während man in der allgemeinen Beschreibung von Diensten und Schnittstellen viele Gemeinsamkeiten zu bestehenden Systemen finden kann, sollte auf einen wesentlichen Unterschied

der bisherigen Schnittstellen hingewiesen werden: Anstelle eines in der IT üblichen generischen Modellierungsansatzes werden XML Strukturen so spezifisch wie möglich modelliert. Dieses Vorgehen bringt Vorteile:

- Die Schnittstelle kann leichter von Partnern oder vielen Projekten genutzt werden, die nicht über Insiderwissen über die Schnittstelle verfügen
- Konkrete Strukturen verhindern Fehlinterpretationen. Beispiel: „Person hat explizit eine Adresse mit Strasse, PLZ...“ anstelle von „...Container mit Name Person hat Container mit Name Adresse hat Feldliste mit Name-Wert Paaren.“
- Genaue Formatierungsanweisungen ermöglichen auch das Einhalten von Formaten, beispielsweise von einer korrekten Kundennummer im Sinne des Empfängers (z.B. 2 Zahlen, Bindestrich, 4 Buchstaben)
- Da eingehende Daten eine hohe Konsistenz und Plausibilität aufweisen, ist die automatische Verarbeitung beim Empfänger im hohen Grade möglich.

Schritt 1: Analyse des bestehenden Systems und der Infrastruktur

Bereits etablierte Standards sind zu berücksichtigen: Standards für Transaktionssysteme zur Wahrung eines „conversational Gedächtnisses“ - eines transaktionsübergreifenden Kontexts – für die Berechtigungssysteme mit Authentifizierung und Autorisierung, für die Fehlerbehandlung und für die Nutzung weiterer Dienste.

Aus der Analyse der allgemein genutzten Funktionen resultieren spezifische Anforderungen an:

- Den SOAP-Server, z. B. für die Fehlerbehandlung, den Session-Kontext, den Name Service.
- Den Strukturen für das SOAP- Protokoll. Meist sind spezifische Erweiterungen des SOAP-Headers notwendig.
- Die bestehenden allgemeinen Host-Module.

Schritt 2: Analyse der bestehenden Programme

Je nach Programmtyp sind bestimmte Anpassungen im SOAP Server notwendig, z. B. um eine generische Schnittstelle auf eine konkrete Schnittstelle abzubilden. Es gibt verschiedenartige Schnittstellentypen für den Programmaufruf und auch für die Datenstrukturen. Hier unterscheiden wir folgende Muster:

Strukturiert	Erkennbar z.B. an Eye-Catchern, Hierarchien, Semantik, Abstraktionen Datenorientiert / Funktionsorientiert / Objektorientiert
Generisch	Wenig fachliche Semantik Häufige Verwendung der OCCURS-Klausel in COBOL Technische Feldnamen
Container	Orientiert an 32 K Größe Technischer Header Freier Bereich für die eigentlichen Daten / Copystrecke
All-In-One	Große Datenstrukturen, die oft ein ganzes Fachgebiet abdecken
Einfache Programmierschnittstellen	Technisch Konkret

Wichtig ist die Frage: Wie gut sind die Schnittstellen dokumentiert? Die Antwort auf die Frage ist mitbestimmend für den erforderlichen Aufwand. Ist zum Beispiel ein Repository vorhanden, so sehen wir darin eine gute Grundlage für die Generierung von Datenstrukturen und Schnittstellen.

Sind Dialoganwendungen zu integrieren, so ist zu klären, ob das System eine Schichtenarchitektur vorsieht oder nicht. Wo wird der jeweils nächste Dialogschritt entschieden? Es gilt ferner, die Art der Dialoganwendung zu bestimmen:

- Anzeige / Retrieval
- Pflege
- Assistenten mit viel Eingabehilfen, kleinen Schritten

Nicht jedes Ergebnis aus der Analyse der bestehenden Anwendungsprogramme führt notwendigerweise zu einem Integrationsaufwand. Erst die Zusammenfassung der Analyse-Ergebnisse und die Formulierung der Integrationsziele führen zu zusätzlichen Anforderungen an den SOAP Server und möglicherweise auch an die Web Services Protokollstruktur.

Sie führen auch zu der Entscheidung, wie die Mapping-Programme aussehen sollen: Reines Mapping von XML nach COBOL pur oder ob es Sinn macht, eine zusätzliche Prozesslogik einzuführen, z.B. um zwei Hostschritte als eine Operation anbieten zu können.

Auf Basis der Analyseergebnisse wird auch die Entscheidung gefällt, ob es sich lohnt, Generatoren zu erstellen für:

- Datenstrukturen
- Operations
- Mapping-Cobolprogramme

Je nach Mengengerüst und Gleichartigkeit der Programm-Schnittstellen lohnt es sich, einen Generator einzusetzen.

Schritt 3: SOAP Server und Infrastruktur an zusätzliche Anforderungen anpassen

Der Aufwand für die Anpassung der Funktionalität von SOAP Server und Infrastruktur lässt sich relativ schnell feststellen. Wir haben hierfür in einem Fallbeispiel etwa einen Zeit-Monat benötigt für die Implementierung eines SOAP Servers in einer Mainframe-Umgebung.

Schritt 4: Datenstrukturen und Serviceschnittstelle festlegen

Jeweils anwendungsspezifisch sind zwei Aufgaben zu lösen:

- Parameter in Schemas modellieren (IN/OUT Parameter und Fehler)
- Operationen in WSDL-Beschreibungen modellieren.

Schritt 5: Rahmen-Programme für COBOL erstellen

Nach Festlegung der Parameter und Service-Beschreibungen können nun aus den WSDL Dokumenten Service-Programme generiert werden. Es entstehen Copybooks, Programm-Rahmen und Beispielcode für die Parameterbearbeitung.

Auch hier ist je nach Mengengerüst und Einheitlichkeit der Schnittstellen der Einsatz eines Generators sinnvoll.

Die Integrationsziele bestimmen, was die aus der WSDL entstehenden Service-Programme leisten sollen. Soll eine neue Anwendung entstehen, so kann das Service-Programm zu einem Web Service ausgebaut werden. Sollen bestehende Anwendungen als Service integriert werden, so kann das Service-Programm dazu dienen:

- XML Daten für die bestehende COBOL-Schnittstelle aufzubereiten. Das COBOL-Service-Programm bereitet die XML Daten als Cobolstrukturen auf und ruft das entsprechende bestehende COBOL-Programm.
- Mehrere bestehende COBOL Programme nacheinander aufrufen. Hierzu muss eine fachliche Prozesslogik in die Service/Mapping-Programme gelegt werden.

Schritt 6: Testtreiber erstellen

Ein Testtreiber dient dazu, einen Dienst als Web Service auf dem Host anzufordern. Sie können den Testtreiber wie vorgesehen zu Testzwecken nutzen. Sie können diesen möglicherweise auch bereits als Rahmen für die E-Business-Anwendung nutzen, die auf den Host über Webservices zugreift.

Daneben gibt es auch Werkzeuge, die es relativ einfach machen, einen Service mit Hilfe der WSDL zu testen. Voraussetzung ist allerdings, dass als Transportprotokoll HTTP verfügbar ist.

Maas bietet Generatoren für J2EE Server wie z. B. Websphere von IBM. Diese generieren aus den WSDL Dokumenten Java-Code für den SOAP-Client.

5 Schlußbemerkung

Wir sind nicht alleine darüber begeistert, wie einfach mit Hilfe von XML Technologien Schnittstellen gebaut werden können. Unsere Kunden sind es auch – denn sie setzen seit mehreren Jahren XML4cobol und unsere XML Server für CICS/IMS produktiv für ihre EBusiness-Anwendungen ein.

Mit XML4cobol Web Services lassen sich die Grenzen zwischen Systemen schneller und kostengünstiger überwinden.

6 Weitere Informationen

Wenn Sie an weitere Informationen interessiert sind, nehmen Sie einfach mit uns Kontakt auf:

Wolfgang Maas

Maas High Tech Software GmbH

Hornbergstr. 49

70794 Filderstadt

Deutschland

Telefon: 0711 – 77917 – 0

Telefax: 0711 – 77917 – 17

Email: xml4cobol@maas.de

WWW: <http://xml4cobol.de>