Exploiting SAS Servers from COM environments using SAS Integration Technologies and VB

Frank Rosner, SAS Institute
Presentation objectives

- SAS/Integration Technologies use cases for COM clients
- overview
  - Workspace Manager
  - Data access from VB(A)
About

Integration Technologies

- Application Facilities
- Distributed Objects
- Application Messaging
- Directory Services (LDAP)

Java Clients

COM Clients

This presentation

Not in this presentation:

using the builtin objects programmatically
using SCL components from VB
Agenda

- Basics
- Workspace Manager
- Data acces via ADO
  - with IOM OLE DB provider
  - other available SAS OLE DB providers
- Event trapping
Definition:

Integrated Object Model (IOM)

The set of object-based interfaces surfaced by the SAS system when it is run as an object server.
Workspace

Definition: „A SAS Integrated Object Model (IOM) workspace represents a single session with the SAS System. The workspace provides the same set of resources and facilities as an interactive or batch SAS session. “

All access to SAS initiated through Workspaces (=>additional functionality)

- LanguageService (submit, list/log, stored processes)
- DataService (librefs, plus access through ADO, OLE-DB or JDBC)
- FileService (filerefs and directories)
- Utilities (result packages, formats, host info)
Klassen
- OptionService
- Error
- PackageType
- ResultPackage
- ResultPackageFile
- ResultPackageHTML
- ResultPackageLibrary
- ResultPackageReference
- ResultPackageService
- ResultPackageViewer
- StoredProcessService
- StreamOpenMode
- TextStream
- TextStreamERRORS
- Utilities

Elemente von 'Workspace'
- Close
- DataService
- FileService
- GetApplication
- GetNewWorkspace
- LanguageService
- Name
- UniquelIdentifier
- Utilities

Class Workspace
Element von SAS
Public Sub localTest()

Dim sas As sas.Workspace

Set sas = New sas.Workspace

' this will create a new (local) workspace
' => do some work (after SAS process creation)
sas.Close
Set sas = Nothing

End Sub
IOM Connectivity: 3 options for (D)COM clients

- SAS
- VBA
- Delphi
- VBScript
- Visual Basic™
- Visual C++™
- Visual J++™
- ...

COM client

TCP/IP

MVA SAS

IOM Bridge-OS/390®, Unix®, Windows NT™

COM/DCOM

MVA SAS

COM/DCOM- Windows®
IOM Connectivity - IOM Bridge mit Windows-Client

Mainframe or Unix

SAS Workspace

Object Spawner

Create Process

COM Bridge Protocol

Startup

SAS Workspace

Windows Client

COM

MFC or Visual Basic
The SAS Workspace Manager creates and manages SAS Workspaces. It can create a SAS Workspace in one of three ways:

1. Through local COM if the SAS Server runs on the same machine as the client.
2. Through DCOM if the SAS Server runs on another machine that supports DCOM.
3. Through the IOM Bridge for COM (SASComb.dll) if the SAS Server runs on another machine that does not support COM/DCOM functionality (Unix or OS/390).

It is an in-process COM server which can be used in ASP and scripting environments.
Seen from the VBA type lib viewer

Interfaces

Class WorkspaceManager
Element von SASWorkspaceManager
WorkspaceManager Class
There are three definitions that can be created to assist in launching an IOM Server:

- Server definition (ServerDef)
- Login definition (LoginDef)
- Logical name definition (LogicalNameDef)

These definitions may be stored in a file on the local system or may be stored in a network directory (LDAP server).

The WorkspaceManager can create, update and use this information.
Workspace Manager

Client Program

Workspace Manager (DLL)

SAS (server process)

Configuration data in LDAP server

Object Spawner

may use

may use

may use (should - if Bridge used)

uses

launches, if DCOM

launches
A bit longer code: Using Workspace Manager

Public Sub WMTest()
    Dim mgr As New SASWorkspaceManager.WorkspaceManager
    Dim sas As sas.Workspace
    Dim errMsg As String
    Set sas = mgr.Workspaces.CreateWorkspaceByServer("WS1", VisibilityNone, Nothing, "<domain>\<userid>", "<password>", errMsg)
    ' use this workspace, do some work
    sas.Close
    Set sas = Nothing
    Set mgr = Nothing
End Sub
Dim mgr As New SASWorkspaceManager.WorkspaceManager
Dim sas As sas.Workspace
Dim errMsg As String
Dim sdef As New SASWorkspaceManager.ServerDef

sdef.MachineDNSName = "gerfrr.ger.sas.com"
  sdef.Protocol = ProtocolBridge
sdef.Port = 6338

Set sas = mgr.Workspaces.CreateWorkspaceByServer("WS1", VisibilityNone, sdef,
  "<domain>\<userid>", "<password>", errMsg)

' use this workspace, do some work

  sas.Close
  Set sas = Nothing
  Set sdef = Nothing
  Set mgr = Nothing
WSM components (details)

IWorkspaceManager

attribute FilePath : String
attribute Scope : Scope

ILogicalNameDefs LogicalNameDefs
IServerDefs ServerDefs
ILoginDefs LoginDefs
IWorkspaces Workspaces

void SetLDAPServer([in]String ldapURL,
                   [in]Boolean persistInRegistry )

void SetLDAPUser([in]String userName, [in]String Password,
                 [in]Boolean persistInRegistry )
WSM components (details)

IWorkspaces

long Count
IUnknown _NewEnum
attribute UseXMLInErrorInfo : Boolean

IWorkspace GetWorkspaceByName([in]String workspaceName )
IWorkspace GetWorkspaceByUUID([in]String uuidString )
void RemoveWorkspace([in]IWorkspace pIWorkspace )
void RemoveWorkspaceByUUID([in]String uuidString )
void AddExternalWorkspace([in]Visibility Visibility, [in]IWorkspace existingWorkspace )
WSM components: create workspace (1. way)

IWorkspaces

IWorkspace CreateWorkspaceByLogicalName(
    [in]String Name,
    [in]Visibility Visibility,
    [in]String LogicalName,
    [in]String ReferenceDN,
    [out]String xmlInfo )
WSM components: create workspace (2. way)

IWorkspaces

IWorkspace CreateWorkspaceByServer(
    [in]String Name,
    [in]Visibility Visibility,
    [in]IServerDef pIServerDef,
    [in]String LoginName,
    [in]String Password,
    [out]String xmlInfo )
Dim mgr As New SASWorkspaceManager.WorkspaceManager
Dim sas As sas.Workspace
Dim errMsg As String
Dim sdef As New SASWorkspaceManager.ServerDef

' mgr.Scope = ScopeGlobal
' mgr.SetLDAPServer "LDAP://gerfrr.ger.sas.com:389/o=ger.sas.com", True 'LDAP-Server
' mgr.SetLDAPUser "uid=gerfrr,ou=People,o=ger.sas.com", "<password>", True 'LDAP-UserDN + Password

Set sas = mgr.Workspaces.CreateWorkspaceByLogicalName("someName", VisibilityNone, "red", "uid=admin,o=ger.sas.com", errMsg)

' this will create a new workspace
' do some work !!!

sas.Close
Set sas = Nothing
Set sdef = Nothing
Set mgr = Nothing

LDAP-Info (can be persisted in Windows registry)

Logical Name: this is used to join LDAP data

LDAP-Info
(user, registered in LDAP (may be different to 1. user)
What happens under the cover?

The real passwords are hidden (never appear in programs) !!!
needed Type Libraries

- SAS: Integrated Object Model (IOM)
- SASWorkspaceManager
- (only for script languages) Scripto 1.0 Type Library
Demo (Workspace Manager)
Data Access via ADO/OLE DB

- Basic facts:
  - You need the ADO type libraries (part von Microsoft‘s MDAC)
  - If you have a Workspace in your hands it’s very simple to get an ADO connection
  - Create this workspace with the Workspace Manager and „VisibilityProcess“ flag
  - You should use the appropriate OLE DB provider
    - SAS has three for data tables!
ActiveX Data Objects (ADO)
SAS IOM Data Provider: get data from IOM Server

SAS SHARE Data Provider: get data from SHARE Server

SAS Local Data Provider: get local data (without SAS session)
ADO and VB

- Add ADO reference in project
  Microsoft ActiveX Data Objects 2.5 Library

- Connection Object

  Dim cn As New ADODB.Connection
  cn.Provider = "sas.IOMProvider.1"
  cn.Open

  - Starts a new SAS session and workspace
  - Would like to use an existing session ...
ADO and WorkspaceManager

- WorkspaceManager
  - Must use VisibilityProcess
  - Include Workspace ID in connection's settings
  - Can include provider info on open method

```vbnet
Dim cn As New ADODB.Connection
Dim cs As String
cs = "Provider=SAS.IOMProvider.1; SAS Workspace ID=" + obWS.UniqueIdentifier

cn.ConnectionString = cs
cn.Open
or
cn.Open cs
```
ADO Recordset

- Represents
  - entire set of records from a base table
  - or the results of an executed command

Dim obRS As ADODB.Recordset
obRS.Open Source, ActiveConnection, CursorType, LockType, Options

Options is optional.
A Long value (Type 10) that indicates how the provider should interpret the Source parameter.

Source is optional.
A Variant that evaluates to a valid Command object, an SQL statement, a table name, a stored procedure call, a file name of a persistently stored Recordset, or a URL.

ActiveConnection is optional.
Either a Variant that evaluates to a valid Connection object variable name, or a String that contains ConnectionString parameters.

CursorType is optional.
Type of cursor that the provider should use when opening the Recordset.

LockType is optional.
Type of locking (concurrency) the provider should use when opening the Recordset.
Simple Program (1)

Dim mgr As New SASWorkspaceManager.WorkspaceManager
Dim sas As sas.Workspace
Dim errMsg As String

Dim con As New ADODB.Connection
Dim rs As New ADODB.Recordset


con.Open "provider=SAS.iomprovider.1; SAS Workspace ID=" + sas.Uniqueldentifier

Defined in ADO typelib
Now needed!
Link with existing workspace
Simple program (2, end)

rs.Open "SASHELP.CLASS", con, adOpenForwardOnly, adLockReadOnly, adCmdTableDirect

rs.MoveFirst
Debug.Print rs("Name").Value

rs.Close

Set rc = Nothing
con.Close
Set con = Nothing
mgr.Workspaces.RemoveWorkspace sas
sas.Close
Set sas = Nothing
Set mgr = Nothing

Real work

Cleanup part:
additional statement
„RemoveWorkspace“
ADO Recordset

- rs.MoveFirst
- rs.MoveLast
- rs.MoveNext
- rs.MovePrevious
- rs.GetString(StringFormat, NumRows, ColumnDelim, RowDelim, NullExpr)

Create an HTML table from a recordset
sTable = "<TABLE BORDER=0><TBODY><TR><TD class=Data>" &
sTable = sTable + rtADOrs.GetString (,,"</TD><TD class=Data>", _
    "</TD></TR><TR><TD class=Data>" ) &_
    "</TD></TR></TBODY></TABLE>"
Demo (ADO with different providers)
Events und Connection Points

- The IOM event model is the COM event model (Connection Points, Advise, Unadvise).
- Example: builtin events for LanguageService:

  ```plaintext
  procedure start  event
      ProcStart(Procname)
  
  program execution completion  event
      SubmitComplete(return code)
  
  procedure completion  event
      ProcComplete(Procname);
  
  datastep start  event
      DatastepStart();
  
  datastep complete  event
      DatastepComplete();
  
  step error  event
      StepError();
  ```
COM: Connection Points


1. IConnectionPointContainer::FindConnectionPoint
2. IConnectionPoint::Advise
   (hands out sink interface pointer)
3. Server calls some methods of the sink interface ... until client calls IConnectionPoint::Unadvise
Event handling (for builtin events)

- Declare a variable of needed type, but with „WithEvents“
- Write an event handler
  - name is of the form Object_Event
- Set variable to nothing if done (to avoid trouble)
- Example:

  ```vba
  Dim WithEvents ls As SAS.LanguageService
  ...
  Private Sub ls_ProcStart(ByVal Procname As String)
      Debug.Print "Starting PROC: " & Procname
  End Sub
  ...
  Set ls = Nothing , in some Cases
  ```
Different possibilities for Workspace access via Workspace Manager with or without LDAP

Data Access with ADO and different OLE DB providers
- IOM provider
- Share provider
  (Local provider)

Event handling is simple and powerful
Questions