Supporting Windows IPC mechanisms in Samba

Jelmer Vernooij
Samba Team
jelmer@samba.org

SambaXP
May 2005
Thinking as a Windows ISV...

- Agenda
  - Transports involved
  - Discussion of available IPC Mechanisms
  - External use of RPC
  - External use of WMI
History of Windows IPC mechanisms

- Mailslots
- RPC
- DCOM
- WMI
- .NET Remoting

Win 3: Samba 1.9
NT 3: Samba 2.0
NT 4: Samba 4.0
Win2k: Samba 4.0
Win2k: Mono?
Transports

- SMB Named Pipes
- IPX/SPX
- NetBEUI
- TCP/IP
- “Local”
- ... and more
Named Pipes

- Similar to unix domain sockets
- `\PIPE\...` on local systems
- Shared over SMB as `IPC$`
Registering Named Pipes

- Plugin
- Fallback to Unix Domain Sockets(?)
RAP

- Supported by OS/2, Windows for Workgroups
- “Original” RPC mechanism in SMB
- Not user-extendible (global opnums)
Mailslots

- Used for browsing (Network Neighborhood)
- One-Way
- 425- and 426-byte messages not possible
- Uses Named Pipes for small messages, datagrams for larger ones
- Unreliable
(MS)RPC

- Based on DCE/RPC
- Works over SMB, TCP/IP or SPX
- Interfaces specified in IDL
- Used extensively by 3rd-party developers
DCOM

- Distributed version of COM
- Uses RPC underneath
- Basis of various other technologies
  - WMI
  - ActiveX
WMI

- Based on DCOM
- Windows Management Interface
- Implementation of WBEM
- Used by the Windows MMC
- Scriptable
WMI – Where it gets complicated...
WMI and the Windows MMC
WMI - Scripting

Scripting in all COM-compatible languages: VBScript, Jscript, Python(!)

For Each Host In WScript.Arguments
    Set WMIIservice = GetObject("winmgmts:{impersonationLevel=impersonate}!\" _ & host & "\root\cimv2")

    Set colsettings = WMIIservice.ExecQuery("SELECT * FROM Win32_Processor")

    For Each proc In colsettings
        Wscript.Echo(host & ": " & proc.description)
    Next
Next
using System;
using System.Management;

class Class1
{
    static void Main(string[] args)
    {
        ConnectionOptions co = new ConnectionOptions();
        co.Username = "john"; co.Password = "john";
        ManagementScope ms =
            new ManagementScope("\\\192.168.1.2\root\cimv2", co);

        ObjectQuery oq =
            new ObjectQuery("SELECT * FROM Win32_OperatingSystem");

        ManagementObjectSearcher query1 = new ManagementObjectSearcher(ms, oq);
        query1.Get()[0].InvokeMethod("Reboot", {""});
    }
}
.NET Remoting

- Mono?
- Various backends (“Channels”)
  - DCOM
  - “Simple”
  - Indigo
Reasons for external use of RPC/DCOM

- Other projects
  - Wine / ReactOS
  - OpenChange
  - GUI Utilities
  - OpenPegasus
  - Mono?
  - Support ActiveX controls in Mozilla?
  - 3rd party vendor integration
External use of RPC

- Interface implementation scenarios:
  - Standalone (just epm registration)
  - Proxy-forwarded
  - Internal (plugin-based)

- Use Samba libraries
  - Installed headers
  - pidl
GUI - gregedit
State of affairs

- *pidl*
  - Updated for being MIDL-compatible
  - Not installed (yet)

- *RPC subsystem*
  - Headers need to be installed for use by other projects

- *DCOM*
  - Very simple sample example working, working on infrastructure at the moment
  - Python support planned

- *WMI*
  - Not started (yet)
Further Resources

- Slides available at http://samba.org/~jelmer/
- http://devel.samba.org/