From: Peter Wilson (WINDOWS)
Sent: Thursday, June 26, 2003 9:44 AM
To: Bill Veghte
Subject: Mail to JimAll
Attachments: Server Foundation Roles jimall.ppt

How's this?

Jim,

A virtual team from Embedded and Server has put together a plan to deliver a composable Longhorn Server. This builds on the OS Foundations work that David & Peter presented to you in May. The attached deck describes their plan. We presented this to the WELT on 6/23.

The plan they have put together has the following benefits·

- Server customers will install fewer patches (compared to full server)
- Server will be more secure and reliable (compared to full server)
- We deliver a platform for server applications that maintains a common set of base APIs across both client and server
- This allows us to take an incremental and targeted approach to re-architecting server

The plan contains the following tasks:

- Design and implement a Foundation, common to client and server that is the platform for isolated server role apps. This foundation is made up of MinWin, Server base components, COM, Security, Networking. It removes Shell, IE/OE, multi-media, GUI, etc. from the server.
  - o Using dependency data from the BDD tool, the team has analyzed that this foundation contains ~160 binaries. Separating this foundation from the full OS will require touching ~50 binaries to remove ~ 155 dependencies.
  - o We will need to define and cost the management infrastructure for this platform, and the subset of CLR required to support the Foundation management platform and to support a subset of managed APIs for headless server apps (Server-LAPI?).
  - o Port server apps that support high priority server roles to the Foundation. This primarily involves removing dependencies on UI for management, enforced by an SDK for the server foundation.

    The data that indicates we can achieve the benefits listed above is·
    - *Server customers will install fewer patches* Based on Win2K servicing data, 40% of the patches deployed to customers affect binaries in the Foundation. Additionally by explicitly defining the foundation we can package patches to minimize impact to customers who have deployed Foundation plus server app only.
    - *Server will be more secure and reliable* The Foundation contains 6% of the binaries and 4% of the footprint of full server (not including drivers), reducing surface area for attack. Additionally it allows us to target security investments to securing the Foundation and isolated roles, rather than to locking down client components that have no value-add to server customers.
    - *We deliver a platform for server applications that maintains a common set of base APIs across both client and server.* The work items for creating the server Foundation indicate it can be delivered during M7 (with the right team in place). This allows us to

stabilize the Foundation so we maintain a common set of core binaries across client and server. Evangelizing the Server Foundation provides a Microsoft alternative to Linux for server app developers wanting to build low service area servers.

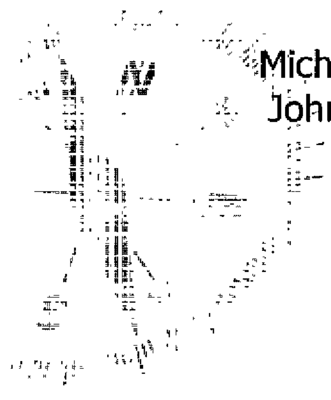
- *This allows us to take an incremental and targeted approach to re-architecting server.* As the common server foundation allows server apps we have ported to the foundation and server apps that require full server to run on the same server (modulo existing side-effects). This means we can target dev investment to port the server apps that support roles that deliver customer or competitive value. The server foundation is the platform for future server app development.

The next steps are:

- Put together a development team to build the foundation and SDK in M7. This team will need to be comprised of senior developers from around Windows, and will be in place through M7. The WELT asked us to come back with a list of candidates. Bill Laing owns staffing the team.
- Define and cost the required MinMonad and MinCLR components of the Foundation. Mark Brown and David Treadwell own this, working with Bill Laing and DavidDS's teams.

peter wilson | product unit manager | embedded windows group | microsoft corporation | 425.703.1933

Windows Server Foundation and Roles



Michael Beck, Cheryl Evans, Steve Jang, Bill Laing, John Macintyre, Rich Pletcher, Janet Smith, Peter Wilson

Agenda

- ►Problem Statement, Proposal & Benefits
- ►Proposed Server Architecture
- ►Architecture Details and Work Items
- ►Implementation Cost & Possible Schedule

Problem Statement, Proposal & Benefits

Customer Problem Statement

- Windows Server is frequently deployed to support a single, fixed workload
  - Examples: Network, Web, Security, File & Print
- In this scenario, customers are required to deploy and service all of Windows Server (& Client)
  - Required "server inappropriate" components: IE/OE, Shell, GUI, Multi-media, etc.
  - Un-used server apps (Example: IIS is installed on file server, etc.)
- These non-value add features (wrt fixed workload server) present a significant servicing & security burden
  - For Win2K Server 60% of patches were for "non-core" components
- Linux today provides a platform for building small, reliable, low maintenance servers
  - Windows does not

Proposal (50,000ft) Overview

- Architect Windows as:
  - A Server Foundation
    - MinWin, Security, Networking, Server Base
    - No GUI, Shell, IE, etc.
    - Common core binaries for client and server
  - A set of server applications that are ported to this Foundation
- Deliver to customers
  - Full client and server
  - Low surface area servers for targeted workloads
  - A platform for developing small, reliable, low maintenance servers
    - Server subset of Win32, LAPI

Server Foundation Benefits*

**for GUI-less servers composed from "foundation plus roles"*

1. Server customers install fewer patches**
2. Server is more secure, more reliable**
3. Deliver a platform for server applications (that is common subset of client)
4. Enable targeted and incremental development investment (vs. "big bang")

***compared to full server*

Server Customers Install Fewer Patches

- Today:
  - Customers must service entire server OS
    - Even features they are not using
    - Even client features that have no value to server
- Foundation & Roles:
  - Server Foundation reduces #patches by ~60% (based on Win2K data)
    - Total servicing burden is reduced to # patches for foundation plus # patches for deployed roles only
  - Servicing burden can be further reduced:
    - Explicitly defining, building and testing the server foundation allows us to better target security and reliability investments where they are most-leveraged
    - Defining the server as "foundation + roles" allows us to package patches to minimize customer impact

Server is More Secure, Reliable

- Today:
  - Windows Server is (largely) a monolith: only as secure and reliable as the weakest part
  - Security investment targets configuring rather than removing potentially insecure components
- Foundation & Roles:
  - Architecting server as "Foundation plus roles" allows:
    - Focus on driving security, reliability into the core foundation
    - Each app team to focus on driving security, reliability into their app, independent of side-effects from other server apps
    - Removal of non-value add legacy & client components from server which lower overall security, reliability

Deliver a Server Platform

- Today:
  - There is no platform smaller than full Server for server apps
  - There is no guidance for which APIs to use on server vs. full client
  - Results in server apps that are dependant on (high servicing cost) client components
    - Example: DNS, DHCP, WINS, IIS, etc. all pull in IE/OE
- Foundation & Roles:
  - Deliver a platform for building GUI-less, manageable, composable server applications
    - Platform plus SDK
    - Avoids inadvertent dependency taking
    - Allows us to be prescriptive about no-GUI, manageability, etc.
  - Deliver a common base API between client and server
    - Server-Win32 and Server-LAPI subsets
    - Maintain single implementation of shared APIs
  - Deliver an alternative to Linux for building low footprint servers
    - Evangelize Server SDK as LAPI subset for developing small, reliable, low maintenance, manageable server applications

Allow Targeted Dev Investment

- Today:
  - Server re-architecture proposals are super disruptive:
    - New base API: CLI/NT
    - Wide spread impact & regression: remove IE
- Foundation & Roles:
  - Abstract the Server Foundation platform along a "natural seam" to reduce development cost
    - Remove all GUI
    - Pragmatic balancing of cost of removing each component vs. (servicing) cost of keeping
  - Server apps can be ported to this platform based on customer-driven priority
  - Development tasks
    - Abstracting server foundation from full OS
      - Design, development, SDK development
      - Identify and break dependencies in core files
    - Extend Server Foundation with new Longhorn server features
      - Design & Implement MinCLR/Server-LAPI
      - Design & Implement Server Foundation management infrastructure
      - Deployment infrastructure
    - Target server apps that deliver customer/competitive value
      - Platform plus SDK allows this work to be owned by application teams

Proposed Server Architecture

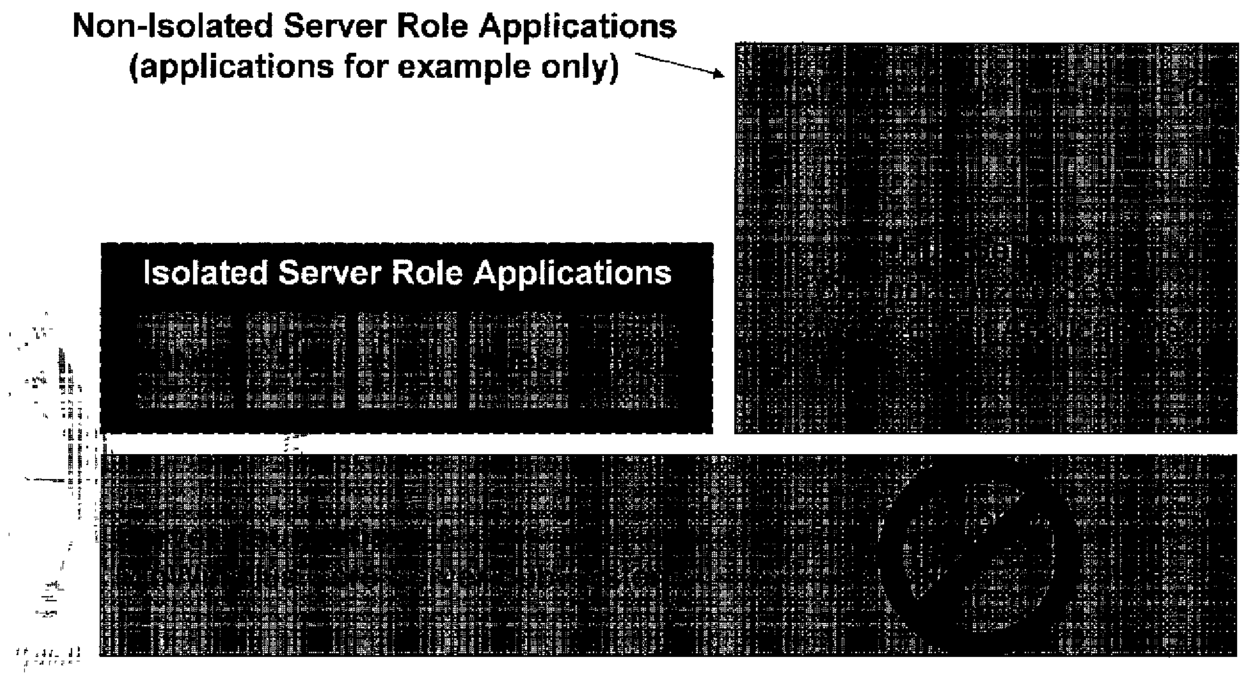
Prioritized Server Roles and Requirements

Server Type	Workload *Note Workloads in* ***bold italic*** *are first priority*	Server Role *Note Roles in* ***bold italics*** *are first priority*	Rationale for Prioritization (or lack of prioritization)	BOL for defining components
IT Infrastructure	***Directory/ Domain Controller***	***AD***	High volume/Many components can be removed (Abbrev H Vol/H Comp Remov)	Andreas Luther
		ADAM	Strategic for ISV	Andreas Luther
		UDDI	Too many components	Rosa Thomas
	Networking	***DNS***	H Vol/H Comp Remov	Kamal Janardhan
		DHCP	H Vol/H Comp Remov	Ljubomir Bradic
		WINS	High Comp Remov	TBD
	Access/VPN	*VPN*	H Vol/H Comp Remov	Elliot Lewis
		RAS	H Vol/H Comp Remov	Elliot Lewis
	Identity Mgmt	Certificate Server	Low volume	Laudon Williams
		Trustbridge	Low Volume	Matt Hur
		IAS	Low Volume	Ashwin Palekar
		AZ Man	Low Volume	Dave McPherson
		Rights Mgmt	Component # TBD	Marie Maxwell
	Streaming Media	WMS	Too many components	Rick Prologo
	Virtual Machining	***Minimum Core*** (allows VMs to run on top)	Strategic for virtual machine strategy	TBD
App Platform	***Static Web*** *(includes web services, Web UI apps and ISAPI/ASP Apps)*	***IIS, FTP, HTTP***	H Vol/H Comp Remov	Jeff Kercher
	Dynamic Web and .NET App Execution Environment	.NET Fx; COM+	Too many components (Deeper analysis underway)	Jeff Kercher
	Win 32 App Environment (without UI)	***Base + Win 32 APIs***	H Vol/H Comp Remov, Strategic for ISVs	TBD
I-Worker Infrastructure	***File***	***File***	H Vol/H Comp Remov	David Golds
	Print	***Print***	H Vol/H Comp Remov	Tali Roth
	Collaboration	WSS	Too many components	Anuraag Tiwari
		DRM	Component # TBD	T Lindeman
		RTC	Component # TBD	V Kumbalimutt
	Terminal Services	TS	Too many components	Adam Henderson
Management Infrastructure	Software Distribution	SUS	Low Volume	Joseph Dadzie
		ADS/RIS	Low Volume	Charlie Chase

Prioritized Requirements:

1. Apply Compose-ability to Complete Server Roles (See prioritized list on left) Prioritization given to:
   - High Volume Roles (based on Server Tracker data)
   - Roles which require few components (esp. freq. patched components)
   - ISV needs
   - VM Needs
2. Provide a patch and mgmt capabilities so that all LH servers are patched and managed similarly
3. Provide a componentized server w/o UI and provide SDK for ISVs.
4. Support headless administration of server roles.
5. Provide an easy UI to install, change and manage the server components
6. Support multiple roles on a single server.
7. Stretch Requirement: Provide a means for ISVs and VAR/VAPs to indicate their required components and allow auto install and configuration of those components

Proposed Server Architecture

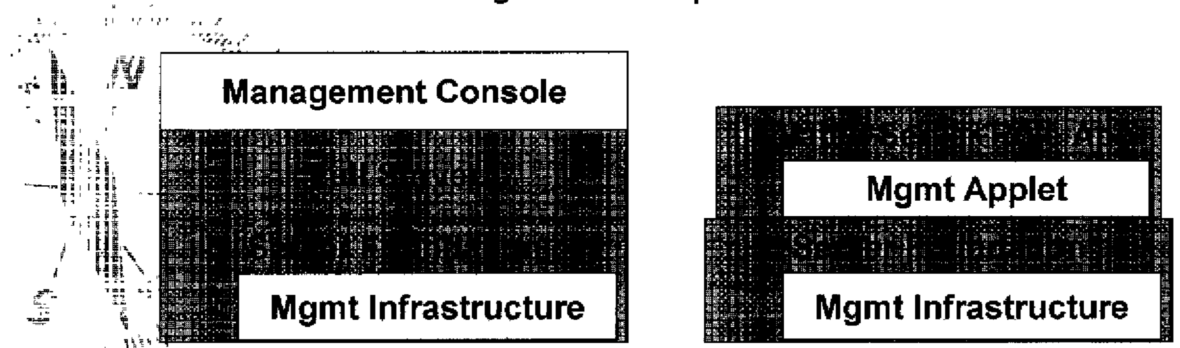

Non-Isolated Server Role Applications
(applications for example only)
Isolated Server Role Applications

Server Architecture Parts

- Server Foundation
  - Core Subset of Windows Server on which server apps are built and ported
    - Common core for isolated server roles ***and*** full server (and client...)
  - API platforms and an SDK
    - All apps and utils are roles on top of Foundation
  - Secure and reliable through "few moving parts"
    - No GUI, IE, OE, Media, Video, etc. etc.
    - Only universally required server sub-systems by design
    - Expected low servicing frequency
  - The core for long-term server evolution
- Isolated Server Role Applications
  - Server applications supporting customer visible server roles
  - Developed/ported to Server Foundation
    - No GUI, uses supported sub-systems, brings in other required libraries
  - Targeted engineering investment to customer/competitive requirements
- Full Server
  - Full server, separated from common Server Foundation
  - Allows legacy server apps and client apps install/run/be managed alongside ISRs

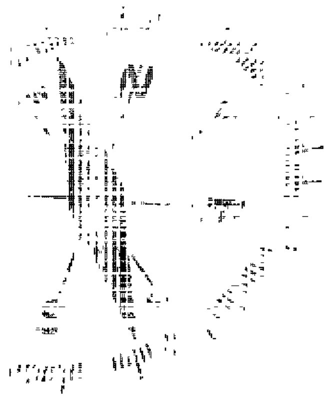
What About Management?

- Goal: design management into Server Foundation and Roles
- In this architecture management is:
  - A platform supported in the server foundation to provide management infrastructure to server apps
  - A management applet associated with each server role application
  - A management application console to support a "management server" role
- ODP Team owns building these components



Management Client/Server Role *Focused Workload Server*

Architecture Details and Work Items



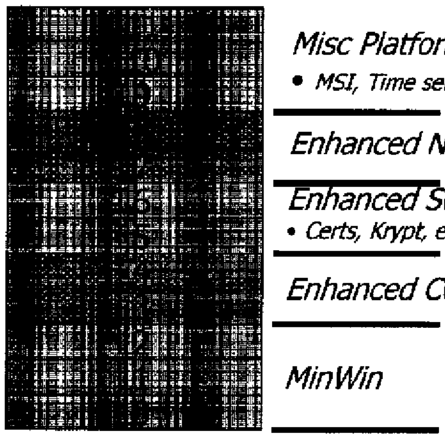
Foundation Design Approach

- Data Sources
  - BDD data from WS2K3 (Dependencies between binaries)
  - Server Role priorities and definitions from Composability MRD
  - Component layering from WinComp classification and MinWin/Foundation design
  - Bug/QFE data for Win2K and WinXP
- Design Approach
  - Identify apps that support server roles
  - Identify common binaries required for all identifies server roles/apps
  - Identify files to remove, and work items to remove these files
  - Apply layering model to remaining binaries to identify good/bad dependencies, and work items to remove dependencies
  - Identify new/missing sub-systems, and identify componentization requirements

Server Foundation Overview

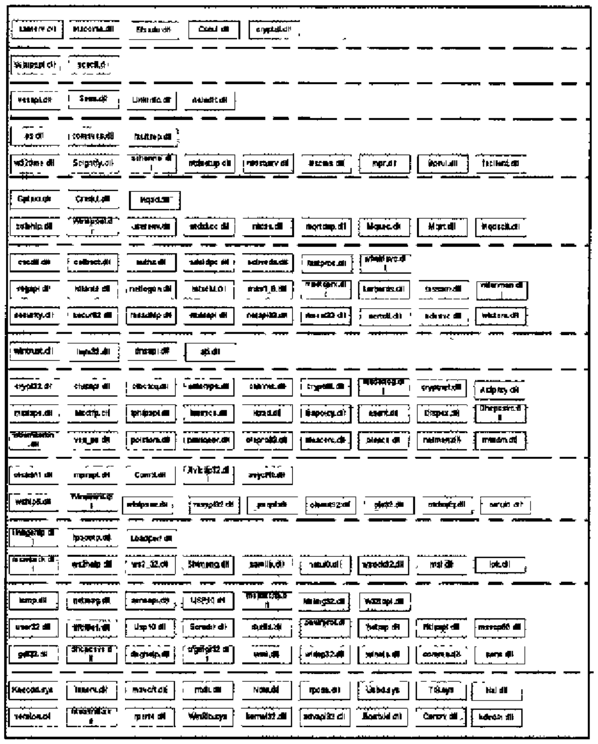
- Process
  - Initial 246 common binaries for server roles
  - Remove Shell, GUI, IE, Media etc.
  - ~160 files remain in Foundation
  - Apply layering model to remaining binaries
  - Identify dependencies to remove
- Server Foundation Overview
  - Contains
    - MinWin, Server Base Components, COM, WMI, Power Management, Networking, User/GDI
  - Removed
    - Shell, IE (OE, Browser UI etc.), Multi-media, Terminal Server, Wireless_Zero_Config, TAPI
  - Missing (need to be defined)
    - MinCLR/Framework
    - Management Infrastructure
    - Server-Win32 and Server-LAPI...

- Net net:
  - ~160 binaries
  - ~53 binaries touched
  - ~155 dependencies removed



Foundation Layer Detail

- Foundation Layering
  - Provides a framework for organizing dependencies during development
  - Identify layers using dependency analysis tools and architectural reviews (BDD, Max, Magellan)
  - Components can only depend on components in lower layers



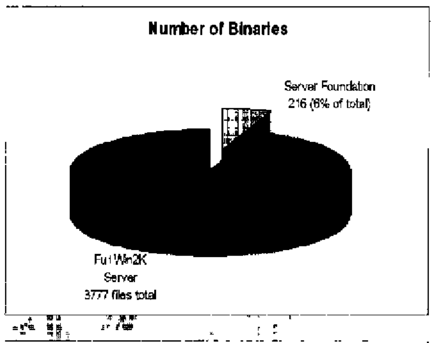
MS-CC-RN 000001068745
HIGHLY CONFIDENTIAL

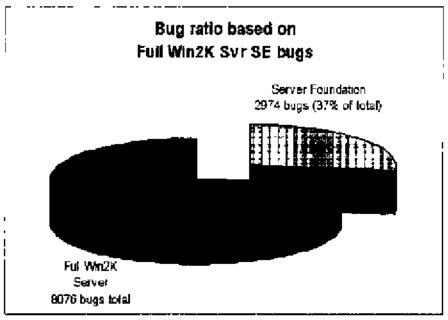
Foundation vs. Full Server

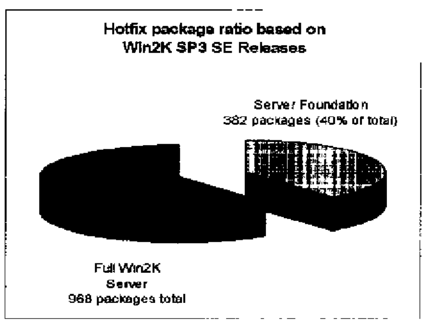
Binaries and Footprint

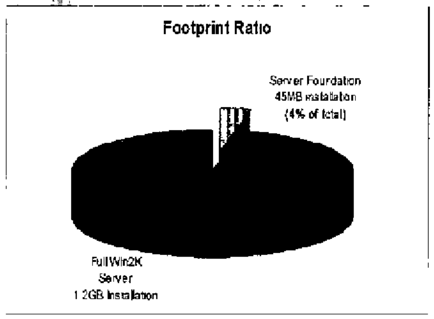
SE Bug Counts

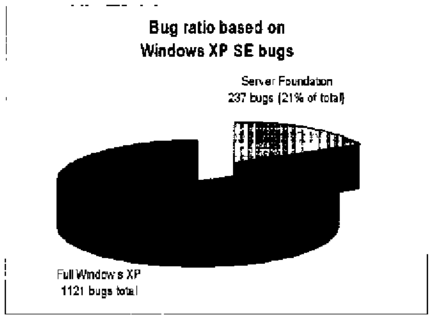
SE Hotfix Packages

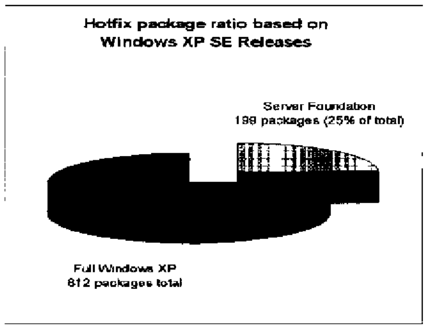












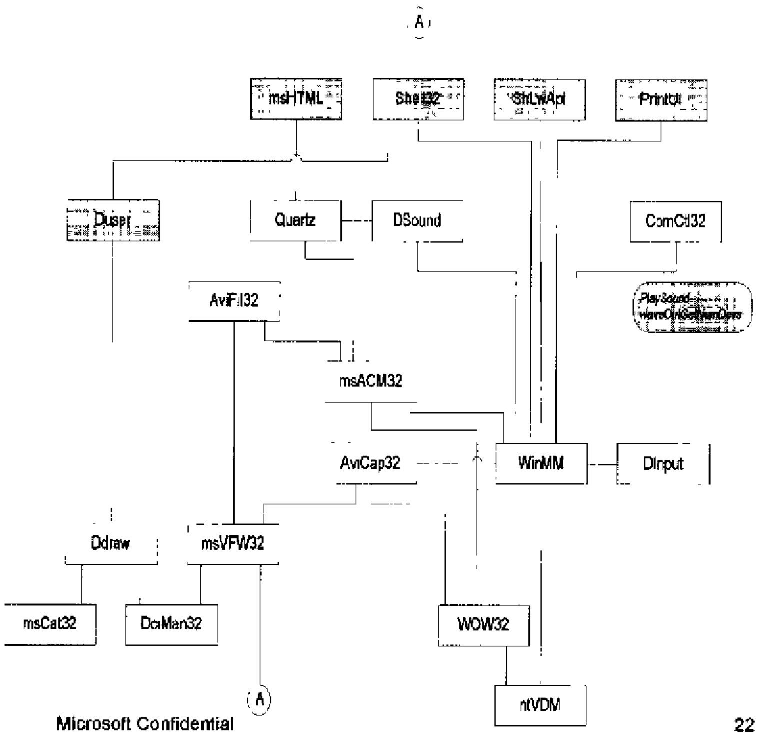
Server Roles Overview

- For each server role application
  - Identify dependencies outside of Foundation
  - Identify tasks to remove dependencies
- Demo: dependency analysis spreadsheet

- Server Roles
  - DHCP
  - DNS
  - WINS
  - File
  - Print
  - FTP
  - Web
  - App

Server Roles Case study

- Removing Shell Platform eliminates most dependencies upon Multimedia
- ComCtl32 still has a couple of delay loaded dependencies upon WinMM
- Possible Resolutions
  - Verify robustness to delay load failure when WinMM not present
  - Change code in ComCtl32 to dynamic bind to WinMM (LoadLibrary/GetProcAddress)
  - Remove "feature" of playing sound in ComCtl32
  - Only use standard ComCtl32 that does not depend upon WinMM; rather than the enhanced side by side version
- This change also removes WOW32 dependency



Remaining Work Items

- Servicing & Management
  - Define end-to-end management story for fixed workload servers
    - Remote & on-server
  - Define low surface area management infrastructure to support server roles
  - Define end-to-end servicing story for fixed role servers
  - Define low surface area SUS component for fixed workload servers
  - Define requirement of other infrastructure to support management & servicing
    - Crimson, Indigo, CLR, etc.
- Deployment
  - Define deployment mechanism for deploying and switching server roles
  - Server deployment should run on server foundation...
- Longhorn Features
  - Define Server-LAPI features that will be in server foundation
  - Factor CLR to allow Server-LAPI only to be installed on server (example: no WinForms, no Avalon UI, etc.)

Implementation Cost & Possible Schedule

Implementation Discussion

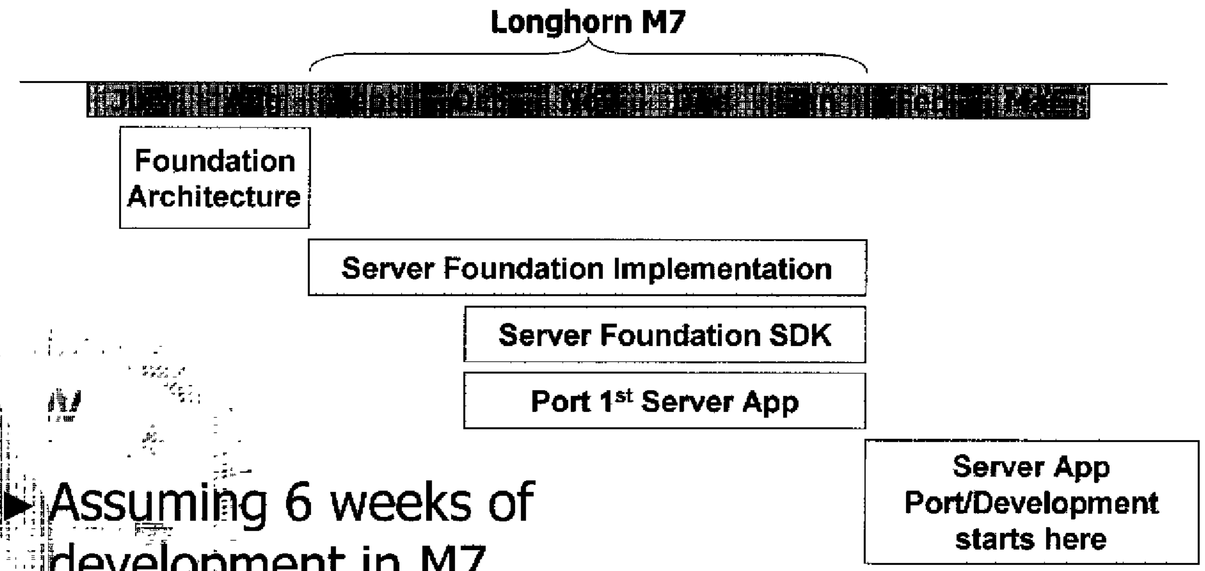
- Implementation Drivers
  - Server Foundation must be complete in M7 due to destabilization of client core components
- Critical Path Work Items
  - Server Foundation architecture design
    - Definitive foundation task list
    - Dependency removal design patterns
    - Requirements for new Longhorn Server features
  - Server Foundation Implementation
    - Server Foundation
    - Server Foundation SDK (internal vs. external)
- Non-critical Path Work Items
  - Management infrastructure for Foundation design
  - Design and task list for server role applications

Implementation Cost

- **Note: there is no team staffed and ready to do this work**
- Critical path tasks only

Task		Cost (Days)	Basis for Estimate
Server Foundation architecture design		40	SWAG (2 people, 1 month, based on MinWin experience)
Server Foundation Implementation			
	Server Foundation	300	~155 dependencies @ 2 days per. Based on historical bug fix cost
	Server Foundation SDK	20	SWAG (1 person, 1 month)
Total		**360**	

Potential Schedule



- Assuming 6 weeks of development in M7

Development Team Detail

- Architecture Design & pilot, now – 1/31/04
  - 2 architects
- Development Team 9/1/03 – 1/31/04
  - 10 senior developers from
    - COM+/OLE
    - Base
    - Management
    - Security
    - Shell
    - Networking
  - 1 developer for Server SDK
  - 10 test, 2 PM, 1 manager

- Additional Development
  - MinCLR: Owner davidtr
  - Management & Servicing: Owner chuckc
  - Deployment: Owner DavidDS

Cost & Schedule Issues

- There is no team staffed and ready
  - Requires core OS technical dev skills
  - Requires deep engagement with Longhorn server new feature teams
- Basis for estimation
  - Requires detailed design and prototyping to confirm estimates
  - Based on BDD data to identify dependencies: requires verification
- Server component creation for component based build is also happening at this time