Lights-Out Automation
for a Small MCP Shop

Paul Kimpel

2011 UNITE Conference
Session MCP-4001

Tuesday, 24 May 2011, 2:45 p.m.

Copyright © 2011, All Rights Reserved

Paradigm Corporation

Lights-Out Automation for a Small MCP Shop

2011 UNITE Conference
Garden Grove, California

Session MCP-4001

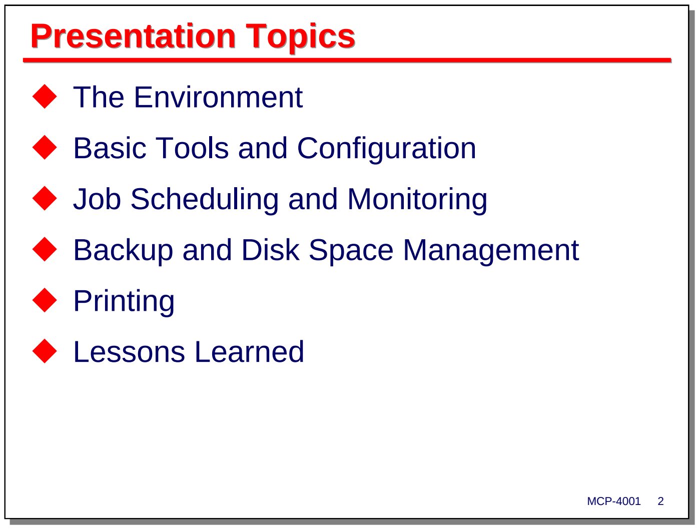
Tuesday, 24 May 2011, 2:45 p.m.

Paul Kimpel

Paradigm Corporation
San Diego, California

http://www.digm.com

e-mail: paul.kimpel@digm.com



Presentation Topics

- The Environment
- Basic Tools and Configuration
- Job Scheduling and Monitoring
- Backup and Disk Space Management
- Printing
- Lessons Learned

MCP-4001 2

I have been supporting a small MCP operation for a number of years, since it upgraded from A Series to ClearPath. Over that period of time I have been working with their applications programmer to improve the reliability of the overall operation, and to reduce the amount of time the programmer and I need to spend monitoring the system and doing repetitive administration tasks.

In the past couple of years, we've achieved a fairly comfortable position in terms of daily operations and support of that system. This presentation discusses how we have the system operating today, and how we have tried to set it up for unattended operation. This presentation also discusses how we have used simple tools to instrument the system so that when things don't work the way they should, we find out about it in a timely manner.

I will begin talking about the environment within which this system operates. I will also discuss the basic tools we use for administering and monitoring the system, and some highlights of the hardware and MCP configuration.

Next, I will discuss the simple batch job scheduling and monitoring system we've developed, and how we have instrumented it for remote support.

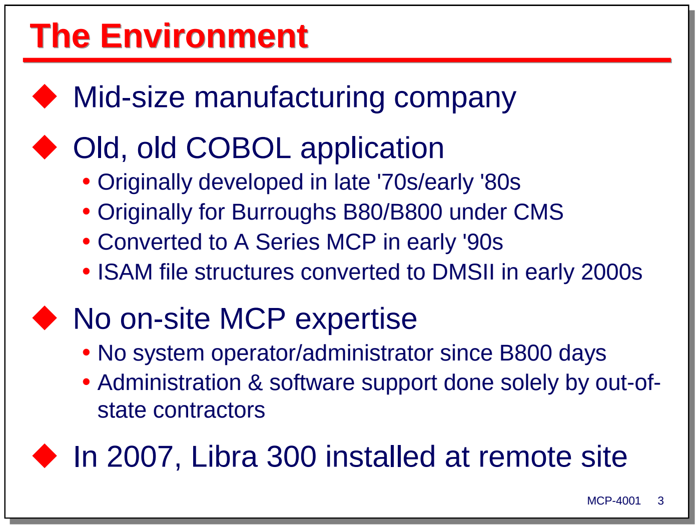
Perhaps the most important administrative responsibility is backup and space management for the MCP disk subsystem. I will discuss how we do backup and how we monitor it for successful completion.

The last major topic is printing. I will discuss how we have configured printing for minimal administrator involvement and how using standard Unisys facilities has made the process easier for everyone – users and administrators alike.

I'll close the presentation with a brief discussion of the lessons we've learned over the years and what works well.

The approach described here is very minimalist. We have tried to take maximum advantage of a minimum set of tools and resources, and tailor the result to this specific situation. This is just one approach that could have been taken to address the administration and support issues at this site, but I think it is interesting how far you can go using standard MCP facilities and a small amount of home-grown software. Like most things that work well, it's composed of at most 20% technology and at least 80% technique.

Accompanying this presentation are a set of files containing all of the tools, utilities, and WFLs we currently use at this shop for remote administration and monitoring of the MCP environment. These may or may not be useful in your environment, but you are welcome to mine these materials for ideas and techniques, and to adapt them in any way that you may find useful.



The Environment

- Mid-size manufacturing company
- Old, old COBOL application
  - Originally developed in late '70s/early '80s
  - Originally for Burroughs B80/B800 under CMS
  - Converted to A Series MCP in early '90s
  - ISAM file structures converted to DMSII in early 2000s
- No on-site MCP expertise
  - No system operator/administrator since B800 days
  - Administration & software support done solely by out-of-state contractors
- In 2007, Libra 300 installed at remote site

MCP-4001 3

The environment within which the MCP system operates is a mid-size manufacturing company. They buy stuff and make other stuff out of it. They have the usual business applications, G/L, purchasing, order entry, production control and scheduling, inventory management, shipping, invoicing, and customer relationship management. Originally all of this ran on the MCP. As time has passed, some functions have migrated out to other systems, some purchased and some custom built, but the MCP-based application remains at the core of operations for the enterprise.

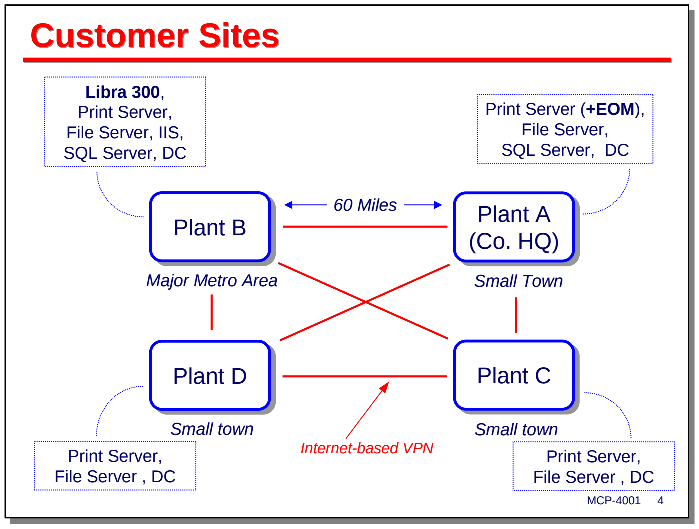
The MCP-based application is really old. It is mostly COBOL-74, with some newer COBOL-85 pieces, but everything is now compiled with COBOL-85. It was originally written in the late 1970s and early '80s for Burroughs B80/B800 systems under the CMS operating system. After support for the B800 ended, the application was converted to A Series MCP in the early '90s and ran on Micro-A and A7 boxes. The application was originally a package with multiple customers, but this site is the only one left, and the software has been extensively customized for their business.

The original application was based entirely on ISAM and "flat" files, which were retained through the A Series migration until the early 2000s. At that point, most of the application data was converted to DMSII.

The customer has not had any on-site operations staff since the early B800 days, and has never had any on-site CMS or MCP administrative staff. Until the early 2000s, they didn't even have a full-time network administrator. The application has been supported for more than 20 years by a member of the original development team, who lives and works more than 500 miles away. I became involved during the DMSII conversion, and have been the de facto MCP administrator since. I live and work about 2,000 miles away.

Until 2007, the B800 and MCP boxes had been located at the company headquarters, which are in a small town, too far away from a major metropolitan area to get same-day support from Unisys. In 2007, the customer upgraded to a Libra 300, which they decided to place in a new production facility about 60 miles from headquarters, but close enough to a major city that they could get same-day response from Unisys. The new facility also had a good server room and better Internet access than was available in the headquarters town, which were further reasons to locate the Libra remotely.

Thus began our journey towards completely hands-off, lights-out, remote operation and administration of this MCP system and its applications.



Customer Sites

Libra 300, Print Server, File Server, IIS, SQL Server, DC
Print Server (**+EOM**), File Server, SQL Server, DC
Plant B
60 Miles
Plant A (Co. HQ)
Major Metro Area
Small Town
Plant D
Plant C
Small town
Small town
Internet-based VPN
Print Server, File Server , DC
Print Server, File Server , DC
MCP-4001 4

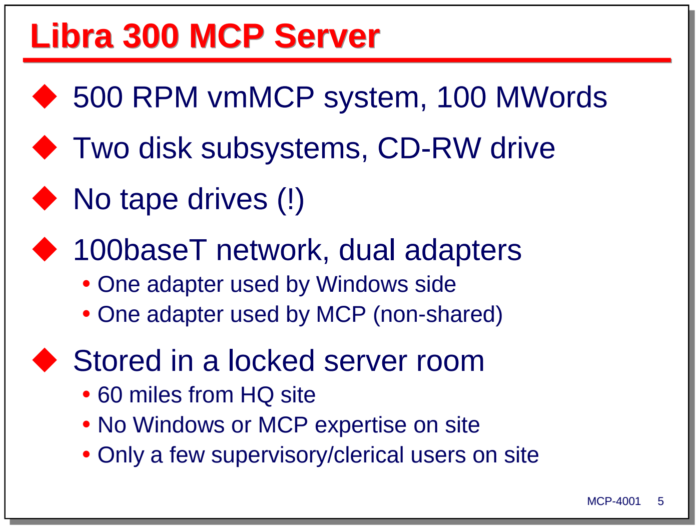
This slide shows a rough schematic of the customer locations that access the MCP system.

Plant A is the company headquarters. It no longer hosts the MCP system, but does have a print server, file server, Microsoft SQL Server box, domain controller, and a few other Windows-based servers. We are running Enterprise Output Manager on the print server, which I'll discuss in more detail later.

Plant B has a server configuration similar to Plant A's, plus an IIS box and the Unisys Libra 300. It is located in a major metropolitan area about 60 miles from the company headquarters at Plant A. As mentioned earlier, the Libra was located here primarily because we could get same-day service from Unisys (which, thankfully, has not been necessary) and because Plant B had good infrastructure, including a modern server room.

Plants C and D have more limited server environments, providing only what is needed to support their local users. They are much farther away – hundreds of miles from both Plants A and B and from each other.

All of the facilities are linked by router-based VPNs over standard Internet circuits. Every location can access the internal network of every other location. There is a 5 megabit/second link between Plant A and B to support the relatively large number of users at company HQ accessing the servers located at Plant B. The virtual tape server for the Windows-based backup system is located at Plant A, so there is a lot of network activity at night, backing up the servers at Plant B over the wire to the virtual tape server at Plant A.



Libra 300 MCP Server

- 500 RPM vmMCP system, 100 MWords
- Two disk subsystems, CD-RW drive
- No tape drives (!)
- 100baseT network, dual adapters
  - One adapter used by Windows side
  - One adapter used by MCP (non-shared)
- Stored in a locked server room
  - 60 miles from HQ site
  - No Windows or MCP expertise on site
  - Only a few supervisory/clerical users on site

MCP-4001 5

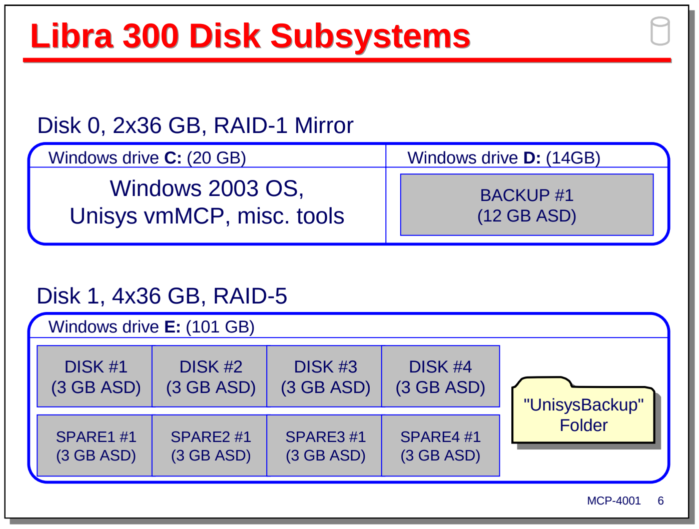
The Libra 300 MCP server at Plant B is a modest affair. Physically, it is a standard Windows server, running Windows 2003, with the emulated MCP virtual machine (vmMCP) running on top of that. At the time it was acquired, it was the smallest, lowest-capacity, non-SDK MCP system available. It is licensed for 500 RPM (20 MIPS) and is currently configured for 100 megawords of MCP memory.

The Libra has two disk subsystems, which will be described on the next slide. It has the standard DVD-R/CD-RW drive that comes with such systems. That's it. There are no physical tape drives on the system. I'll spend time discussing how we do backup later in this presentation.

The Libra has dual network adapters. We decided to devote one of those to the Windows side of the box, and the other to the MCP as a Network Services (NX/Net) non-shared adapter.

There is no MCP operations or administration support on-site at Plant B. In fact, there isn't even any Windows or network support on site. The closest internal support is the network administrator, who normally works at Plant A, 60 miles and an hour's drive away.

The Libra 300, along with the rest of the Windows servers and the network routers and switches, sits in locked room, completely unattended. It is quite literally a lights-out environment. The network administrator visits as needed, typically every week or two, for network or Windows server issues. He knows to keep his hands off the Unisys box.



The Libra 300 has two physical disk subsystems, which we have divided into three NTFS partitions.

Disk 0 is a RAID-1 mirror composed of two 36GB drives. It is configured as:

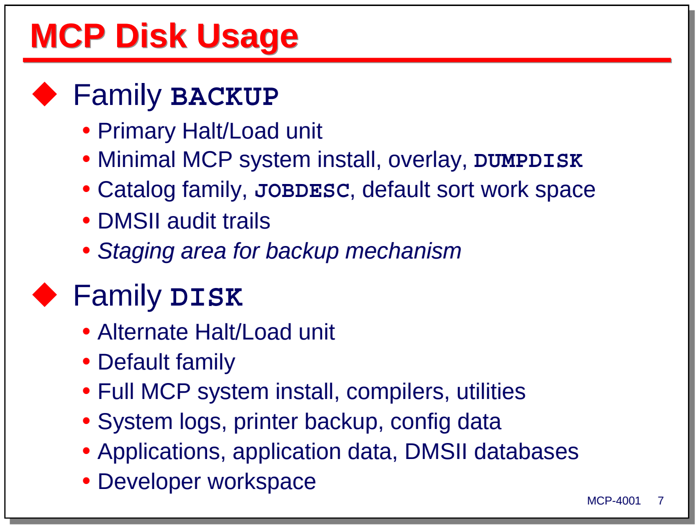
- A 20GB partition as drive C, which holds the Windows 2003 software, the Unisys vmMCP software, and the usual set of miscellaneous tools and utilities you would find on a system like this.
- A 14GB partition as drive D, which holds a single 12GB MCP Logical Disk (`.asd`) file, which the MCP environment knows as family `BACKUP`.

Disk 1 is a RAID-5 array composed of four 36GB drives, yielding about 101 GB of usable space. It is configured in a single partition as drive E. This resource is primarily used for:

- Four 3GB MCP Logical Disk files, which together are known to the MCP environment as family `DISK`.
- An additional four 3GB MCP Logical Disk files, which are individually known to the MCP as single-unit families `SPARE1` through `SPARE4`. These families are not actively used. They exist to reserve space for additional MCP disk capacity, should that be needed.
- A folder in the Windows file system known as "`UnisysBackup`". This folder is associated with a network share of the same name, and is used primarily by the backup mechanism, as will be discussed shortly.

The two MCP families that are actively used are `DISK` and `BACKUP`. Note that each has a total capacity of 12GB, but `DISK` is configured as four 3GB logical disk units, while `BACKUP` is a single 12GB logical disk unit. This is intentional, and the reason has to do with the way that Windows handles disk I/O and caching for RAID vs. non-RAID disks. Unisys recommends that:

- For Logical Disks on mirrored or non-RAID devices, the MCP family member be configured as one `.asd` file.
- For Logical Disks on RAID-5 and higher subsystems, the MCP family members be configured as multiple `.asd` files.



MCP Disk Usage

- Family **BACKUP**
  - Primary Halt/Load unit
  - Minimal MCP system install, overlay, **DUMPDISK**
  - Catalog family, **JOBDESC**, default sort work space
  - DMSII audit trails
  - *Staging area for backup mechanism*
- Family **DISK**
  - Alternate Halt/Load unit
  - Default family
  - Full MCP system install, compilers, utilities
  - System logs, printer backup, config data
  - Applications, application data, DMSII databases
  - Developer workspace

MCP-4001 7

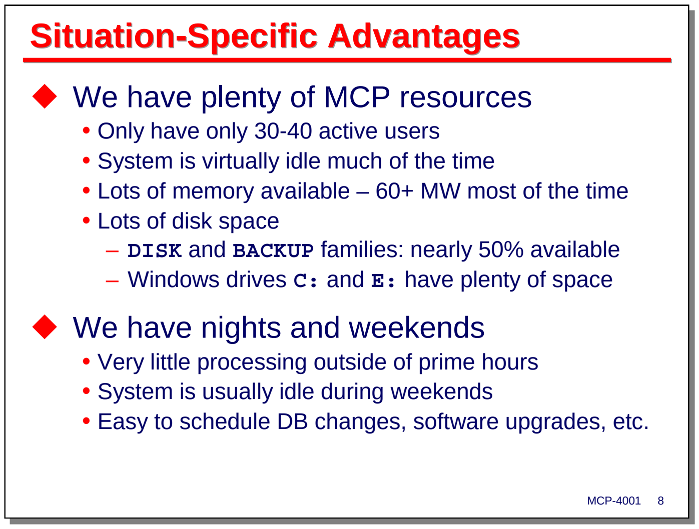
We gave quite a bit of thought to how we should distribute files across the two active MCP disk families, **DISK** and **BACKUP**. We tried to take into account the need to balance I/Os across the two physical subsystems, balance disk space usage between the families, separate data and audit I/Os for DMSII, and provide for a recoverable configuration in case of physical disk problems. We also had to take into account that the application had been built and operated for years assuming only one family, **DISK**.

In the scheme we finally implemented, family **BACKUP** is the halt/load unit, but it contains only a minimal set of MCP system files – just enough to get the system up and running so we can use the ODTs and Library/Maintenance to begin recovery and restoration should something happen to family **DISK**. This family also has DL assignments to handle overlay, **DUMPDISK**, catalog, **JOBDESC**, and sort work space. Since the DMSII databases would be on family **DISK**, this family has the audit trail files.

All of these functions require only a modest amount of space – about 1.5 GB (with **DUMPDISK** taking a gigabyte by itself). The bulk of the space on family **BACKUP** is reserved for use by the backup mechanism, as will be described later.

Family **DISK** contains everything else. It is an alternate halt/load unit, and is of course the default family for the system. It contains a full set MCP software, including the compilers, utilities, libraries, etc. If family **BACKUP** fails we can boot from DISK to start recovery and restoration. Thus, if either disk family fails, we should still be able to get at least a basic system up and operating fairly quickly.

Family disk also has the system log files, printer backup, and all of the configuration data for the MCP services. It has the DMSII databases, non-database application files, and application source and object code. It also has the workspace for the two people who support the system and applications.



Situation-Specific Advantages

- We have plenty of MCP resources
  - Only have only 30-40 active users
  - System is virtually idle much of the time
  - Lots of memory available – 60+ MW most of the time
  - Lots of disk space
    - **`DISK`** and **`BACKUP`** families: nearly 50% available
    - Windows drives **`C:`** and **`E:`** have plenty of space
- We have nights and weekends
  - Very little processing outside of prime hours
  - System is usually idle during weekends
  - Easy to schedule DB changes, software upgrades, etc.

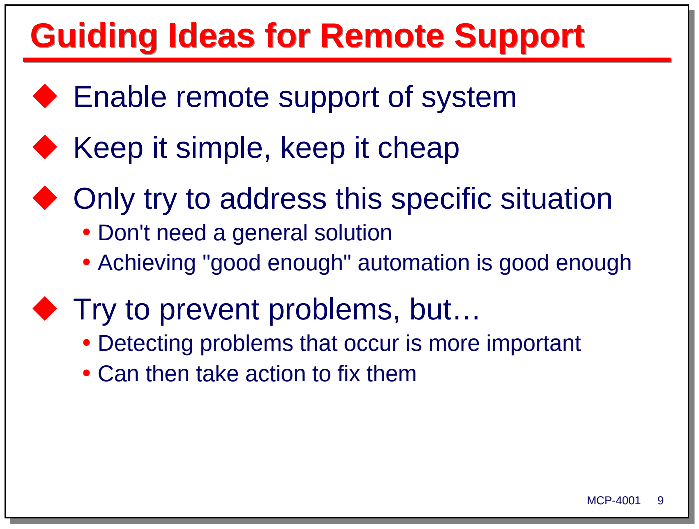
MCP-4001 8

There are some characteristics of this environment that are advantageous, and we have tried to leverage them as much as possible in our approach to unattended operation and remote support.

The first advantage is that we have plenty of MCP resources. We try not to waste them, but we have at most 30-40 users at any one time, we are swimming in available memory, and much of the time the processor utilization is relatively low. In addition to lots of MCP available memory, Windows has over 3GB of memory, so it is able to do a lot of disk caching.

We have lots of disk space for both the MCP and Windows. The two MCP families typically run near 50% available, and Windows drives **`C:`** and **`E:`** have plenty of unused space.

We also have lots of time available for support. The bulk of the system usage takes place during weekdays from 7:00 a.m. until about 10:00 p.m. There are some background activities that run outside that window, but those can usually be skipped or deferred. Very little happens on the weekends, so it is easy to get time for database reorganizations, software upgrades, and the other events that complicate an administrator's life in environments that require almost constant uptime.



Guiding Ideas for Remote Support

- Enable remote support of system
- Keep it simple, keep it cheap
- Only try to address this specific situation
  - Don't need a general solution
  - Achieving "good enough" automation is good enough
- Try to prevent problems, but…
  - Detecting problems that occur is more important
  - Can then take action to fix them

MCP-4001 9

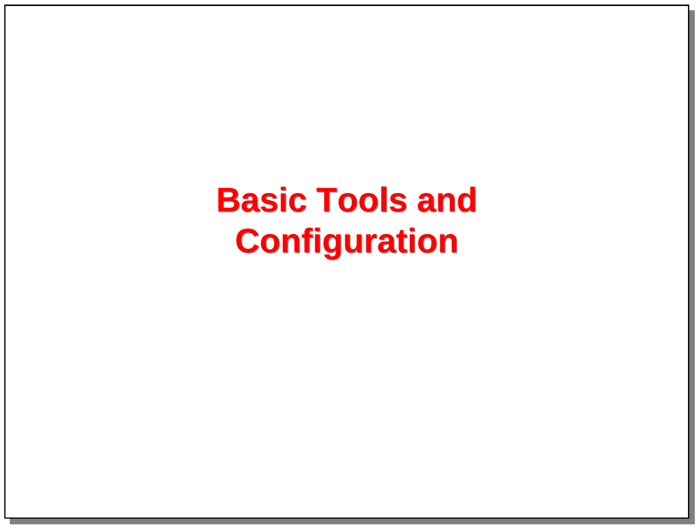
The small size of the system along with the resource and schedule flexibility of the environment have given us some guiding ideas for supporting unattended operation and remote administration of the system.

First, we don't have a choice – unattended MCP operation in this environment is a given, because there's no one there to do it, and there's never going to be. The customer is not going to find anyone who is local to support an MCP environment, so remote support is the only option.

Second, we need to keep it simple. The customer does not want to spend any more money on its IT operation than it can get away with, so we need to try to leverage existing tools and resources wherever we can.

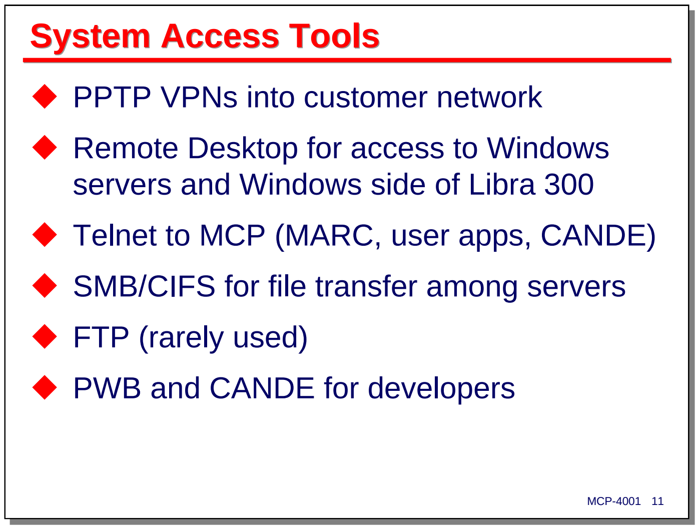
Third, we only need to address *this specific situation*. Our solutions probably won't work quite right for anyone else, but that is not something we need to worry about. We’re after a "good enough" solution.

Finally, while we'd like to prevent problems from happening wherever we can, we don't want to spend a lot of time and effort (and money) trying to forestall rare or unlikely events. It's really much more important that when problems happen, *we find out about them*. Then we try to fix them so they won't happen again. One of the sad lessons we've learned is that things change in ways you can't see when attempting to support a site remotely, and you can't prevent stuff you can't anticipate. We have found it much more worthwhile to put our investment in problem detection than in problem prevention.



Basic Tools and Configuration

With that background, I will next discuss the basic tools we use to administer and support this site, and some of the basic system configuration that helps us to do that.



System Access Tools

- PPTP VPNs into customer network
- Remote Desktop for access to Windows servers and Windows side of Libra 300
- Telnet to MCP (MARC, user apps, CANDE)
- SMB/CIFS for file transfer among servers
- FTP (rarely used)
- PWB and CANDE for developers

MCP-4001 11

We use a number of standard tools to access the system, both for remote administration and support, and for day-to-day operations.

The customer's network administrator has set up PPTP VPN capabilities in its routers for Plants A and B. These give us secure remote access to the local networks at those locations. These VPNs are the critical resource we need to do both support and development remotely. I usually establish a VPN to Plant B (where the MCP server resides) and then configure a route in my Windows client to access resources in Plant A by going through the Plant B network.

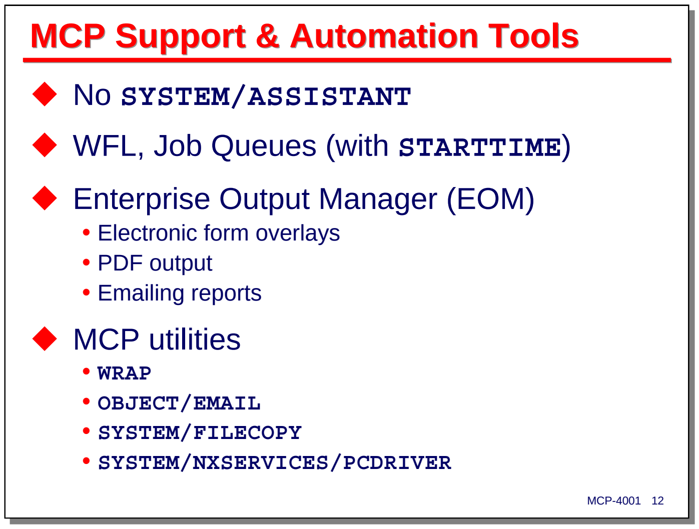
For access to the Windows servers and the Windows side of the Libra, we simply use the native Windows Remote Desktop facility. Unless you license additional Microsoft Terminal Server seats, you get only two simultaneous Remote Desktop connections to a server, so we try to keep our sessions short and disconnect when we are not actively accessing the Windows server.

Most of our MCP access is through Telnet and a standard terminal emulator. We use this extensively for development and testing, but you can perform almost all MCP administration functions through MARC if you have SYSTEMUSER privileges. Some people prefer the MARC menus, but I generally use ODT-style commands on the Action line.

We use the SMB/CIFS (Windows networking) capabilities of the MCP extensively for file transfers. I occasionally use FTP, but its use has fallen off significantly in the past several years, primarily because none of the Windows servers are currently configured to run an FTP server process.

Finally, we use PWB and CANDE extensively for both development and support of the system. We use PWB for the bulk of our source code development and maintenance. We also use SYSTEM/PATCH, and have a simple but effective source code archiving mechanism in place based on PWB-generated patch files, WFL compile jobs, and some WFL jobs that handle the migration of programs from development into production.

I especially like PWB for editing the system configuration files that are non-usercoded. With a privileged usercode, you can directly edit and create non-usercoded files from PWB.



MCP Support & Automation Tools

- No **SYSTEM/ASSISTANT**
- WFL, Job Queues (with **STARTTIME**)
- Enterprise Output Manager (EOM)
  - Electronic form overlays
  - PDF output
  - Emailing reports
- MCP utilities
  - **WRAP**
  - **OBJECT/EMAIL**
  - **SYSTEM/FILECOPY**
  - **SYSTEM/NXSERVICES/PCDRIVER**

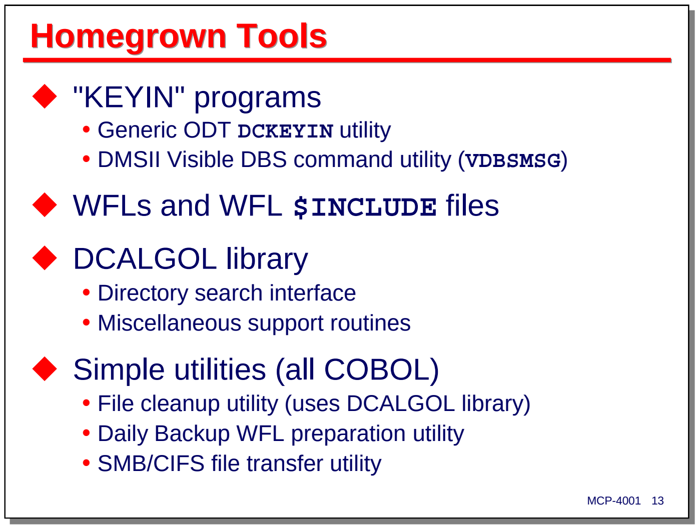
MCP-4001 12

This customer did not license **SYSTEM/ASSISTANT**, so none of our unattended or automated operations are based on that tool's capabilities. It would be nice to have, but we are getting along fine without it.

We have managed to make very effective use of WFL and the job queue mechanism. Our simple batch scheduling facility, discussed later, is built entirely using WFL and the **STARTTIME** attribute.

One of the most significant automation tools has turned out to be Enterprise Output Manager (EOM). We have used this extensively to implement electronic form overlays for laser printers so we could eliminate the use of multi-part preprinted forms and impact printers. That transition alone resulted in a significant easing of our day-to-day support burden. We are also using EOM to generate PDF output and to email some reports. I will discuss the role of EOM in more detail later in this presentation in the section on printing.

There are a few MCP utilities that we use extensively in our approach to support and automation. One of the few additions to the standard IOE the customer has licensed is **WRAP**, the role of which I will discuss shortly. We use **OBJECT/EMAIL** extensively to send success/failure messages from batch jobs and the batch scheduling mechanism. This has become a significant tool for our ability to remotely support the system and to jump on problems before they get out of hand. **SYSTEM/FILECOPY** is used in the backup mechanism I'll describe shortly. Finally, **SYSTEM/NXSERVICES/PCDRIVER** is used extensively as a file transfer tool. We are not currently taking advantage of its ability to communicate with the Windows-based Launcher utility, however.



Homegrown Tools

- "KEYIN" programs
  - Generic ODT **`DCKEYIN`** utility
  - DMSII Visible DBS command utility (**`VDBSMSG`**)
- WFLs and WFL **`$INCLUDE`** files
- DCALGOL library
  - Directory search interface
  - Miscellaneous support routines
- Simple utilities (all COBOL)
  - File cleanup utility (uses DCALGOL library)
  - Daily Backup WFL preparation utility
  - SMB/CIFS file transfer utility

MCP-4001 13

In addition to those standard MCP tools, there are a few home-grown utilities we have developed over the years and applied to our automation and remote support issues.

The first of these is the ubiquitous "**`KEYIN`**" program, which we use to submit ODT commands from batch jobs through the DCALGOL **`DCKEYIN`** API. This is used primarily by our batch scheduling mechanism.

We also have an equivalent program, **`VDBSMSG`**, for submitting DMSII Visible DBS commands from a batch program. This is used by the backup mechanism to close the current DMSII audit trail after a database backup, and in the WFL jobs we use for database updates and reorgs. This program uses the DMALGOL **`DMINQ`** API, and must be tailored for each database, but that is just a one-line change in the source code.

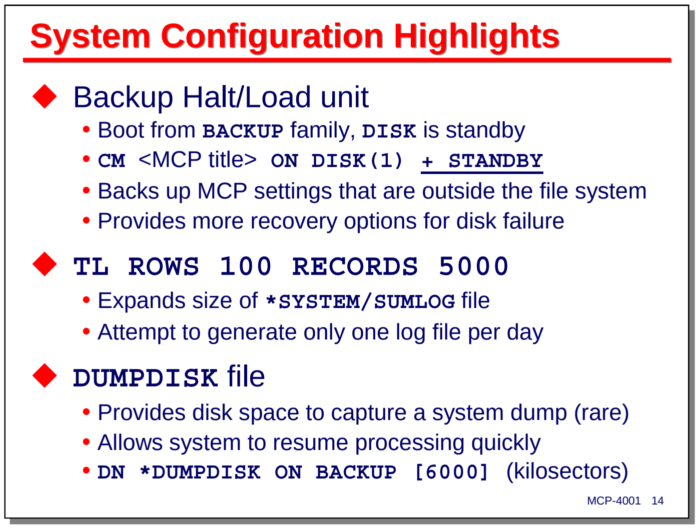
We have a number of WFLs and WFL **`$INCLUDE`** files that we use for daily scheduling, backup, and other administrative functions. The role of these will be discussed as we go through the presentation.

I wrote a DCALGOL library a number of years ago to provide utility functions for COBOL programs. The most significant of these for the purpose of this discussion is a procedure that will search an MCP disk directory and return the names of the files therein. It does not support wildcards, but the procedure is restartable so that the contents of large directories can be retrieved by making a series of calls on the procedure. This capability forms the core of our disk file cleanup process, also discussed later.

Finally, there are three relatively simple utilities, all written in COBOL:

- A program to search specified directories for obsolete files and delete them. This program uses the DCALGOL library to do the disk search and file removal.
- A program to prepare WFL job files for the backup process.
- A SMB/CIFS (Redirector) program to convert MCP fixed-length record files to stream format and transfer them to a network share. This program does special trimming of control characters from the ends of records, so we use it instead of **`SYSTEM/NXSERVICES/PCDRIVER`** as necessary for those files that require this special treatment.

The source code for all of these utilities is included in the materials accompanying this presentation.



System Configuration Highlights

- Backup Halt/Load unit
  - Boot from **BACKUP** family, **DISK** is standby
  - **CM** <MCP title> **ON DISK(1)** **+ STANDBY**
  - Backs up MCP settings that are outside the file system
  - Provides more recovery options for disk failure
- **TL ROWS 100 RECORDS 5000**
  - Expands size of ***SYSTEM/SUMLOG** file
  - Attempt to generate only one log file per day
- **DUMPDISK** file
  - Provides disk space to capture a system dump (rare)
  - Allows system to resume processing quickly
  - **DN *DUMPDISK ON BACKUP [6000]** (kilosectors)

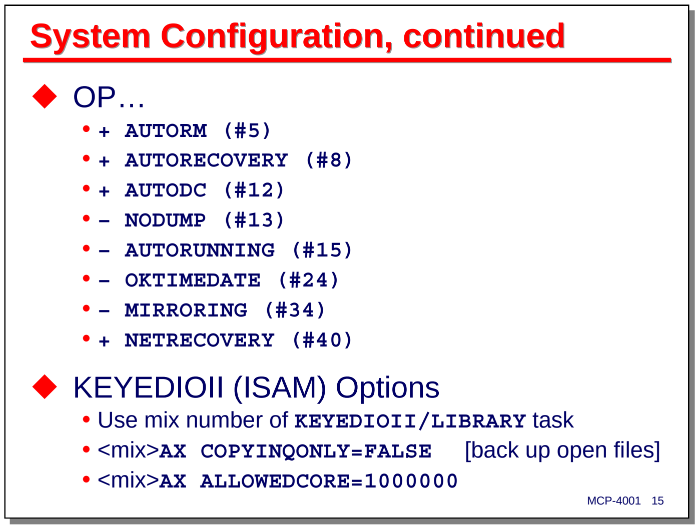
MCP-4001 14

There is nothing exotic about the software installation and configuration of the MCP environment for this system. The only things the customer has licensed beyond the standard IOE is **WRAP** and the COBOL-85 compiler. There are a few highlights, however, that are worth pointing out.

First, recalling that the **BACKUP** family is the primary halt/load unit and **DISK** is an alternate halt/load unit, we also have **DISK** configured as a "standby" family. What this means is that the MCP automatically maintains a copy of its internal settings on another family. By "internal" settings, I mean the values from commands such as **OP**, **SYSOPS**, **AI**, **DL**, **SF**, and so forth. These are stored on the halt/load unit outside the MCP file system, and therefore cannot be backed up by normal means. Having a standby family assures that we will always have a copy of these settings if we should lose the primary halt/load unit. The slide shows the **CM** command syntax that enables the standby capability.

We use the system SUMLOG quite a bit for abort reporting and problem diagnosis. It is convenient if we can have a single log for each day, so we have increased the size of the log file. As the slide shows, the **ROWS** and **RECORDS** options of the **TL** command effectively set the **AREAS** and **AREASIZE** attributes of the log file. The settings shown here yield for us a single log file each day, except in very rare cases where a runaway program dumps large numbers of **DISPLAY** messages into the log before we can catch it.

We have configured about a gigabyte of disk space on the **BACKUP** family for use as a **DUMPDISK** file. This is an area of disk the MCP's **MEMDUMP** mechanism can use to write system dumps. After the system returns to normal operation, the memory dump data can be pulled from this file and converted into a form that **SYSTEM/DUMPANALYZER** can use. We seldom see a memory dump on this system, but when we do, the **DUMPDISK** facility allows the system to take the dump quickly and then return to operational status (either by resuming or halt/loading) without requiring any operator intervention. The slide shows how we have used the **DN** command to configure this file.



System Configuration, continued

- OP…
  - `+ AUTORM (#5)`
  - `+ AUTORECOVERY (#8)`
  - `+ AUTODC (#12)`
  - `- NODUMP (#13)`
  - `- AUTORUNNING (#15)`
  - `- OKTIMEDATE (#24)`
  - `- MIRRORING (#34)`
  - `+ NETRECOVERY (#40)`
- KEYEDIOII (ISAM) Options
  - Use mix number of **KEYEDIOII/LIBRARY** task
  - <mix>**AX COPYINQONLY=FALSE** [back up open files]
  - <mix>**AX ALLOWEDCORE=1000000**

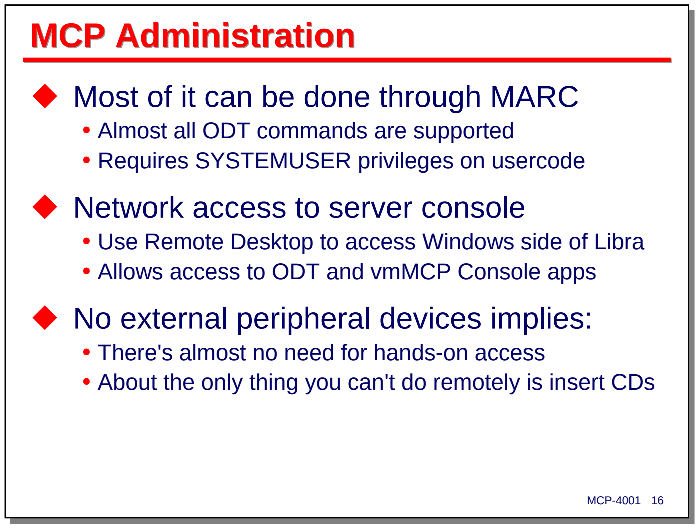
MCP-4001 15

Several of the **OP** command options are significant for unattended operation. Most of these should be obvious; I will discuss a couple of them in later sections of the presentation where they are relevant.

The only **SYSOPS** setting of note is **QUEUEDAX SET**. We do not use this feature very much, but it is useful for signaling on-line or long-running programs by means of **AX** commands.

There are two KEYEDIOII options of note. You set these using an **AX** command with the mix number of the **KEYEDIOII/LIBRARY** task:

- **COPYINQONLY=FALSE** allows Library/Maintenance to copy an ISAM file while it is open for update. While this can result in a copy of the file that is corrupted due to non-flushed memory buffers, the alternative is to have Library/Maintenance hang with an RSVP and wait for someone to give it an **OK**, **OF**, or **DS** – which in any case means the file won't get backed up at all, unless is happens to get closed while the RSVP was waiting. We chose this setting as the lesser of two evils for unattended backup operations.
- Since we have a lot of available memory, we gave KEYEDIOII some of it by means of the **ALLOWEDCORE** option. The default is only 50,000 words.



MCP Administration

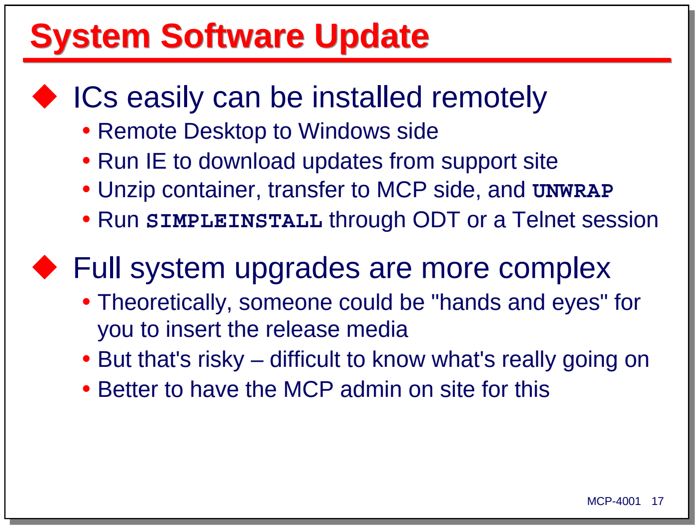
- Most of it can be done through MARC
  - Almost all ODT commands are supported
  - Requires SYSTEMUSER privileges on usercode
- Network access to server console
  - Use Remote Desktop to access Windows side of Libra
  - Allows access to ODT and vmMCP Console apps
- No external peripheral devices implies:
  - There's almost no need for hands-on access
  - About the only thing you can't do remotely is insert CDs

MCP-4001 16

In terms of administering the MCP, I mentioned earlier that you can enter almost all of the ODT commands from the Action line of a MARC screen if you run under a usercode with SYSTEMUSER privileges. Therefore, we are able to do essentially everything you would normally do through the ODT by means of a terminal emulator, Telnet, and MARC. We use the real ODT interface only rarely.

Since we are able to establish a PPTP VPN connection into the local network of Plant B, we can use Windows Remote Desktop to access the Windows side of the Libra whenever necessary to run the ODT interface. We can also use this mechanism to access the Console for MCP application. Again, we need to do this only rarely.

Since this system has no external peripherals (except the CD drive), that means we have almost no need for hands-on access to the physical box. Just about the only thing we can't do remotely is insert a CD into the system's drive. The only reason we typically need to do this is MCP software upgrade, which is the next topic.



System Software Update

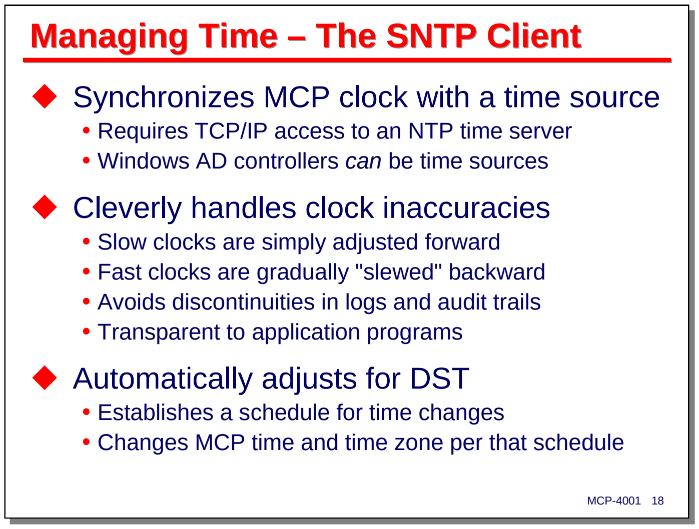
- ICs easily can be installed remotely
  - Remote Desktop to Windows side
  - Run IE to download updates from support site
  - Unzip container, transfer to MCP side, and **`UNWRAP`**
  - Run **`SIMPLEINSTALL`** through ODT or a Telnet session
- Full system upgrades are more complex
  - Theoretically, someone could be "hands and eyes" for you to insert the release media
  - But that's risky – difficult to know what's really going on
  - Better to have the MCP admin on site for this

MCP-4001 17

There are two main types of MCP software upgrade – full upgrades, when you are migrating from, say, MCP 12.0 to MCP 13.0, and Interim Corrections (ICs), which are updates of individual software products.

It turns out that installing ICs remotely is quite easy, although it can be a little tedious if you need to do a lot of them at the same time. We simply establish a Remote Desktop session to the Windows side of the server, access the Unisys support site using Internet Explorer from that session, download the appropriate product updates, decompress them into a temporary folder, and then transfer the result (usually a wrapped container file) to a shared MCP directory using either **`xcopy`** or Windows Explorer click-and-drag. Once on the MCP side, we unwrap the container file and run **`SYSTEM/SIMPLEINSTALL`** to install the updated software, either through **`COMS/ODT/DRIVER`** (i.e., the MARC interface you get when entering **`??MARC`** on the ODT), or through a separate Telnet session to MARC. If necessary, you can halt/load the MCP side through the Remote Desktop session after installing the updates.

Full updates are more problematic, partly because they are riskier, and partly because there are CDs involved. In theory, you could remain remote and have someone in the server room act as your hands and eyes to insert CDs for you and to tell you what is happening if the upgrade has a problem that breaks your connectivity to the system. It's often difficult for someone who is not familiar with the MCP to give you a useful description of what's going on, however, so my preference is to be on site when it's time to do a full upgrade, and to do it myself.



Managing Time – The SNTP Client

- Synchronizes MCP clock with a time source
  - Requires TCP/IP access to an NTP time server
  - Windows AD controllers *can* be time sources
- Cleverly handles clock inaccuracies
  - Slow clocks are simply adjusted forward
  - Fast clocks are gradually "slewed" backward
  - Avoids discontinuities in logs and audit trails
  - Transparent to application programs
- Automatically adjusts for DST
  - Establishes a schedule for time changes
  - Changes MCP time and time zone per that schedule

MCP-4001 18

Historically, one of the annoying things about MCP administration has been the need to monitor the system clock and adjust it periodically. The clock in most MCP systems seems to run a little fast, which is worse than running slow, since it means you need to adjust the time backwards, which can cause grief with, at a minimum, the system logs and DMSII audit trails. Then there is the necessity for most sites to switch between standard and daylight-savings time twice each year.

The good news is that there is much better way to deal with this issue, called the **`TIME`** DSS, which is an implementation of SNTP – the Simplified Network Time Protocol, which in turn is derived from NTP, the full-bore Network Time Protocol. This mechanism has been available on the MCP since the early 2000s.

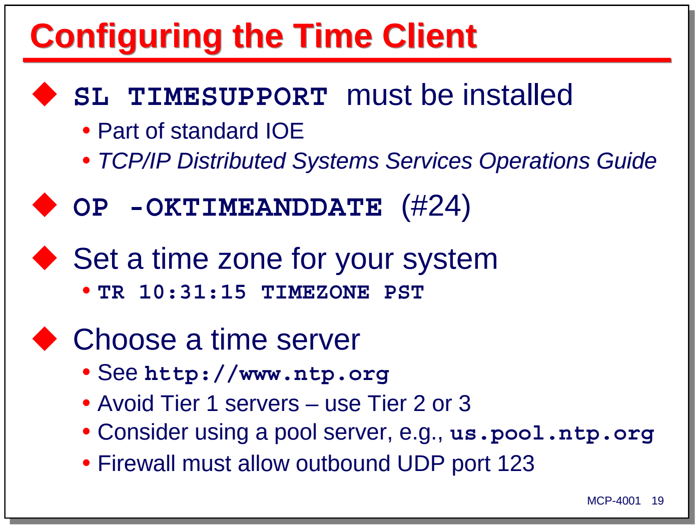
What it does is synchronize the MCP system clock with an external time source. All that it requires is TCP/IP access to an NTP server (the time source), and a little bit of configuration on the MCP side. If you are running a Windows Active Directory domain, you may already have an NTP server – DCs can be configured to act as a time source. If that is not an option for you, access to external time servers is readily available, and free.

The **`TIME`** DSS implements a very clever way of dealing with clock inaccuracies. If the clock runs slow, the DSS periodically adjusts it forward to the correct time. If the DSS runs continuously (which it should), this adjustment is typically on the order of a few milliseconds. If the clock runs fast, the DSS gradually "slews" the time backwards to the correct time. This slewing is done in a way that is transparent to applications, logs, and the audit trails. The **`TIME`** DSS also measures the clock's drift over a period of time, and uses that measured drift to apply a bias to the system clock, so that the clock stays more accurate between synchronizations with the time server.

The clock for the system under discussion here runs fast about seven seconds per day, but the **`TIME`** DSS's drift adjustment is good enough that the clock only needs to be corrected about 10 milliseconds every 11 hours when the DSS synchronizes with the external time server.

But wait! There's more! One of the nicest features of the **`TIME`** DSS is that it can *automatically adjust the clock for daylight-savings time*. You establish a schedule when the time changes are to occur, and specify the time zone change (e.g., EST to EDT). The DSS can change the time automatically (and use the same slewing mechanism to avoid time discontinuities in the logs and audit trails) according to the schedule you have set. You can now sleep in on those two Sundays every year.

Over the next couple of slides I will show how to set up the **`TIME`** DSS and configure it. I love this feature of the system, and highly recommend that you use it.



Configuring the Time Client

- **SL TIMESUPPORT** must be installed
  - Part of standard IOE
  - *TCP/IP Distributed Systems Services Operations Guide*
- **OP -OKTIMEANDDATE** (#24)
- Set a time zone for your system
  - **TR 10:31:15 TIMEZONE PST**
- Choose a time server
  - See **http://www.ntp.org**
  - Avoid Tier 1 servers – use Tier 2 or 3
  - Consider using a pool server, e.g., **us.pool.ntp.org**
  - Firewall must allow outbound UDP port 123

MCP-4001 19

The main prerequisite is that the **SYSTEM/TIME/SUPPORT** library must be installed and **SL**-ed. The appropriate **NA REG** commands must also have been done to establish the library as a DSS. This software is part of the standard IOE, so **SIMPLEINSTALL** should have already done all of this for you. In any case, the installation requirements are documented in the section on Time Synchronization in the TCP/IP DSS guide.

The next thing you probably want to do is reset **OP** option 24, **OKTIMEANDDATE**. This option will suspend system initialization until you confirm the system time and respond to an RSVP. Since we will be running the DSS, we don't really need that (and since we are going for unattended system operation, we *really* don't need that).

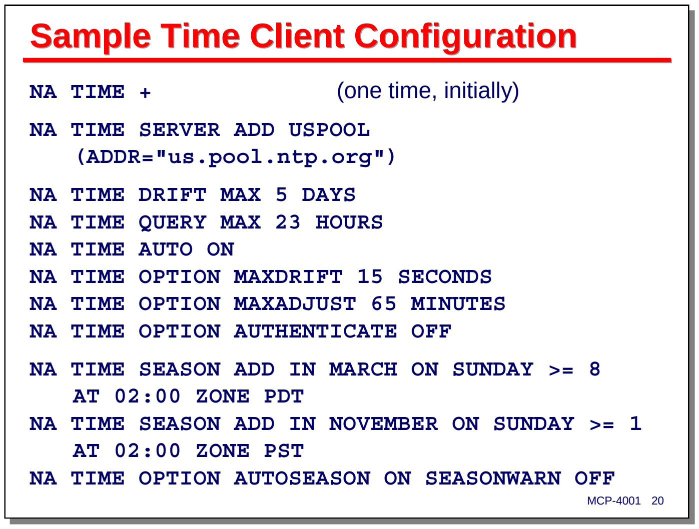
Next, set an appropriate time zone for your system using the **TR** command, as shown on the slide.

Perhaps the most challenging thing to do is select a time server. If your Windows domain controller is configured to supply this service, choosing it is probably the best option, as this will help keep all of the servers in your network closely time-synchronized. If that is not an option, check out the NTP web site at http://www.ntp.org for instructions on how to choose a time server.

Time servers are arranged in layers, termed *strata*. The highest layer is Stratum 1, which communicates directly with high-quality time sources, such as the atomic clocks at the U.S. Naval Observatory. Stratum 2 servers get their time from Stratum 1 servers using NTP. Stratum 3 servers get their time from Stratum 2 servers, and so forth. Time clients (such as the MCP's DSS) can get their time from a server at any stratum, but NTP etiquette suggests that you to avoid connecting to Stratum 1 servers to avoid overloading them. Find a Stratum 2 or 3 server near you – the quality of its time will be plenty good enough.

If you don't have a good idea for a time server to use (or just don't want to bother looking for one), a perfectly adequate choice is to use one of the time server pools. This is just a group of time servers who have agreed to be members of the pool. The pool's DNS address assignment changes randomly to switch among the pooled servers. The intent of this is to automatically spread the load across the pool. See the NTP web site for a full list of pools and their domain names. In the U.S., you can use **us.pool.ntp.org** for a time server address. In Canada, it's **ca.pool.ntp.org**, in Great Britain it's **uk.pool.ntp.org**, and in Australia it's **au.pool.ntp.org**.

One more thing to consider is that NTP and SNTP use UDP on port 123. Your firewall must allow outbound connections from your system to your designated time server over this port.



Sample Time Client Configuration

```
NA TIME +                    (one time, initially)

NA TIME SERVER ADD USPOOL
   (ADDR="us.pool.ntp.org")

NA TIME DRIFT MAX 5 DAYS
NA TIME QUERY MAX 23 HOURS
NA TIME AUTO ON
NA TIME OPTION MAXDRIFT 15 SECONDS
NA TIME OPTION MAXADJUST 65 MINUTES
NA TIME OPTION AUTHENTICATE OFF

NA TIME SEASON ADD IN MARCH ON SUNDAY >= 8
   AT 02:00 ZONE PDT
NA TIME SEASON ADD IN NOVEMBER ON SUNDAY >= 1
   AT 02:00 ZONE PST
NA TIME OPTION AUTOSEASON ON SEASONWARN OFF
```

MCP-4001 20

This slide shows a series of **NA TIME** commands that will enable and configure the DSS to synchronize the clock on an MCP system. The settings shown here are a good default set, but you may wish to change some of them for your particular environment.

NA TIME + initiates the DSS. You should need to do this only once.

NA TIME SERVER defines a time server internally named **USPOOL** and associates it with with a domain name or IP address. Multiple time servers may be defined for redundancy; the DSS will automatically choose one.

NA DRIFT MAX specifies the maximum amount of time between drift recalculations

NA TIME QUERY MAX specifies how much time elapses between synchronizations with the time server. Once or twice a day is usually more than adequate.

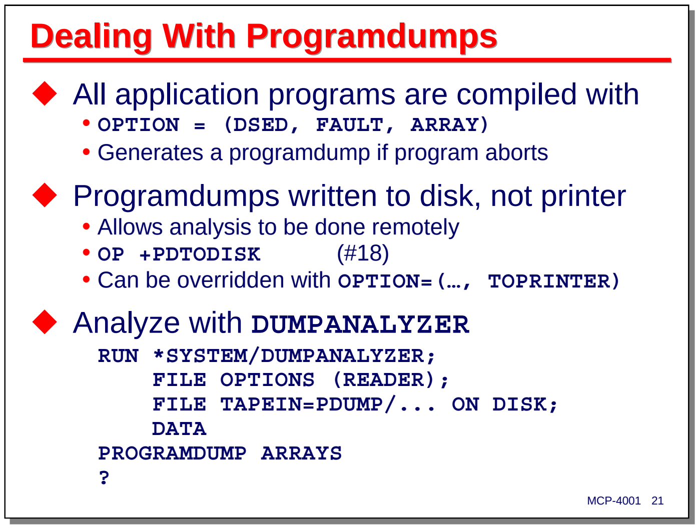
NA TIME AUTO ON allows the DSS to begin using shorter drift and query delays, and automatically increase them over a period of time until their respective maxima are reached. This speeds up the determination of drift adjustment after the system is initialized.

NA TIME OPTION MAXDRIFT specifies the maximum time per day the clock is allowed to drift. **MAXADJUST** specifies the maximum amount of time the system will correct the clock automatically (larger corrections will generate an RSVP). **AUTHENTICATE** indicates whether authentication with the time server(s) is enabled. This is not typically done.

The two **NA TIME SEASON** commands establish the schedule for daylight-savings time changes. The first one specifies that on the first Sunday in March with a date greater than or equal to **8** (i.e., the second Sunday of the month), at 2:00 a.m., the time zone should be changed to Pacific Daylight Time. The second command specifies that on the first Sunday in November with a date greater than or equal to **1** (i.e., the first Sunday of the month), at 2:00 a.m., the time zone should be changed to Pacific Standard Time. Thus the system's time will change automatically between PDT and PST on those dates.

NA TIME AUTOSEASON ON enables the automatic seasonal time change mechanism to operate on the schedule just established. **SEASONWARN OFF** suppresses an RSVP message requiring operator approval of the time change.

That's all there is to it. The rest of the configuration options for the DSS can be left at their defaults. The DSS automatically saves all configuration settings in the file ***TIME/CONFIG**.



Dealing With Programdumps

- All application programs are compiled with
  - `OPTION = (DSED, FAULT, ARRAY)`
  - Generates a programdump if program aborts
- Programdumps written to disk, not printer
  - Allows analysis to be done remotely
  - `OP +PDTODISK` (#18)
  - Can be overridden with `OPTION=(…, TOPRINTER)`
- Analyze with **DUMPANALYZER**

```
RUN *SYSTEM/DUMPANALYZER;
    FILE OPTIONS (READER);
    FILE TAPEIN=PDUMP/... ON DISK;
    DATA
PROGRAMDUMP ARRAYS
?
```

MCP-4001 21

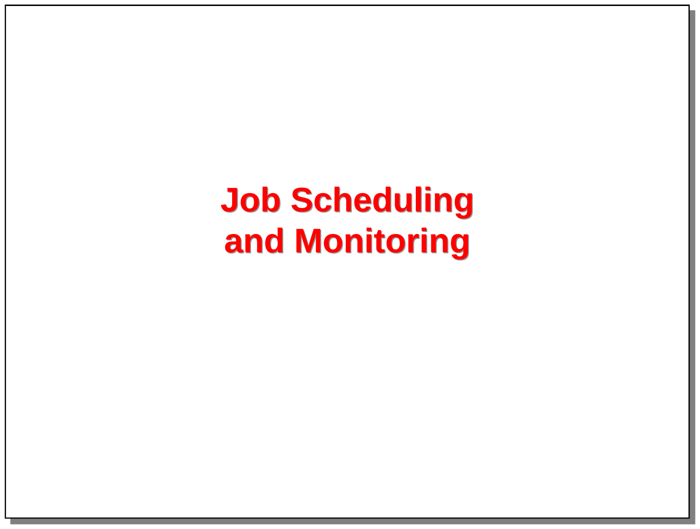
The final topic in this section of the presentation discusses how we deal with programdumps. These are useful for analyzing aborts and other application software problems, but dealing with them remotely requires a couple of special techniques.

First, we have standard WFL jobs for compiling the application programs, and those jobs always compile the programs with **OPTION=(FAULT, DSED, ARRAY)**. This will cause a programdump to be generated if the program aborts or is DS-ed.

If we did nothing else, those dumps would go to whatever default printer applied to the program at the time. That makes it difficult to read the dump remotely, so we need to override the system's default programdump behavior by setting the **PDTODISK** option (**OP+18**). This causes programdumps to be written by default to a disk file with a **PDUMP** prefix. If you actually want a printed dump, you can override this system option on a case-by-base basis by using the **TOPRINTER** item in the **OPTION** task attribute.

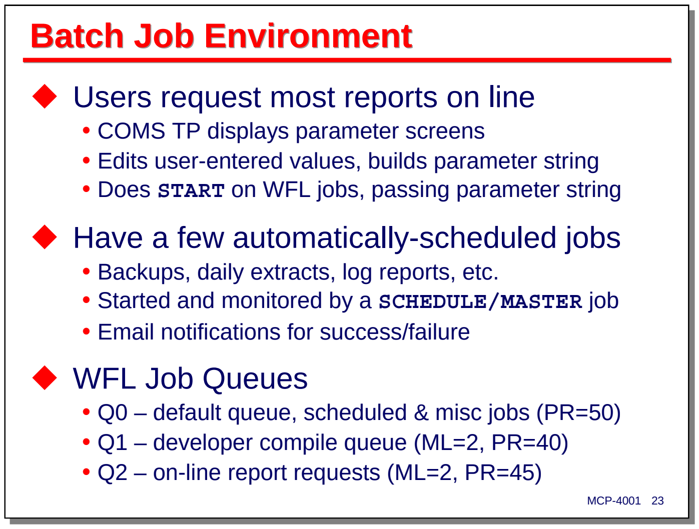
That disk file will have a name of the form **PDUMP/**<codefile name>**/**<date>**/**<time>**/**<mix number>. This file can be analyzed with the **SYSTEM/DUMPANALYZER** utility. **DUMPANALYZER** has an interactive mode, but I've never cared for that. I like the printed programdump format. **DUMPANALYZER** will generate a printer file with almost exactly the same format as a standard programdump using its **PROGRAMDUMP** command.

I have WFL job that is set up to run **DUMPANALYZER** and email the resulting printer file to me. I just fill in the appropriate **PDUMP** file name, run the job, and wait for the email to show up. It usually takes only seconds.



Job Scheduling and Monitoring

The next topic is the relatively simple way we've developed to schedule and remotely monitor our batch jobs.



Batch Job Environment

- Users request most reports on line
  - COMS TP displays parameter screens
  - Edits user-entered values, builds parameter string
  - Does **START** on WFL jobs, passing parameter string
- Have a few automatically-scheduled jobs
  - Backups, daily extracts, log reports, etc.
  - Started and monitored by a **SCHEDULE/MASTER** job
  - Email notifications for success/failure
- WFL Job Queues
  - Q0 – default queue, scheduled & misc jobs (PR=50)
  - Q1 – developer compile queue (ML=2, PR=40)
  - Q2 – on-line report requests (ML=2, PR=45)

MCP-4001 23

The bulk of the batch work on the system consists of reports and posting runs that the users request themselves. The majority of these are simple one-program executions; only a few of the jobs involve multi-step execution.

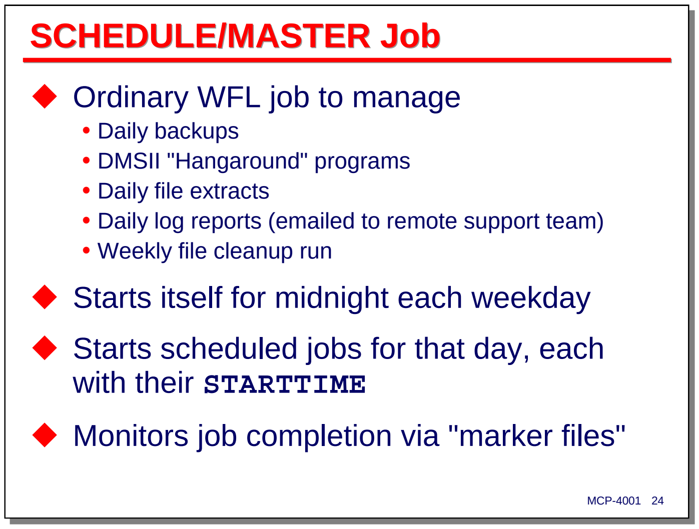
There is a COMS TP that handles requests for these batch jobs. It displays menus of selections, and as necessary, forms for parameter data. It edits the parameters entered by the users and builds a parameter string in the form acceptable to that job. Each batch job has its own WFL, which the COMS TP starts (via **CALL SYSTEM WFL** in COBOL). The WFL accepts the parameter and passes it as necessary to one or more of the programs within the job. We have a WFL **$INCLUDE** file that contains a set of parameter parsing routines the jobs use to extract portions of the parameter string they need – printer destinations and the like.

The batch jobs simply run open loop. The users do not have a good way to track progress of their jobs – they typically suspect there's a problem when they don't get printer output in a reasonable time. This is probably the weakest part of the batch environment, and is a fairly frequent source of calls for support. We (the support team) often do not know there was a problem until the next morning, when we get a series of emailed log reports. While this situation isn't the best, it's adequate for this customer, and we haven't felt driven to try to improve upon it.

In addition to these on-request jobs the users run themselves, we have a few jobs that run automatically on a schedule. These consist primarily of the daily backup, some database extract/file transfer jobs, and the log reports for the support team. To schedule and monitor these automatic jobs, we have built a WFL job we call the **SCHEDULE/MASTER** – sort of a grand name for a fairly simple facility, which I'll discuss on the next slide. This job has been extensively instrumented with email notification to tell us whether the automatically-scheduled jobs succeed or fail, since no one on site is monitoring them.

We use three job queues to manage the batch workflow.

- Q0 is the default queue, and it is where all of the automatically-scheduled jobs go, along with any miscellaneous batch work we do for development or support. It defaults to priority 50 with no mix limit.
- Q1 is a compile queue for the developers. We do all of our application compiles through a couple of standard WFL jobs. CANDE compiles are rare. This queue defaults to priority 40 with a mix limit of 2.
- Q2 is used for on-line requests. All of the jobs initiated by the COMS TP mentioned above go through this queue. It defaults to priority 45 with a mix limit of 2. On rare occasion application programs will loop or go into a waiting state, causing this queue to back up. This often gets reported to us as a printing problem, because users are submitting batch requests but not getting their print output. We've learned to look at the job queues for these "printer problems."



SCHEDULE/MASTER Job

- Ordinary WFL job to manage
  - Daily backups
  - DMSII "Hangaround" programs
  - Daily file extracts
  - Daily log reports (emailed to remote support team)
  - Weekly file cleanup run
- Starts itself for midnight each weekday
- Starts scheduled jobs for that day, each with their **`STARTTIME`**
- Monitors job completion via "marker files"

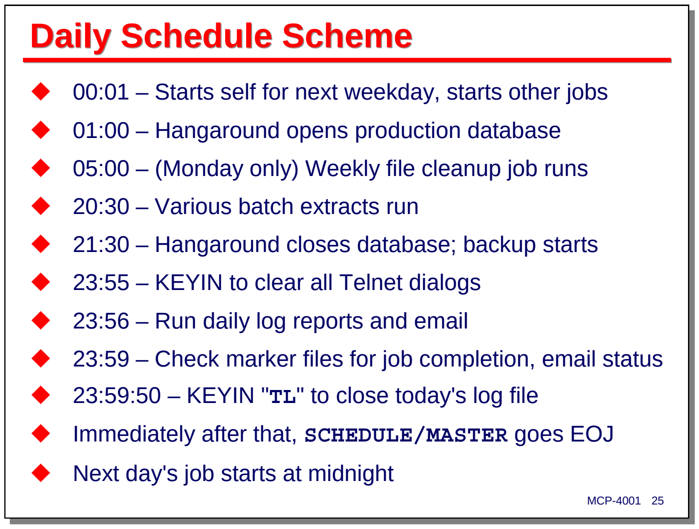
MCP-4001 24

The **`SCHEDULE/MASTER`** job handles all of the automatically-scheduled jobs for the system. These include the daily backup runs, DMSII "hangaround" programs to keep the DBS open, some nightly file extract/transfer jobs, and daily log reports for the support team. Once a week, this job schedules a file cleanup job, which I will describe shortly.

This job runs each weekday. There is normally so little activity on the system over weekends that we do not bother to do backups or other scheduled jobs. One of the first things the job does each day is a **`START`** with **`STARTTIME`** to schedule itself for 00:00 the next weekday. The job is designed to run for just that day and end itself just before midnight.

One function of this job is to schedule each of the automatic jobs that are to run that day. It does this simply using **`START`** with **`STARTTIME`**. The automatic jobs then sit in their job queue until their appointed time to run. There is nothing fancy about this mechanism – the **`START`** statements are coded directly in the WFL source. To change the schedule, we edit the WFL.

Since jobs initiated from other jobs are asynchronous and independent, **`SCHEDULE/MASTER`** does not have any way to directly monitor or control the automatic jobs. The way we have chosen to monitor these jobs is through a technique we call "marker files," which I will also describe shortly.



Daily Schedule Scheme

- 00:01 – Starts self for next weekday, starts other jobs
- 01:00 – Hangaround opens production database
- 05:00 – (Monday only) Weekly file cleanup job runs
- 20:30 – Various batch extracts run
- 21:30 – Hangaround closes database; backup starts
- 23:55 – KEYIN to clear all Telnet dialogs
- 23:56 – Run daily log reports and email
- 23:59 – Check marker files for job completion, email status
- 23:59:50 – KEYIN "**`TL`**" to close today's log file
- Immediately after that, **`SCHEDULE/MASTER`** goes EOJ
- Next day's job starts at midnight

MCP-4001 25

This slide shows the basic daily routine for the **`SCHEDULE/MASTER`** job. Just after midnight it starts itself for the next weekday, and starts the other jobs for that day with their respective **`STARTTIME`**s. At 1:00 the "hangaround" job for the production database runs to hold the DBS open for the remainder of the day. Some SQL Server DTS extracts run about this time, which is why we open the DBS so early.

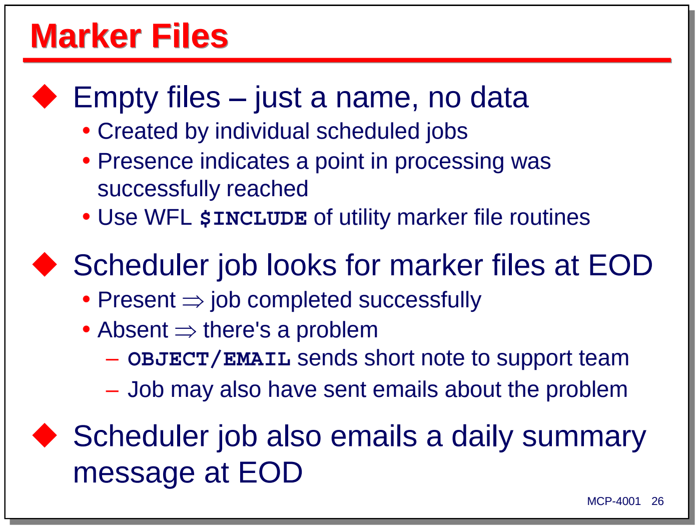
On Mondays at 5:00, the weekly file cleanup job runs. Other than that, the **`SCHEDULE/MASTER`** job does nothing else throughout the day. Most users are usually off the system by 6:00 p.m.

At 8:30 p.m., various file extract/transfer jobs begin to run, and at 9:30 the daily backup begins.

At 11:55 p.m., we run our **`KEYIN`** utility to clear all Telnet dialogs. This will terminate the COMS sessions for anyone who neglected to log off earlier. Shortly after that, the daily log reports for the support team are run and emailed. The "marker files" are also checked, and emails are sent for any jobs that did not indicate they completed successfully. The **`SCHEDULE/MASTER`** job also sends an email if everything did complete successfully, so that we know it's still working properly.

The final thing the job does is **`KEYIN`** "**`TL`**" command to close that day's system log and start a new one for tomorrow. It then goes EOJ.

The next weekday, the whole cycle starts over.



Marker Files

- Empty files – just a name, no data
  - Created by individual scheduled jobs
  - Presence indicates a point in processing was successfully reached
  - Use WFL **`$INCLUDE`** of utility marker file routines
- Scheduler job looks for marker files at EOD
  - Present ⇒ job completed successfully
  - Absent ⇒ there's a problem
    - **`OBJECT/EMAIL`** sends short note to support team
    - Job may also have sent emails about the problem
- Scheduler job also emails a daily summary message at EOD

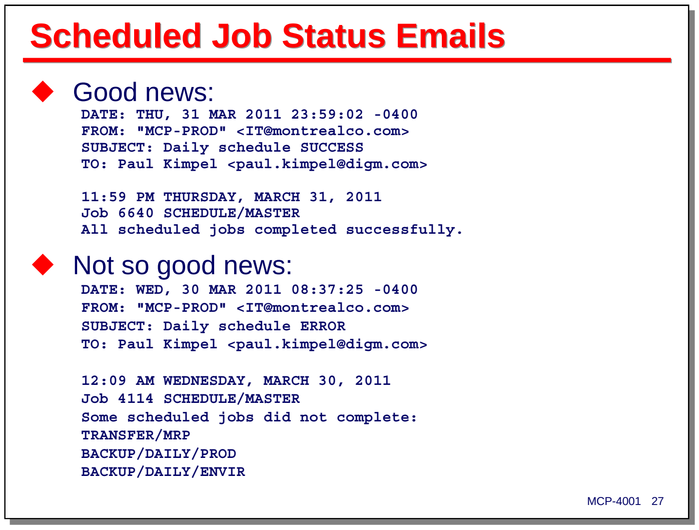
MCP-4001 26

Marker files are a very simple technique to allow independent jobs and tasks to communicate status information with each other. These are empty files – just a name in the disk directory with no data associated with them.

The files are created by individual jobs. It is their presence or absence that is most significant, although they could be used to transfer small amounts of data between jobs using, say, the **`NOTE`** attribute. Generally, it is the presence of a marker file that indicates a job has reached a certain point successfully. If the file is absent, that generally indicates there's a problem. File naming is a convention that must be established among the jobs creating and monitoring the marker files. We have a series of WFL subroutines in an **`$INCLUDE`** file that makes it easy for jobs to create these files and test for their presence.

The primary use of marker files is to allow the automatic jobs to report their completion status back to the **`SCHEDULE/MASTER`** job. It polls for the presence of these files as part of its end-of-day processing. If any of the files are not present, it sends an email to the support team indicating that fact. Most of the automatic jobs also try to send a more detailed error email if they detect a problem.

`SCHEDULE/MASTER` always sends at least one email at the conclusion of its end-of-day processing. If all jobs completed successfully, it says so. If some jobs did not complete successfully, it sends a summary email indicating which ones had problems.



Scheduled Job Status Emails

- Good news:

```
DATE: THU, 31 MAR 2011 23:59:02 -0400
FROM: "MCP-PROD" <IT@montrealco.com>
SUBJECT: Daily schedule SUCCESS
TO: Paul Kimpel <paul.kimpel@digm.com>

11:59 PM THURSDAY, MARCH 31, 2011
Job 6640 SCHEDULE/MASTER
All scheduled jobs completed successfully.
```

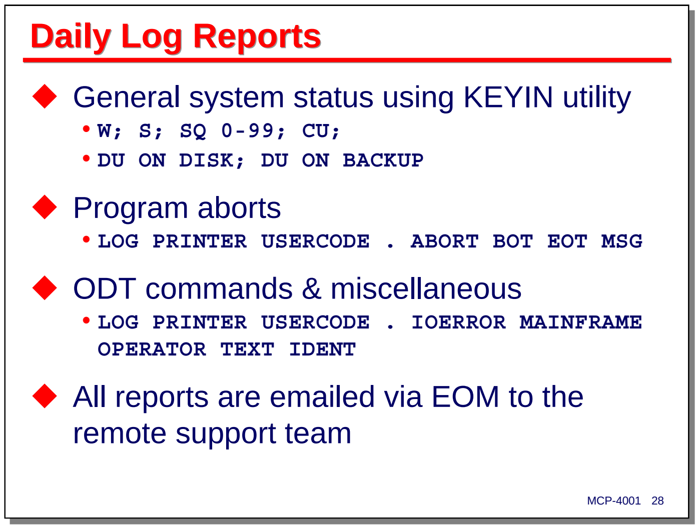
- Not so good news:

```
DATE: WED, 30 MAR 2011 08:37:25 -0400
FROM: "MCP-PROD" <IT@montrealco.com>
SUBJECT: Daily schedule ERROR
TO: Paul Kimpel <paul.kimpel@digm.com>

12:09 AM WEDNESDAY, MARCH 30, 2011
Job 4114 SCHEDULE/MASTER
Some scheduled jobs did not complete:
TRANSFER/MRP
BACKUP/DAILY/PROD
BACKUP/DAILY/ENVIR
```

MCP-4001 27

This slide shows examples of the status emails sent by **`SCHEDULE/MASTER`**. Hopefully, we see the first one each morning. If there were problems, we will get one like the second.



Daily Log Reports

- General system status using KEYIN utility
  - `W; S; SQ 0-99; CU;`
  - `DU ON DISK; DU ON BACKUP`
- Program aborts
  - `LOG PRINTER USERCODE . ABORT BOT EOT MSG`
- ODT commands & miscellaneous
  - `LOG PRINTER USERCODE . IOERROR MAINFRAME OPERATOR TEXT IDENT`
- All reports are emailed via EOM to the remote support team

MCP-4001 28

Another function of the **SCHEDULE/MASTER** job is to email the support team some status reports at end of day. We use the email capability of EOM to send these.

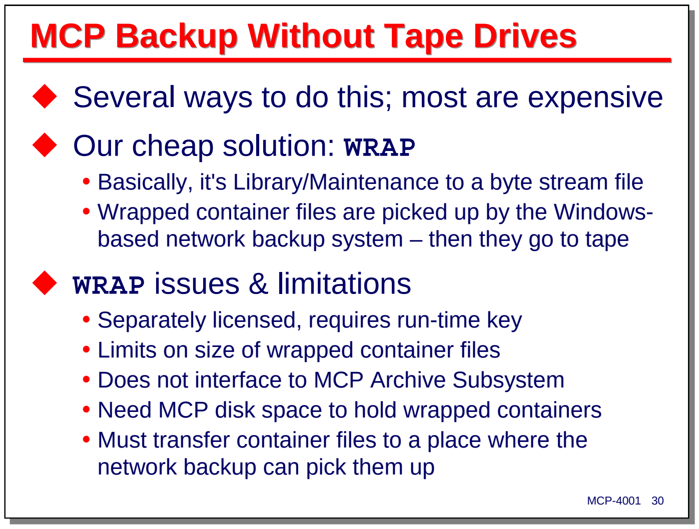
Our **KEYIN** utility outputs to a printer file, so the first of these status reports is a **KEYIN** that shows us the result of a series of ODT commands. These include waiting entries, scheduled entries, what is in the job queues, and current MCP memory utilization. This tells us if jobs have gotten stuck in the mix or the job queues have backed up. It also reports disk utilization on our two active families, so we can monitor whether something is chewing up available space.

Next are two **SYSTEM/LOGANALYZER** runs. The first looks for any tasks that aborted during the previous day. This has turned out to be one of the most effective support tools we have. A few years ago, we found that several programs were crashing on a regular basis, but no one was bothering to tell us. You know how users are – if something doesn't work, they just try it again. This log report has allowed us to find out when programs abort, for whatever reason, and to be proactive in determining what happened. Usually there is a **PDUMP** file, which I am able to analyze remotely. Then I either try to fix the problem myself, or pass it along to my colleague (1500 miles away) who does most of the application program maintenance.

The second log report shows any I/O subsystem or mainframe errors, plus any operator commands that occurred during the day. We have yet to see any hardware errors reported, but I like to look at the commands being done, to make sure someone isn't getting to a MARC screen or the ODT and doing things they shouldn't. What we normally see in this report is a bunch of **LP** and **PR** commands submitted by the Print System, but occasionally I see something that needs attention.

Backup and Disk Space Management

Since the Libra 300 does not have any tape drives, you are probably wondering how we do backup. This next section of the presentation describes our approach.



MCP Backup Without Tape Drives

- Several ways to do this; most are expensive
- Our cheap solution: **`WRAP`**
  - Basically, it's Library/Maintenance to a byte stream file
  - Wrapped container files are picked up by the Windows-based network backup system – then they go to tape
- **`WRAP`** issues & limitations
  - Separately licensed, requires run-time key
  - Limits on size of wrapped container files
  - Does not interface to MCP Archive Subsystem
  - Need MCP disk space to hold wrapped containers
  - Must transfer container files to a place where the network backup can pick them up

MCP-4001 30

There are several ways to back up an MCP system without tape drives, but unless we want to use the CD-R drive, or start plugging in various forms of USB storage (both of which would require physical access to the machine and conflict with our unattended operations goal), they all come down to some sort of network-based backup. There are commercial backup solutions for the MCP (and other environments) that will work over a network, but most are expensive. The customer had already invested in a network-based backup solution for its Windows servers and clients, so we decided to leverage that.

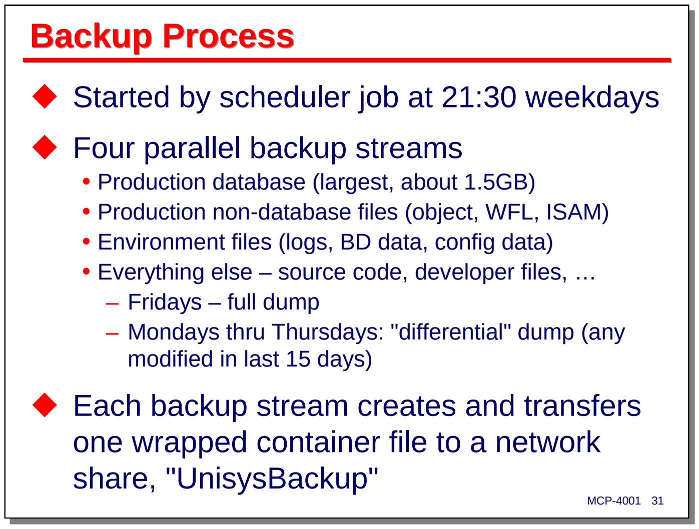
Our cheap-but-workable solution is based on the MCP's **`WRAP`** facility. Basically, this is a form of Library/Maintenance, activated from WFL like **`COPY`**, but it reads and writes backup data to bytestream files instead of tapes or CD-R media. The resulting bytestream file (termed a "wrapped container") is somewhat like a Unix/Linux "tarball." **`WRAP`** encodes all of the file system metadata (record and block sizes, timestamps, etc.) the same way that ordinary Library/Maintenance does, and the bytestream file can be transferred to other systems (and back) using FTP, SMB/CIFS, or any other stream-oriented protocol. The bytestream files an be **`UNWRAP`**-ed to restore individual files within them to the MCP file system.

`UNWRAP` is bundled with the IOE, but **`WRAP`** requires an additional license. It is not very expensive, at least on the low-end ClearPath machines.

The essence of our solution is to **`WRAP`** MCP files for backup, transfer the wrapped container to the Windows file system, and allow the container file to be picked up by the normal Windows-based network backup mechanism. File restoration simply follows the reverse path using **`UNWRAP`**.

`WRAP` is really nice, but it has some issues and limitations:

- **`WRAP`** requires a run-time key, which is what you get with the additional license. **`WRAP`** and **`UNWRAP`** are built into the MCP – there is not a separate piece of software.
- There are some limits on the size of wrapped containers, and there is a 4GB limit on the size of a file that can be transferred using SMB/CIFS.
- One big drawback is that **`WRAP`** does not interface with the MCP Archive system; thus the archive cannot be used to locate and automatically restore files on backup media.
- Since **`WRAP`** creates its wrapped containers in the MCP file system, you need sufficient MCP disk space to hold a copy of the backup before it can be transferred someplace else. This is one of the main purposes for our **`BACKUP`** family.
- Finally, if you are going to use **`WRAP`** for backup, you need to transfer the wrapped container files off the MCP file system to a safe location. FTP and SMB/CIFS are typical ways to do this.



Backup Process

- Started by scheduler job at 21:30 weekdays
- Four parallel backup streams
  - Production database (largest, about 1.5GB)
  - Production non-database files (object, WFL, ISAM)
  - Environment files (logs, BD data, config data)
  - Everything else – source code, developer files, …
    - Fridays – full dump
    - Mondays thru Thursdays: "differential" dump (any modified in last 15 days)
- Each backup stream creates and transfers one wrapped container file to a network share, "UnisysBackup"

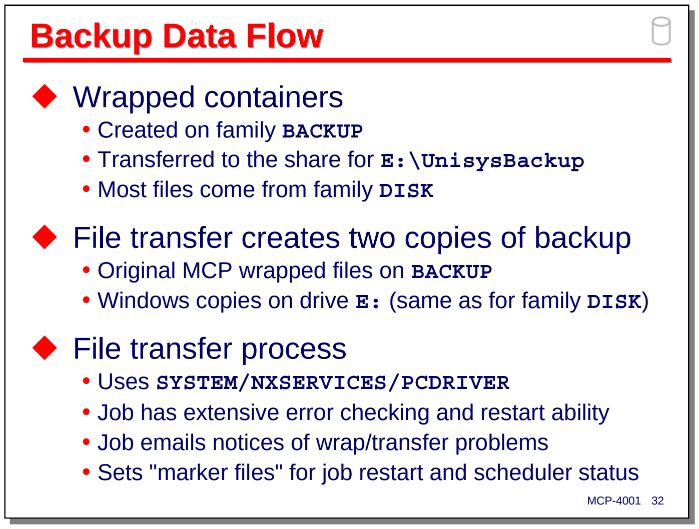
MCP-4001 31

Our daily backup process is started by the **`SCHEDULE/MASTER`** at 9:30 p.m. each weekday evening. It does not run on weekends.

We have divided the files to be backed up into four parallel streams. This is done partly to divide the backup into logical pieces, partly to reduce the overall backup time, and partly to limit the size of the resulting wrapped container files.

- Production database files, including the **`DESCRIPTION`** file, DASDL source, and support WFLs. This is the largest backup, currently about 1.5GB.
- Production non-database files. This is basically everything from the production usercode, and includes the production object files, WFLs, and non-DMSII files. This backup formerly included a large number of KEYEDIOII files, but has shrunk considerably since we have completed the conversion to DMSII.
- "Environment" files. These include the system logs, printer backup files, and system configuration data (e.g., CNS and TCP/IP "init" files). These files are kept in a separate backup because we may not want to restore them, or may want to be very selective in restoring them, in a DR situation where we are restoring to a different system in a different location.
- Everything else. This includes all of the application source code, developer files, and some test files. This in turn is divided into two backup streams:
  - On Fridays, we dump everything in this category.
  - During Monday through Thursday, we do something like a differential dump, backing up everything that is new or changed in the last 15 days. Since **`WRAP`** cannot interface to the Archive subsystem, we cannot do a true differential dump, but this is good enough.

Each backup stream gets created and initiated as a separate WFL job. Each job wraps its appropriate set of files, and then transfers the resulting wrapped container to the "UnisysBackup" share, physically located on Windows drive E: of the system.



Backup Data Flow

- Wrapped containers
  - Created on family **`BACKUP`**
  - Transferred to the share for **`E:\UnisysBackup`**
  - Most files come from family **`DISK`**
- File transfer creates two copies of backup
  - Original MCP wrapped files on **`BACKUP`**
  - Windows copies on drive **`E:`** (same as for family **`DISK`**)
- File transfer process
  - Uses **`SYSTEM/NXSERVICES/PCDRIVER`**
  - Job has extensive error checking and restart ability
  - Job emails notices of wrap/transfer problems
  - Sets "marker files" for job restart and scheduler status

MCP-4001 32

The data flow for the backup was carefully thought out and tailored to the physical disk subsystem we have on the Libra 300.

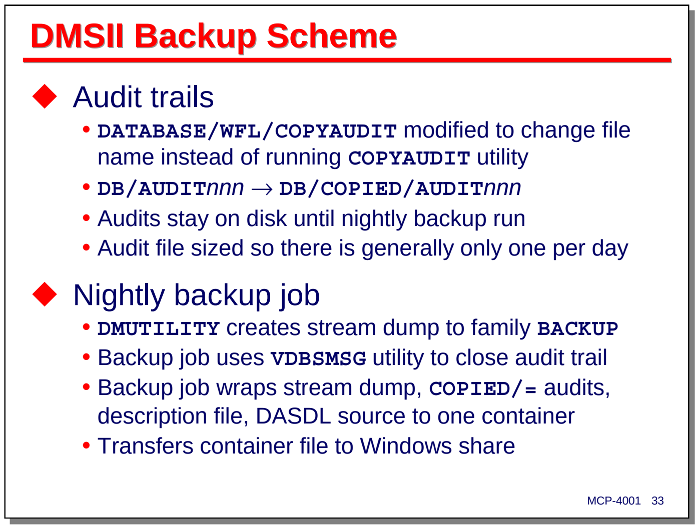
The wrapped container files are all created on family **`BACKUP`** (which is physically resident on Windows drive D:). We have enough space on that family to hold all of the wrapped containers from one night's backups.

Most of the files being backed up come from family **`DISK`** (which is physically resident on Windows drive E:), so this immediately gets the backup to a location separate from most of the original files. The wrapped container files are then transferred using **`SYSTEM/NXSERVICES/PCDRIVER`** to the **`UnisysBackup`** share, which is physically resident on Windows drive E:. From there they are picked up by the customer's Windows-based network file backup mechanism, and ultimately written to tape for off-site storage. The Libra is at Plant B and the network file backup is based at Plant A, so this arrangement gets the backups off the Libra and to a separate location within a few hours.

If you look back to the diagram of the Libra's physical disk configuration [slide 6], you will see that this backup scheme ends up creating two copies of the backup, one in the MCP file system and one in the Windows file system, each on a physically separate disk subsystem. This was done intentionally to maximize the chance we could survive a failure in one of the disk subsystems before the network backup was able to take the wrapped containers off site.

Some time ago we had serious network connectivity and throughput problems between Plants A and B, and that drove us to instrument the backup and file transfer process with lots of error checking, email notification, and restart capability. This worked really well while we were having those problems, and while we seldom have problems with the backups anymore, it is good to know that all of that instrumentation is still in place. **`PCDRIVER`** has proven to be an adequate tool for doing these transfers, especially since the MCP has supported CIFS over port 445 support starting in release 12.0.

The jobs driving the backup and file transfer also use "marker files" to control restart after a halt/load and to signal completion status to the **`SCHEDULE/MASTER`** job.



DMSII Backup Scheme

- Audit trails
  - **DATABASE/WFL/COPYAUDIT** modified to change file name instead of running **COPYAUDIT** utility
  - **DB/AUDIT***nnn* → **DB/COPIED/AUDIT***nnn*
  - Audits stay on disk until nightly backup run
  - Audit file sized so there is generally only one per day
- Nightly backup job
  - **DMUTILITY** creates stream dump to family **BACKUP**
  - Backup job uses **VDBSMSG** utility to close audit trail
  - Backup job wraps stream dump, **COPIED/=** audits, description file, DASDL source to one container
  - Transfers container file to Windows share

MCP-4001 33

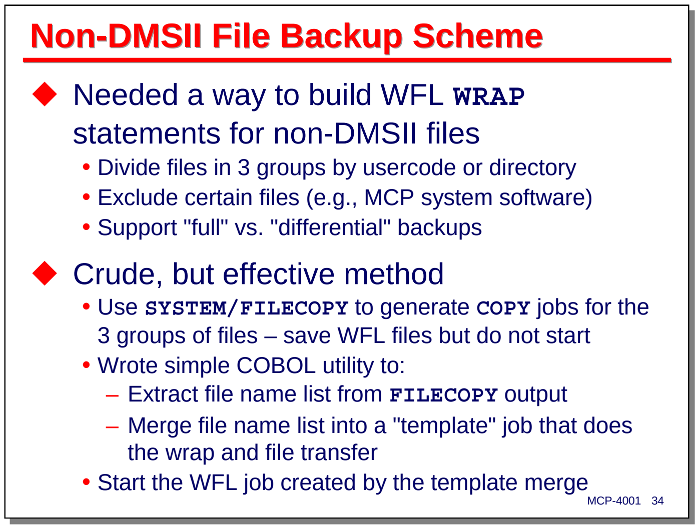
The backup scheme for our production DMSII database has two parts.

First, we do not copy the audit trail files and remove them from the system as they are closed. We use a modified version of **DATABASE/WFL/COPYAUDIT** that simply changes the name of the audit files from **DB/AUDIT***nnn* to **DB/COPIED/AUDIT***nnn*. These closed and renamed audit files simply stay on disk until the nightly run, when they are picked up by the actual database backup.

The audit trail files are sized so that there is typically only one per day. We are currently retaining the audit files on disk for 15 days, which is probably a little excessive – they are eventually removed by the file cleanup process to be described later.

One of the backup streams initiated by the nightly backup job is focused on backing up the production database. It works as follows:

- The job runs **SYSTEM/DMUTILITY** to create a so-called "stream dump" of the database. A stream dump is simply a **DMUTILITY** dump that takes place to a disk file rather than a tape drive. This is a full, on-line dump – the database can still be active and updated while this dump takes place. The stream dump file is stored temporarily on family **BACKUP**.
- The job then runs our **VDBSMSG** utility, which is like a **KEYIN** program for the DMSII Visible DBS interface. It submits an **AUDIT CLOSE FORCE** command to close the current audit trail file (which normally will have all of the database updates since the last backup run). The automatic **COPYAUDIT** mechanism will then cause the audit trail name to be changed to the **COPIED/=** convention.
- The job stages additional files from family **DISK** to **BACKUP**: the **DESCRIPTION** file, DASDL source, utility WFL jobs we use to maintain the database, and the tailored software (**DMSUPPORT**, etc.). When the scheme was originally developed, **WRAP** could only process files from one source family, so this step gathered everything together on family **BACKUP**. Since MCP 12.0, **WRAP** can process multiple source families, so it would now be possible to eliminate this step.
- The job then wraps the stream dump file, all **COPIED/=** audit trail files, **DESCRIPTION** file, DASDL source, tailored code files, and utility WFL files. This wrapped container is also created on family **BACKUP**, so it's a lot of I/O to and from the same disk. After the wrap completes successfully, the stream dump file is removed.
- Finally the job uses **SYSTEM/NXSERVICES/PCDRIVER** to transfer the wrapped container to the **UnisysBackup** share where it will be picked up by the network backup mechanism. The wrapped container is left on family **BACKUP** until the next backup run.



Non-DMSII File Backup Scheme

- Needed a way to build WFL **WRAP** statements for non-DMSII files
  - Divide files in 3 groups by usercode or directory
  - Exclude certain files (e.g., MCP system software)
  - Support "full" vs. "differential" backups
- Crude, but effective method
  - Use **SYSTEM/FILECOPY** to generate **COPY** jobs for the 3 groups of files – save WFL files but do not start
  - Wrote simple COBOL utility to:
    - Extract file name list from **FILECOPY** output
    - Merge file name list into a "template" job that does the wrap and file transfer
  - Start the WFL job created by the template merge

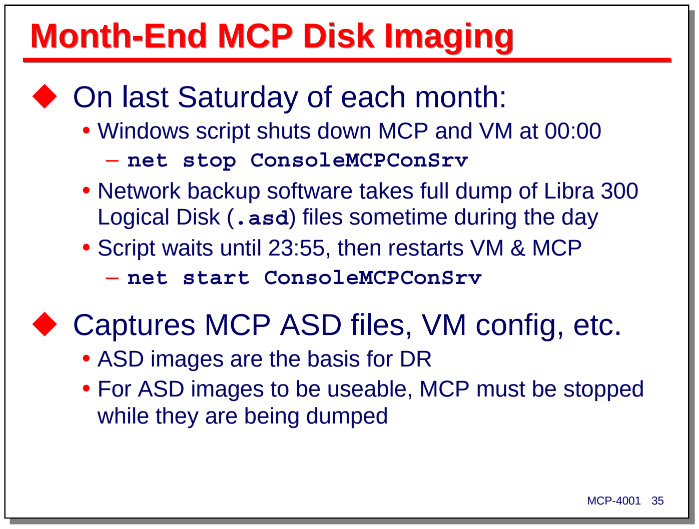
MCP-4001 34

The non-database file backup scheme is a little more complicated. These files are divided into three groups, as previously described, and each group becomes a separate backup stream.

We needed a way to divide the non-database files according to their usercode or directory prefix, to exclude certain files by either directory prefix or filekind, and to support something similar to a full vs. differential file selection process.

The solution we hit upon is simple, a little crude, but very effective.

- **SYSTEM/FILECOPY** has all of the capabilities we need to select or exclude the files to be backed up by a particular stream, so we use it to generate WFL **COPY** jobs, one for each of the three streams. These WFL files are saved but not started.
- I wrote a simple COBOL program, **UTIL/COPYLISTMERGE**, that scans the **FILECOPY** output looking for the "**?COPY**" that signals the start of the list of file names and the "**FROM**" that signals the end of the list. It extracts the WFL records between those two delimiting records and inserts them into a "template" job, creating a new WFL file.
- The main backup job starts the new WFL file created by **COPYLISTMERGE**. This new WFL is temporary and only good for that day's backup.
- The template file is set up to do a **WRAP** instead of a **COPY**, and to then do a file transfer using **SYSTEM/NXSERVICES/PCDRIVER**. Thus it wraps the list of files generated by **FILECOPY** to a container on family **BACKUP**, then transfers that container to the **UnisysBackup** share for later pickup by the network backup mechanism.
- The wrapped containers for each backup stream are kept on family **BACKUP** until they are replaced by the next backup run.
- The template file also supplies logic to restart the wrap and file transfer processes after a halt/load, and to restart the file transfers if they encounter errors for a maximum of three times. It sends email notifications if it encounters any difficulties. It creates marker files to signal completion to the **SCHEDULE/MASTER** job.



Month-End MCP Disk Imaging

- On last Saturday of each month:
  - Windows script shuts down MCP and VM at 00:00
    - `net stop ConsoleMCPConSrv`
  - Network backup software takes full dump of Libra 300 Logical Disk (`.asd`) files sometime during the day
  - Script waits until 23:55, then restarts VM & MCP
    - `net start ConsoleMCPConSrv`
- Captures MCP ASD files, VM config, etc.
  - ASD images are the basis for DR
  - For ASD images to be useable, MCP must be stopped while they are being dumped

MCP-4001 35

The daily backup process works well and does a good job of backing up the MCP file system. What it doesn't do is lend itself to a speedy recovery after a significant disk failure, as we found out the hard way on the predecessor to the Libra 300 when we lost two disks in its RAID-5 array almost simultaneously. Restoring the files that were backed up was not a problem. The problem was essentially having to start from scratch and build a whole new MCP environment – installing the VM software, configuring logical disks, installing system software, and *then* restoring files from the backups. The most difficult part of the restoration was specifying all of the MCP settings, such as **`OP`**, **`SYSOPS`**, etc., that are stored outside the file system – those don't get backed up.

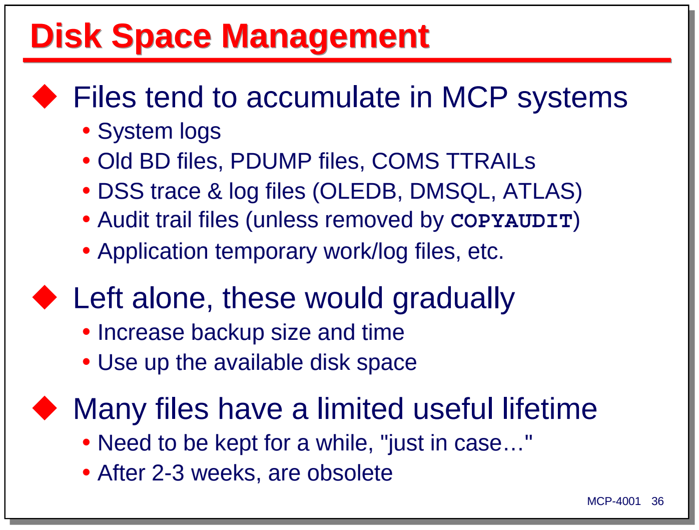
Our solution to this is to periodically back up the vmMCP Logical Disk (**`.asd`**) files themselves. These completely capture all of the hidden MCP settings, plus an image of the entire file system. We can restore the Logical Disk files quickly, then use the daily backups to bring that disk image up to date.

The only caveat in doing this is that the MCP environment must be completely shut down when making a full-image backup of this sort. If the MCP is actively using the Logical Disk files, the resulting backup can be corrupt.

Here's how our disk imaging process works:

- Early on the last Saturday of each month, a scheduled task on the Windows side of the Libra (written in VBScript) shuts down the vmMCP using the **`net stop`** command shown on the slide.
- The script then goes to sleep until 23:55 Saturday night.
- Sometime during the day, the network backup mechanism makes a full backup of all partitions on the Libra 300. This captures not only the Logical Disk files, but the complete Windows and vmMCP installations as well.
- At 23:55, the script awakens and restarts the MCP environment using the **`net start`** command shown. This reinitializes the VM and automatically halt/loads the MCP.

This Logical Disk imaging process, along with the daily backups, form the core of our DR plan.



Disk Space Management

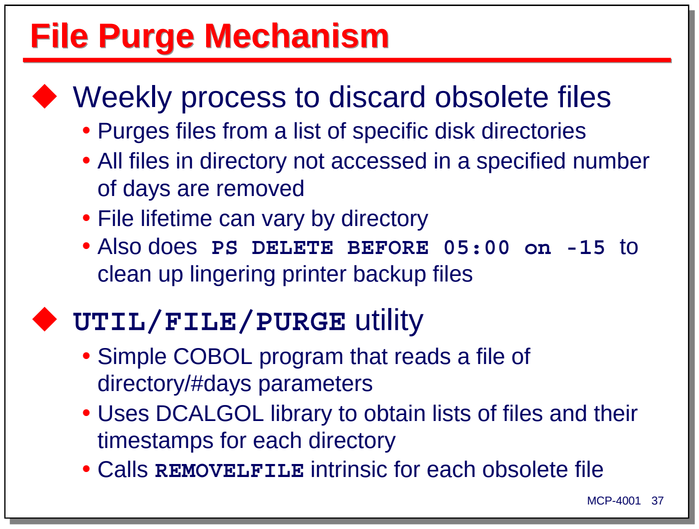
- Files tend to accumulate in MCP systems
  - System logs
  - Old BD files, PDUMP files, COMS TTRAILs
  - DSS trace & log files (OLEDB, DMSQL, ATLAS)
  - Audit trail files (unless removed by **`COPYAUDIT`**)
  - Application temporary work/log files, etc.
- Left alone, these would gradually
  - Increase backup size and time
  - Use up the available disk space
- Many files have a limited useful lifetime
  - Need to be kept for a while, "just in case…"
  - After 2-3 weeks, are obsolete

MCP-4001 36

We back up files to preserve them, but some files are not worth preserving for very long. Certain kinds of files tend to accumulate on MCP systems. These include the system logs, old printer-backup files, **`PDUMP`** files, COMS **`TTRAIL`**s, and the trace and log files from some of the DSSes, such as OLE DB. Given the way we are handling the backup of DMSII audit trails, those files will also tend to accumulate. Applications often generate temporary and/or log files that will need to be cleaned up periodically as well.

Unless we do something about these accumulating disk files, they will gradually increase the size of the backups and the time required to do them, and will eventually take up all of the available disk space.

The thing about many of these accumulating disk files is that they have a limited useful lifetime, or at least a limited useful presence in the file subsystem. They need to be kept for a while, in case they are needed for reference or rerun purposes, but generally after a couple of weeks, we have no further use for them.



File Purge Mechanism

- Weekly process to discard obsolete files
  - Purges files from a list of specific disk directories
  - All files in directory not accessed in a specified number of days are removed
  - File lifetime can vary by directory
  - Also does **`PS DELETE BEFORE 05:00 on -15`** to clean up lingering printer backup files
- **`UTIL/FILE/PURGE`** utility
  - Simple COBOL program that reads a file of directory/#days parameters
  - Uses DCALGOL library to obtain lists of files and their timestamps for each directory
  - Calls **`REMOVELFILE`** intrinsic for each obsolete file

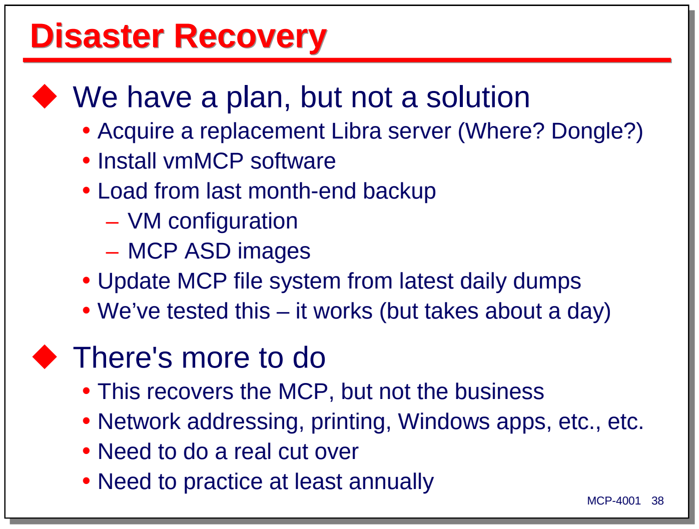
MCP-4001 37

To deal with these accumulating disk files, we have a weekly purge process that looks for them and removes the ones that are obsolete. The scheme is fairly simple – files in certain disk directories are assumed to have a useful life that is measured in the number of days since they were last accessed. If the file has not been touched in at least that many days, it can be removed. The lifetime can vary by directory.

Printer-backup files are among those considered to have a limited useful lifetime. Not only does the weekly process remove lingering **`BD/=`** files, it also submits a **`PS DELETE BEFORE`** command to delete any print requests that are more than 15 days old.

The bulk of the purge process is handled by a COBOL program, **`UTIL/FILE/PURGE`**. This program reads a file of directory names and corresponding lifetimes. For each directory, it calls a DCALGOL library to search that directory and return a list of file names and timestamps. Multiple calls to the library may be necessary for directories with large numbers of files. The program examines the timestamps, and any files that have not been accessed in the number of days indicated by their lifetime are removed. The library has an entry point that wraps the **`REMOVELFILE`** intrinsic, and the program uses that to do the actual file removal.

We need to tweak the list of directories and lifetimes occasionally, but this mechanism does a good job of getting rid of files we no longer need, and helps to maintain an adequate amount of available disk space.



Disaster Recovery

- We have a plan, but not a solution
  - Acquire a replacement Libra server (Where? Dongle?)
  - Install vmMCP software
  - Load from last month-end backup
    - VM configuration
    - MCP ASD images
  - Update MCP file system from latest daily dumps
  - We've tested this – it works (but takes about a day)
- There's more to do
  - This recovers the MCP, but not the business
  - Network addressing, printing, Windows apps, etc., etc.
  - Need to do a real cut over
  - Need to practice at least annually

MCP-4001 38

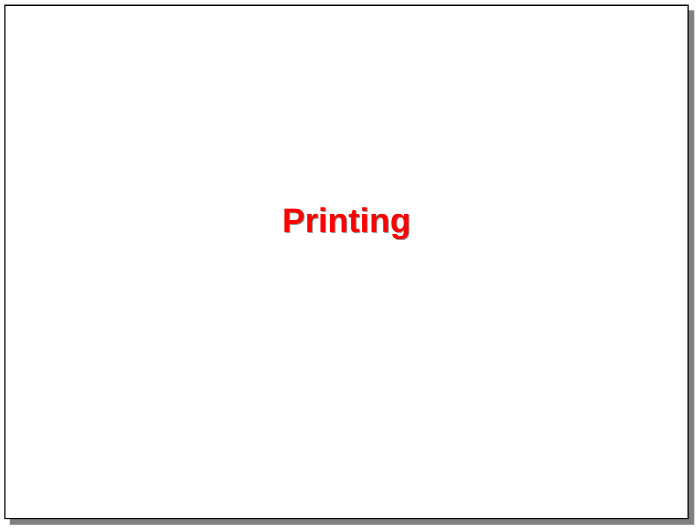
The final topic in this section on backup and disk management concerns disaster recovery. We have been working on this, and have a plan, but like a lot of small shops we do not yet have a full, verified, working solution in place.

In the worse-case scenario of complete IT facility destruction, we would acquire a replacement Libra server. Just where we would get it is as yet an unanswered question. The actual hardware is a standard Dell server, which is readily available on the used market, but if the vmMCP dongle goes, we would need to get another one of those. Somehow.

Then we would need to rebuild the MCP environment, probably by loading the last month-end full image dump, and then loading the latest daily backups on top of that. We've actually tested that mechanism, and it works nicely, but it currently takes us about a day to get everything put back together. This is within management's threshold for downtime during a disaster.

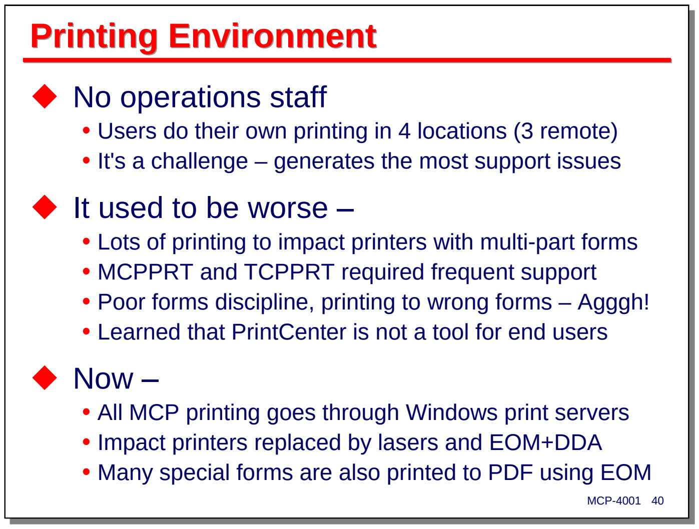
There's obviously a lot more to do. The scheme above recovers the MCP environment, but it does not recover the business. There are a large number of issues that need to be addressed in the areas of network addressing, printing, replacing client systems and applications – the list goes on.

Then we need to actually implement the solution and try it out. My experience with DR plans is that the first time you try to do it, it's a train wreck. You learn a lot from that first attempt, though, and by plowing that experience back into the plan, the subsequent attempts do much better. My other experience is that once you get the DR plan working well enough, you need to repeat it at least once a year so people stay refreshed on their duties, and so you find things that inevitably change in your environment that affect DR but don't get addressed when they initially should.



Printing

The last major section of the presentation concerns how we handle printing for this site.



Printing Environment

- No operations staff
  - Users do their own printing in 4 locations (3 remote)
  - It's a challenge – generates the most support issues
- It used to be worse –
  - Lots of printing to impact printers with multi-part forms
  - MCPPRT and TCPPRT required frequent support
  - Poor forms discipline, printing to wrong forms – Agggh!
  - Learned that PrintCenter is not a tool for end users
- Now –
  - All MCP printing goes through Windows print servers
  - Impact printers replaced by lasers and EOM+DDA
  - Many special forms are also printed to PDF using EOM

MCP-4001 40

Printing has always been a problem at this site. With no operations staff, users do their own printing. This takes place at four locations, three of them remote from the MCP system. One of those remote locations is the company headquarters, where the vast majority of printing takes place. It has proven to be a challenge to keep printing working smoothly. This area of the operation generates more support issues than everything else combined.

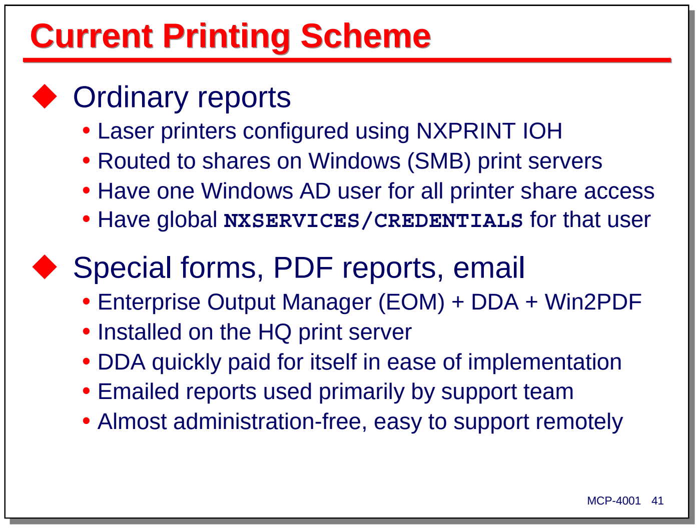
As much of a challenge as printing remains today, it used to be much, much worse. The customer was doing lots of printing to impact printers using multi-part, preprinted forms. We were using mostly MCPPRT and TCPPRT as transport mechanisms. Those tended to require a lot of support. Both are good products, but we had major problems with host name resolution on the network for MCPPRT, and TCPPRT (which was used to print to the larger, shared printers) sometimes had difficulty recovering from timeouts caused by multiple servers trying to print to the same printer at the same time.

The use of preprinted forms meant that forms continually needed to be changed and aligned on the impact printers. The office staff had depressingly poor forms discipline, so we had many cases of output being printed on the wrong forms, printed on misaligned forms, and just about every other screw-up you can imagine. It was a nightmare. We tried using PrintCenter in this environment and learned the hard way that, while it's a great administrator's tool, it's not one for end users.

It took us a while to figure out how to deal with these problems, but it came down to two main changes we made:

- All MCP printing now goes through the customer's Windows print servers. This had two advantages. First, it eliminated the problems caused by multiple hosts contending for shared printers (primarily large laser copier/printers with huge buffers). We have tried to make the Windows print servers the only host sending output to their respective printers. The second advantage is that once the print stream gets off the MCP, it's the network administrator's problem, not mine.
- We ditched the impact printers and preprinted forms, and converted everything to electronic forms overlays using EOM. Initially we were doing just simple overlay printing with EOM, but later acquired DDA. That has allowed us to do some things with forms generation and distribution that we simply could not do before.

We have also acquired the Dane Prairie Win2PDF driver and installed it on the print server which hosts EOM. This has allowed us to generate archival copies of most of the special forms as they are printed, and in may cases we are printing fewer copies of paper forms than before. The office staff loves this capability.



Current Printing Scheme

- Ordinary reports
  - Laser printers configured using NXPRINT IOH
  - Routed to shares on Windows (SMB) print servers
  - Have one Windows AD user for all printer share access
  - Have global **`NXSERVICES/CREDENTIALS`** for that user
- Special forms, PDF reports, email
  - Enterprise Output Manager (EOM) + DDA + Win2PDF
  - Installed on the HQ print server
  - DDA quickly paid for itself in ease of implementation
  - Emailed reports used primarily by support team
  - Almost administration-free, easy to support remotely

MCP-4001 41

The current printing scheme divides printing into two main categories – ordinary reports that go only to paper, and everything else.

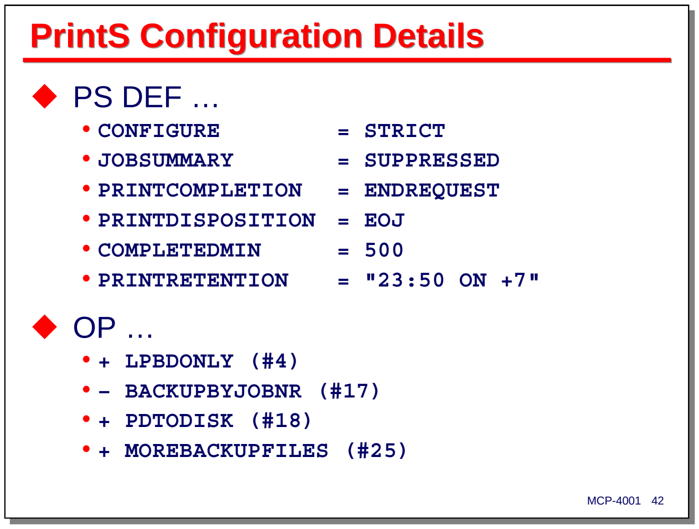
For the ordinary reports, we have a number of printers configured within the MCP using the **`NXPRINT`** I/O Handler (IOH). This is a component of the Redirector (**`SL REDIRSUPPORT`**) that implements SMB/CIFS connections to network printer shares. We have a single Windows Active Directory user defined for the MCP to use in establishing those connections. On the MCP side, the credentials for that network user are stored in a single, global credentials file, **`*NXSERVICES/CREDENTIALS/ MCPPRINTSERVICE`**. This has been defined so that it can be used with any server.

Print requests for ordinary paper reports generally pass right through the MCP Print System. They get sent almost immediately to the appropriate print server, where they are spooled and printed according to the availability of the physical printer.

Everything else goes through Enterprise Output Manager, which runs on the headquarters print server. Win2PDF is also installed on that server. This has allowed us to completely eliminate impact printers and preprinted forms (except for one small Okidata printer used for checks). It took us a while to get up to speed on DDA, but that package quickly paid for itself in the ease with which we were able to convert forms and in the flexibility of formatting it provides.

Emailing reports through EOM has proved to be very useful, although at present only the support team uses that capability.

Our current printing environment is almost administration-free. We still have issues with missing output or disabled printers that need to be researched and dealt with, but most of these turn out to be caused by transient network issues. As discussed on the next slide, we keep both the MCP and EOM print files for a week, and most problems with missing output can be solved simply by reprinting the job from the completed list.



PrintS Configuration Details

- PS DEF …
  - `CONFIGURE          = STRICT`
  - `JOBSUMMARY         = SUPPRESSED`
  - `PRINTCOMPLETION    = ENDREQUEST`
  - `PRINTDISPOSITION   = EOJ`
  - `COMPLETEDMIN       = 500`
  - `PRINTRETENTION     = "23:50 ON +7"`
- OP …
  - `+ LPBDONLY (#4)`
  - `- BACKUPBYJOBNR (#17)`
  - `+ PDTODISK (#18)`
  - `+ MOREBACKUPFILES (#25)`

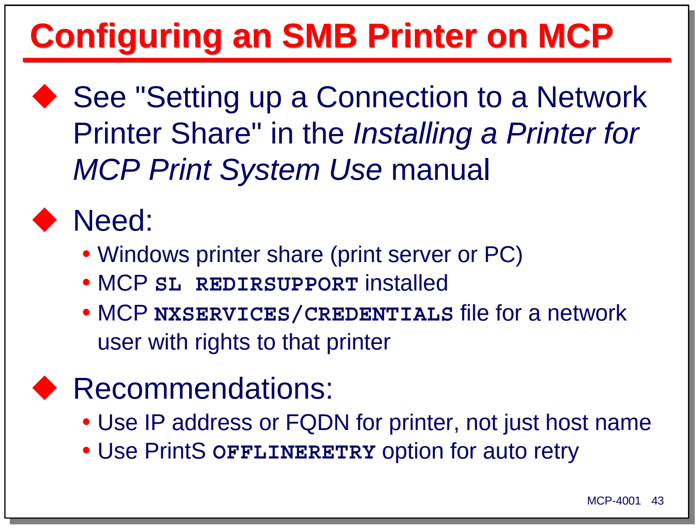
MCP-4001 42

This slide shows the significant Print System configuration settings that we use. Most of these should be obvious, but I want to point out the one that for us has had the most impact.

```
    PS DEF PRINTRETENTION = "23:50 on +7"
```

This tells the Print System to hang onto print requests and their associated **`BD`** files until shortly before midnight, seven days after the printing for them was last completed. In other words, we are keeping all completed print output for a week.

Completed print requests can be reprinted simply by doing a **`PS MOD`** or **`PS FORCE`** on the request number. The request number can be determined from a **`PS SHOW COMPLETED`** display. We almost never need to rerun a job anymore because someone lost the output.



Configuring an SMB Printer on MCP

- See "Setting up a Connection to a Network Printer Share" in the *Installing a Printer for MCP Print System Use* manual
- Need:
  - Windows printer share (print server or PC)
  - MCP **`SL REDIRSUPPORT`** installed
  - MCP **`NXSERVICES/CREDENTIALS`** file for a network user with rights to that printer
- Recommendations:
  - Use IP address or FQDN for printer, not just host name
  - Use PrintS **`OFFLINERETRY`** option for auto retry

MCP-4001 43

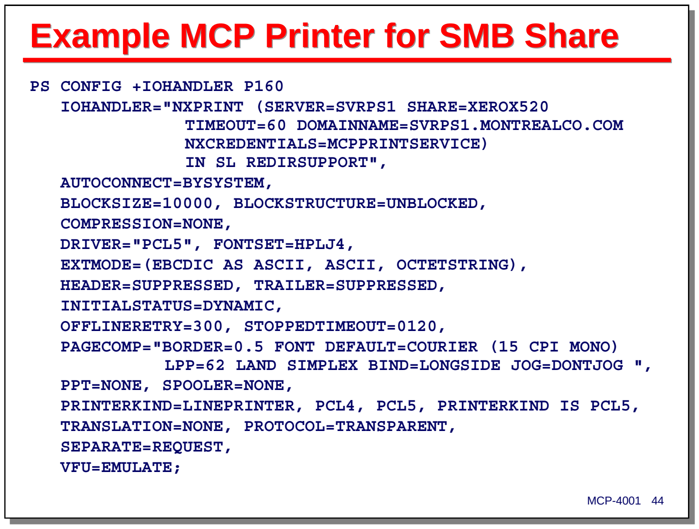
We have found that using SMB printer shares works the best of all the network printing choices in our environment. It is quite easy to set up.

You will need a few things to proceed with the set-up:

- You need to have the shared printer defined on a print server or a PC. We have only tried this with Windows systems, but in theory a Unix or Linux system running SAMBA should be able to export a compatible printer share as well.
- The Redirector must be installed on your MCP system. It is part of the IOE, so it should already be there.
- Unless the network printer share is running completely unsecured, you will need credentials for a network user. The best place to store those in the MCP is a **`NXSERVICES/CREDENTIALS`** file. See the section in the *I/O Subsystem Programming Guide* on **`REDIRSUPPORT`** for information on credentials files.

SMB connections are based on a NetBIOS host name and a share name. We have found that connections are made much more reliably if you use a fully-qualified domain name (FQDN) in addition to a simple host name when configuring the MCP printer.

Also transient network problems can interfere with the transfer of the print data from the MCP to the print server. Using the **`OFFLINERETRY`** option in the MCP printer configuration will enable automatic retry after a delay that you can specify.

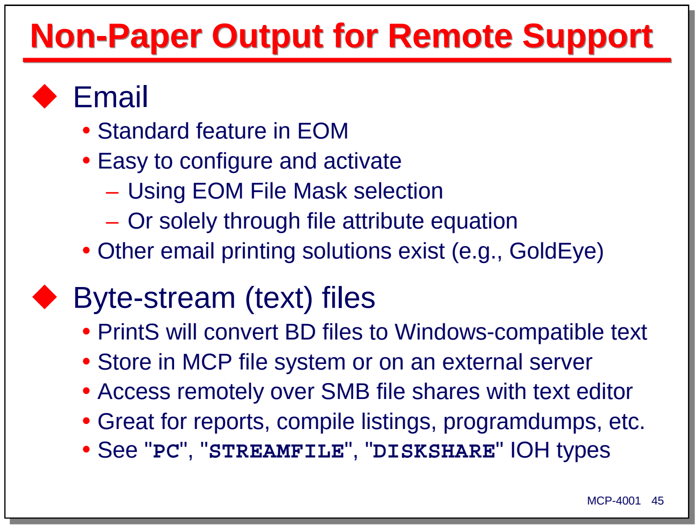


Example MCP Printer for SMB Share

```
PS CONFIG +IOHANDLER P160
   IOHANDLER="NXPRINT (SERVER=SVRPS1 SHARE=XEROX520
               TIMEOUT=60 DOMAINNAME=SVRPS1.MONTREALCO.COM
               NXCREDENTIALS=MCPPRINTSERVICE)
               IN SL REDIRSUPPORT",
   AUTOCONNECT=BYSYSTEM,
   BLOCKSIZE=10000, BLOCKSTRUCTURE=UNBLOCKED,
   COMPRESSION=NONE,
   DRIVER="PCL5", FONTSET=HPLJ4,
   EXTMODE=(EBCDIC AS ASCII, ASCII, OCTETSTRING),
   HEADER=SUPPRESSED, TRAILER=SUPPRESSED,
   INITIALSTATUS=DYNAMIC,
   OFFLINERETRY=300, STOPPEDTIMEOUT=0120,
   PAGECOMP="BORDER=0.5 FONT DEFAULT=COURIER (15 CPI MONO)
             LPP=62 LAND SIMPLEX BIND=LONGSIDE JOG=DONTJOG ",
   PPT=NONE, SPOOLER=NONE,
   PRINTERKIND=LINEPRINTER, PCL4, PCL5, PRINTERKIND IS PCL5,
   TRANSLATION=NONE, PROTOCOL=TRANSPARENT,
   SEPARATE=REQUEST,
   VFU=EMULATE;
```

MCP-4001 44

This slide shows a sample **`PS CONFIG`** command that establishes and configures an MCP printer for an SMB printer share. The most significant part of this command is the **`IOHANDLER`** clause. Note that the **`SERVER`** parameter specifies the NetBIOS host name, but the **`DOMMAINNAME`** parameter specifies the FQDN.



Non-Paper Output for Remote Support

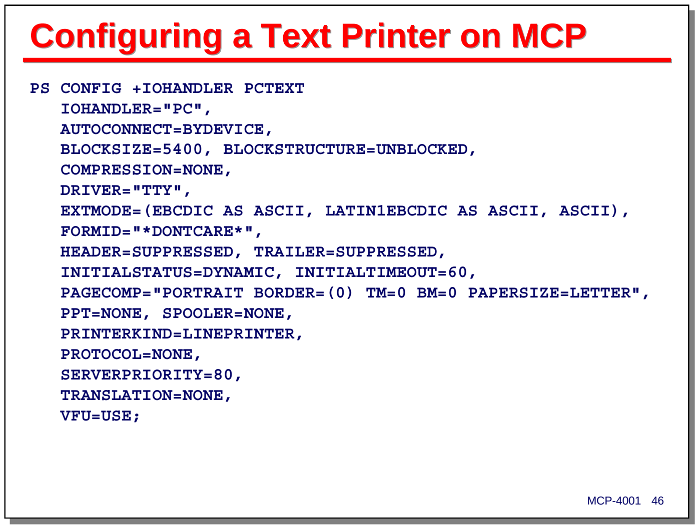
- Email
  - Standard feature in EOM
  - Easy to configure and activate
    - Using EOM File Mask selection
    - Or solely through file attribute equation
  - Other email printing solutions exist (e.g., GoldEye)
- Byte-stream (text) files
  - PrintS will convert BD files to Windows-compatible text
  - Store in MCP file system or on an external server
  - Access remotely over SMB file shares with text editor
  - Great for reports, compile listings, programdumps, etc.
  - See "**`PC`**", "**`STREAMFILE`**", "**`DISKSHARE`**" IOH types

MCP-4001 45

For our remote support duties, we rely on two forms of non-paper printing. The first is email. The ability to email printer files is a standard feature in EOM, and is not difficult to set up. You can configure file masks to select incoming files for output through email, or you can simply define an email "printer" in EOM and use the **`NOTE`** file attribute in the MCP environment to specify addresses, subject lines, and other email-specific parameters.

Another option we use frequently is byte-stream or text file printer output. You can configure a printer in the MCP Print System to convert **`BD`** files to line-oriented text files that are compatible with Windows. By placing these text files in a directory that is shared by the MCP on the network (such as your usercode's "**`_HOME_`**" share), you can then access these files from your personal system using the text editor or other Windows application of your choice. Access from Unix and Linux systems is also possible using this technique. I find this really useful for looking at compilation listings, programdumps, and other large reports that I don't necessarily want to print, or don't want to print in their entirety.

For more information on configuring a text printer, see the **`PC`**, **`STREAMFILE`**, and **`DISKSHARE`** I/O Handler types in the *Installing a Printer for MCP Print System Use* manual.

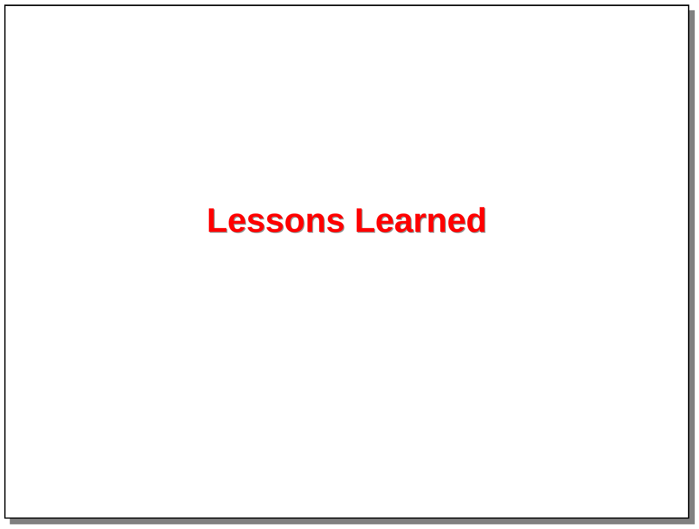


Configuring a Text Printer on MCP

```
PS CONFIG +IOHANDLER PCTEXT
   IOHANDLER="PC",
   AUTOCONNECT=BYDEVICE,
   BLOCKSIZE=5400, BLOCKSTRUCTURE=UNBLOCKED,
   COMPRESSION=NONE,
   DRIVER="TTY",
   EXTMODE=(EBCDIC AS ASCII, LATIN1EBCDIC AS ASCII, ASCII),
   FORMID="*DONTCARE*",
   HEADER=SUPPRESSED, TRAILER=SUPPRESSED,
   INITIALSTATUS=DYNAMIC, INITIALTIMEOUT=60,
   PAGECOMP="PORTRAIT BORDER=(0) TM=0 BM=0 PAPERSIZE=LETTER",
   PPT=NONE, SPOOLER=NONE,
   PRINTERKIND=LINEPRINTER,
   PROTOCOL=NONE,
   SERVERPRIORITY=80,
   TRANSLATION=NONE,
   VFU=USE;
```

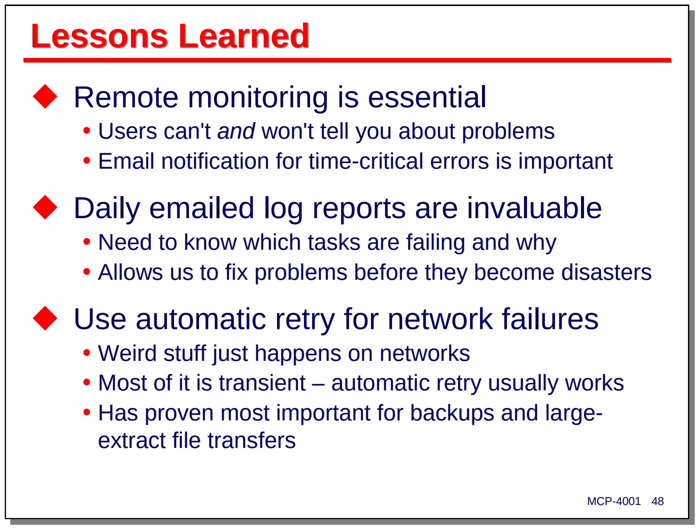
MCP-4001 46

This slide shows a sample Print System configuration for a printer using the **`PC`** IOH. The **`STREAMFILE`** IOH is a more modern implementation – it has more configuration options and more parameters that can be specified in the **`DESTINATION`** file attribute.



Lessons Learned

In the final section of this presentation, I'll discuss some lessons we have learned over the years from our experience with unattended operation and remote support.



Lessons Learned

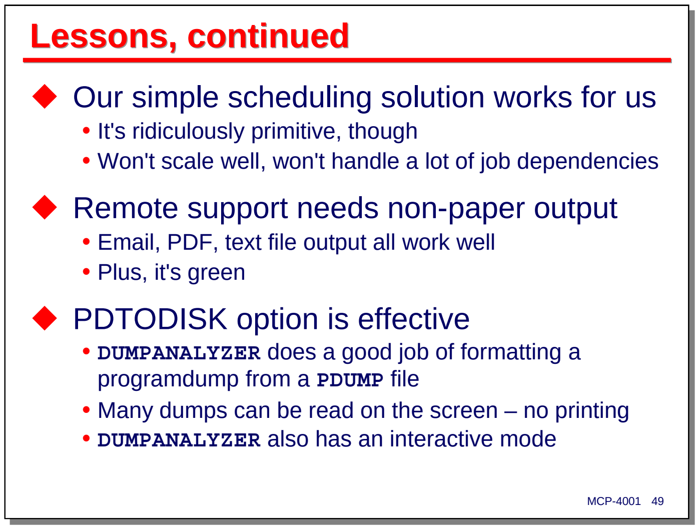
- Remote monitoring is essential
  - Users can't *and* won't tell you about problems
  - Email notification for time-critical errors is important
- Daily emailed log reports are invaluable
  - Need to know which tasks are failing and why
  - Allows us to fix problems before they become disasters
- Use automatic retry for network failures
  - Weird stuff just happens on networks
  - Most of it is transient – automatic retry usually works
  - Has proven most important for backups and large-extract file transfers

MCP-4001 48

First, we have found that remote monitoring of the system is essential. Continuous monitoring is not that important, but you need to have a good view at least once a day of what is happening on the system and whether there are any outstanding issues. Once a day is adequate for this customer's environment; other shops may have tighter service level requirements. For critical tasks, such as the nightly backups, we decided that immediate notification of problems was important and have implemented that. For this environment, email is to notification tool of choice. It is relatively easy to instrument WFL jobs and on-line programs with email notification using the standard MCP **`OBJECT/EMAIL`** utility.

Second, the daily log reports emailed to the support team have proven invaluable in identifying programs that have problems. The "**`LOG ABORT`**" report has been particularly useful. This allows us to identify program crashes whether the users tell us about them or not, and generally to fix the problems before they become serious. Also, we used to get somewhat fuzzy reports from users about things not working properly, or slow response, and we've found that a lot of the time these are actually program crashes the users cannot see. The log reports give us the information to put what the users are telling us in context and understand it better.

When doing file transfers on an automatic basis – especially large file transfers such as the wrapped containers from our nightly backups, we have found it necessary to code in automatic retry when there are errors. Weird stuff just happens on networks, and often the types of problems that cause file transfers to fail are transient. Automatically retrying the transfer some reasonable number of times usually resolves the problem without the need for (and delay involved in) manual intervention.



Lessons, continued

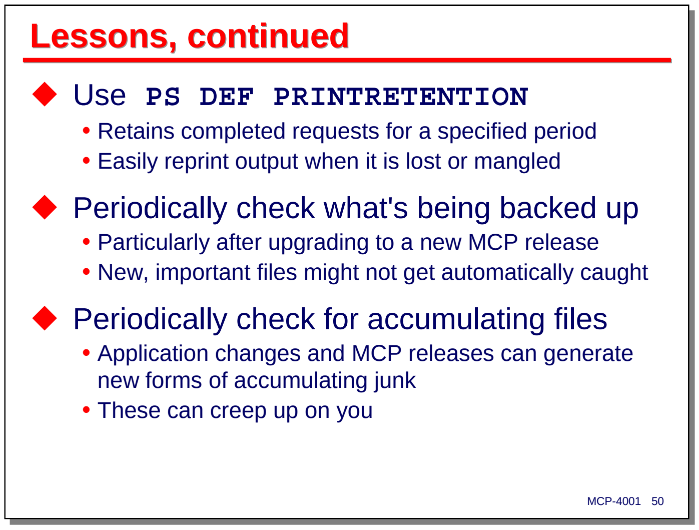
- Our simple scheduling solution works for us
  - It's ridiculously primitive, though
  - Won't scale well, won't handle a lot of job dependencies
- Remote support needs non-paper output
  - Email, PDF, text file output all work well
  - Plus, it's green
- PDTODISK option is effective
  - **`DUMPANALYZER`** does a good job of formatting a programdump from a **`PDUMP`** file
  - Many dumps can be read on the screen – no printing
  - **`DUMPANALYZER`** also has an interactive mode

MCP-4001 49

In terms of automated job scheduling, our simple, home-grown scheduler job works well for us. It certainly doesn't have B&L looking over their shoulder, though. It won't scale well to handle more than a few dozen jobs, and will not handle a lot of inter-job dependencies. For this environment, though, it is adequate and was cheap to build.

If you are going to do remote support, you need to be able to look at printer output, and since you are remote, paper is not much of an option. Email, PDF, and text file printer output all work well over the wire. Even if you are not remotely located, they are good ways to look at output that does not really need to be printed, and they are a step in the direction of having a greener operation.

The **`PDTODISK`** option that directs programdumps to **`PDUMP`** files instead of printer output has been highly effective. We get to choose which dumps to print (useful when one problem generates many dumps that are nearly identical), and can use the non-paper output methods mentioned above to view the dump. I'm finding that many dumps can be analyzed directly on the screen, especially if I have a monitor big enough to view all 132 columns at a time. It is also possible to use **`SYSTEM/DUMPANALYZER`** interactively to read a dump, but I've never mastered that, and still prefer the format of the printed dump, whether I actually print it anymore or not.



Lessons, continued

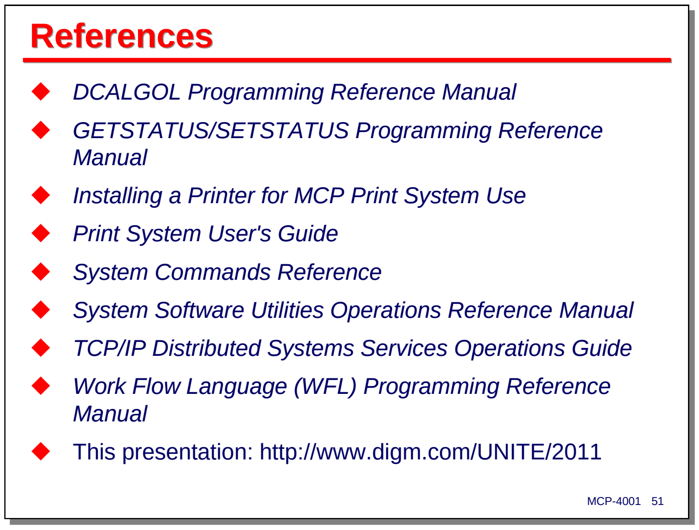
- Use `PS DEF PRINTRETENTION`
  - Retains completed requests for a specified period
  - Easily reprint output when it is lost or mangled
- Periodically check what's being backed up
  - Particularly after upgrading to a new MCP release
  - New, important files might not get automatically caught
- Periodically check for accumulating files
  - Application changes and MCP releases can generate new forms of accumulating junk
  - These can creep up on you

MCP-4001 50

The **`PRINTRETENTION`** option in the MCP Print System has proven to be extremely valuable. Setting this allows you to retain completed print requests for a specified period of time, and then to reprint output easily when it is lost or has gotten damaged during printing. All you need is the print request number. Also, retaining print requests means that the **`BD`** files are retained for several days, which allows them to be backed up, which in turn can be useful for archiving and DR purposes.

Speaking of backup, we've discovered the hard way that you need to check periodically that everything that needs to get backed up actually is being backed up. It is particularly important to verify this after upgrading to new MCP releases, as new configuration data and other files tend to creep in during that process. Even with the automatic file selection mechanism we've built on top of **`SYSTEM/FILECOPY`**, over time the backup list seems to get out of sync with what's needed.

Similarly, you need to check for accumulating files periodically. About once a year, I try to take a good look at all of the files on the system to see what is out that that we do not need. I am especially interested in files with automatically-generated names that are not being removed. These can creep up on you and waste a lot of disk space.



References

- *DCALGOL Programming Reference Manual*
- *GETSTATUS/SETSTATUS Programming Reference Manual*
- *Installing a Printer for MCP Print System Use*
- *Print System User's Guide*
- *System Commands Reference*
- *System Software Utilities Operations Reference Manual*
- *TCP/IP Distributed Systems Services Operations Guide*
- *Work Flow Language (WFL) Programming Reference Manual*
- This presentation: http://www.digm.com/UNITE/2011

MCP-4001 51

This slide lists many of the Unisys documents that are relevant to the subjects I have discussed in this presentation.

A PDF copy of this presentation, including these notes, is available from our web site at the URL shown in the last bullet. Source code for all of the custom WFLs, utilities, and other tools discussed here is also available in a ZIP archive that can be downloaded from that web page.

End

Lights-Out Automation for a Small MCP Shop

2011 UNITE Conference

Session MCP-4001