The SAS® XML/A Web Services Framework
Exploiting SAS® Programs through a Web Services Architecture

Anton Fuchs
EMEA Business Intelligence Technology Strategist
Objectives

- The presentation should give an overview on what is currently possible and what is planned for the future.
- Architecture
- Case Study
- Examples
Agenda

- The importance of integration
- Web services and SAS – The Present
- Web services and SAS – The Future
- Conclusion
The importance of Integration

- Integration is an age-old problem for IT
- More and more information is produced.
- Information coming from various systems and architectures.
- It becomes more and more complex to integrate those.
The importance of Integration

- Integration without a strategy
The importance of Integration

- Integration without a strategy

N-1 different interfaces/bridges
The importance of Integration

- Integration with Web services
The importance of Integration

- Integration with message broker
The importance of Integration

“Web Services will change the face of integration solutions by taking them from being complex and expensive projects to relatively cheap and easy ones. This will have the effect of changing the market perception of integration projects and driving down costs.”

Gartner DataQuest 2002
The importance of Integration

- One aspect of integration is the ability to **interoperate** between different software systems and architectures.
  - Microsoft .NET framework
  - Sun ONE
  - IBM WebSphere

- SAS delivers a highly interoperable, hardware-agile and future-proof platform for IT.
The importance of Integration

Cornerstones to SAS Intelligence Architecture

- Usability
- Scalability
- Manageability
- Interoperability
The importance of Integration

- COM/DCOM and Java bridges
- ODBC, JDBC drivers for SAS tables
- XML interchange engine, application, APIs
- OLE DB for OLAP
- ADO/OLE DB to SAS for Windows clients
- SAS BI Web services support
- J2EE compatibility
- WAP/WML wireless SDKs
- Object Management Group CWM Standard
- LDAP support (Microsoft Active Directory, Netscape xxx)
- Message-oriented Middleware (MOM) adapters
  - WebSphere MQ, Microsoft Message Queue, Tibco Rendezvous
- SAS analytics deployable in Java/C, C++ languages
- 3rd-party RDBMS access through SAS/ACCESS
- Access to SQL Server Analysis Services from Enterprise Guide 2.0
- more, more, more...
The importance of Integration

Our goal as a company: to enhance the value of our customer’s investment in SAS, no matter what the technology exploitation tools are.
Web services and SAS – The Present

- Using SAS version 8/9 it is possible to create Web services.
  - .Net framework
  - J2EE
- SAS Enterprise Integration Community Web page at http://support.sas.com/rnd/eai/samples/
Web services and SAS – The Present

Front-Tier
- Web browser
- Desktop application
- Web service

Middle-Tier
- Web service
- Net or Java

Back-Tier
- SAS Object Server
- SAS Data Server

Connections:
- SOAP
- IOM
- JDBC/OLE DB
Noel-Levitz using .Net Web service framework with SAS 8.2

Data
- SQL
- Scoring Models

Process
- Admissions
- College
- Student
- Enrollment Management
  - Recruitment
  - Retention
  - Assistance

Solution
- Base SAS
- SAS/ACCESS to ODBC
- SAS Enterprise Miner models
- Integration Technologies
- Web services

Copyright © 2003, SAS Institute Inc. All rights reserved.
Noel-Levitz using .Net Web service framework with SAS 8.2

- Reduced time to deliver information to customers
  - “First in the mailbox”
  - Students more likely to apply
- Reduced cost to implement
  - $200K instead of $600K
  - Paradigm shift of implementation to provide instantaneous answers to customers
- Improved customer satisfaction by allowing unlimited scoring
- Increased sales
VKB AG using Java Web service to implement online credit scoring
Web services and SAS – The Future

- SAS Integration Technologies 9.1 includes support for creating Web services.
  - SAS BI web services for .Net
  - SAS BI web services for Java
- Uses XMLA specification as the mechanism
- Uses SAS Stored Processes to implement
XML for Analysis (XMLA)

- XMLA is an open industry-standard web service interface designed specifically for online analytical processing (OLAP) and data-mining functions
- SAS co-sponsors the specification together with Microsoft and Hyperion
- More information at [www.xmla.org](http://www.xmla.org)
- SAS adopted the interface to call Stored Processes
XML for Analysis (XMLA)

- The standard basically specifies 2 methods:
  - **Discover** is used to obtain information and metadata from a Web Service
    - A list of available stored processes
  - **Execute** is used to execute ... or other provider-specific commands
    - Executes a stored process and returns the results
SAS Stored Processes

- A SAS program that is stored centrally on a server
- Client application can then execute the program and receive and process the results
- Supports parameter passing
- Supports streaming tables, XML, ODS output,...
Creating a Stored Process

- **Manual authoring**
  - Write the code in SAS or a text editor
  - Use any existing SAS code
  - Register it with the SAS Management Console

- **‘Point-and-click’ authoring**
  - Enterprise Guide
  - Data Builder
  - Other SAS code generators
SAS BI Web Services

Application

Web Server

Metadata Server

Stored Process Server

HTTP/ SOAP

IOM

IOM
How to create the Web service

- Step 1: Write the SAS program
- Step 2: Register in metadata
- Step 3: Write the client code
Step 1: Write the SAS Program

- **Inputs**: Macro variables and filerefs containing XML
  - Parameter → Macro
  - Stream → Fileref (containing XML)
- **Output**: _WEBOUT fileref containing XML
- **XML Libname Engine**
- **Errors**: SYSCC and SYSMSG
Step 1: Write the SAS Program

```
libname instream xml;
libname _WEBOUT xml;

proc means data=instream.&tablename;
   output out=_WEBOUT.mean;
run;
```
Step 2: Register in metadata
Step 2: Register in metadata

SAS Code

```
libname _WEBOUT xml;
```

Client Code (Excel VBA)

```
Set nodeList = ws.wsm_Execute(...
```
Step 2: Register in metadata

SAS Code
```
libname instream xml;
```

Client Code (Excel VBA)
```
"<Stream name='instream'>
<Table><myData><coll>4</coll></myData></Table>
</Stream>"
```
Step 2: Register in metadata

SAS Code

```
proc means data=instream.&tablename;
```

Client Code (Excel VBA)

```
<Parameter name='tablename'>myData</Parameter>
```
Step 3: Write the client code - Excel

wsm_execute(command, properties)

wsm_discover(requestType, restrictions, properties)
Execute’s Properties Parameter

- `<Content>Data</Content>`
  - Possible values are Schema, SchemaData, Data, None
- `<DataSourceInfo>Provider=sassps</DataSourceInfo>`
  - Obligatory provider. Always has to be sassps.
- `<UserName>dan</UserName>`
- `<Password>abc</Password>`
  - Mandatory username and password
Execute's Command Parameter

<StoredProcess name="themeanone">
  <Parameter name="tablename">myData</Parameter>
  <Stream name="instream">xml</Stream>
</StoredProcess>

Command parameter always starts with StoredProcess with the name of the SAS Stored Process as defined in the metadata. Possible tags are <Parameter> and <Stream>
Step 3: Write the client code - Excel

Web Service References Tool 2.0

- Web Service Search
  - Search for Web services using keywords and/or business name.
  - Keywords:
  - Business Name:

- Web Service URL
  - Provide URL to WSDL or ASMX file to view Web service.
  - URL:
    - http://localhost/SASWS/saswsie.asmx

Click Web services or methods to view descriptions. Select to add to project.
Step 3: Write the client code - Excel

Sub themeanone()
    Dim ws As New clsws_SASAnalysis
    Dim nodeList As MSXML2.IXMLDOMNodeList
    Dim docCommand As New MSXML2.DOMDocument40, docProperties As New MSXML2.DOMDocument40
    Debug.Print RangeToXML(Selection)
    docCommand.LoadXml "<StoredProcess name='themeanone'>" _
        + "<Stream name='datain'>" & RangeToXML(Selection) & "</Stream>" _
        + "<Parameter name='tablename'>myData</Parameter>" _
        + "</StoredProcess>"
    docProperties.LoadXml "<PropertyList>" _
        + "<DataSourceInfo>Provider=SASSPS</DataSourceInfo>" _
        + "<Content>Data</Content>" _
        + "</PropertyList>"
    Set nodeList = ws.wsm_Execute(docCommand.getElementsByTagName("StoredProcess"), _
        docProperties.getElementsByTagName("PropertyList"))
    XMLDocToTable nodeList, i * 1 + 10
End Sub
Step 3: Write the client code - Excel

Function RangeToXML(inputRange As Range) As String
    Dim row As Range, column As Range, dataRange As Range, titleRow As Range
    Dim iRow As Integer, iCol As Integer
    Set titleRow = inputRange.Rows(1)
    Set dataRange = Range(inputRange.Cells(1, 1), inputRange.Cells(inputRange.Rows.Count, inputRange.columns.Count))
    s = "<Table>"
    ' Start this loop at 2 if your data has column headers.
    For iRow = 2 To dataRange.Rows.Count
        s = s & "<myData>
        For iCol = 1 To titleRow.columns.Count
            s = s & "<" & titleRow.Cells(1, iCol).FormulaR1C1 & ">
            s = s & dataRange(iRow, iCol).FormulaR1C1
            s = s & "</" & titleRow.Cells(1, iCol).FormulaR1C1 & ">"
        Next
        s = s & "</myData>"
    Next
    s = s & "</Table>"
    RangeToXML = s
End Function
Sub XMLDocToTable(nodeList As MSXML2.IXMLDOMNodeList, startRow As Integer)
    Dim xNode As MSXML2.IXMLDOMNode, rowNode As MSXML2.IXMLDOMNode
    Dim numRows As Integer, numCols As Integer, iRow As Integer, iCol As Integer

    ' Column Headers
    Set rowNode = nodeList(0).childNodes(0)
    numRows = nodeList(0).childNodes.Length
    numCols = rowNode.childNodes.Length
    ' The headers
    For iCol = 1 To numCols
        Cells(startRow, iCol).Select
        ActiveCell.FormulaR1C1 = nodeList(0).childNodes(0).childNodes(iCol - 1).nodeName
    Next
    ' The data
    For iRow = 1 To numRows
        For iCol = 1 To numCols
            Cells(iRow + startRow, iCol).Select
            ActiveCell.FormulaR1C1 = nodeList(0).childNodes(iRow - 1).childNodes(iCol - 1).Text
        Next
    Next
    Cells(iRow + startRow, numCols + 2).Select
    ActiveCell.FormulaR1C1 = Now
End Sub
Conclusion

- Today, SAS customers can develop Web services utilizing SAS-based technologies with existing .NET and Java infrastructures.
- SAS version 9.1 can easily publish stored processes as a Web service.
References

- XML For Analysis Specification
  http://www.xmla.org

- Microsoft Web Services
  http://www.microsoft.com/webservices

- Java Web Services
  http://java.sun.com/webservices

- SAS Integration Technologies
  http://support.sas.com/rnd/itech

- Anton.Fuchs@eur.sas.com
The Power to Know®