

An ASAP White Paper

From Mainstar Software Corporation

Another Blackout?

You Can't Afford Not to Have the Right Data Backed Up!

By Kelly Smith

Industry studies have shown that 50 to 80 percent of OS/390 data is not required for disaster recovery. Even so, companies continue to back up unnecessary data sets daily, because identifying which data is critical is a daunting, nearly impossible task. Using traditional methods, there is simply not enough downtime to back up all data every day. Yet you must be prepared to recover from a disaster quickly – which requires that you identify and back up only the critical data.

Everyone thinks, "It won't happen to us"! Did you know?

"The hot site industry (operationally-ready) has successfully recovered 582 companies since 1982. That is an average of 40 per year."

"Nearly 44 percent of these recoveries have resulted from regional events where multiple subscribers have been affected simultaneously."

"The event most likely to send a company to a hot site is loss of power, followed by hardware errors and fires."¹

"Disaster recovery has often been thought of in the same way as funeral planning. It isn't pleasant to think about, and everyone thinks that either they don't need it right now or that they have a lot of time to get it done."

Source: *Network World Network Systems Management Newsletter*, 09/26/01 (<http://www.nwfusion.com>)

This white paper provides some facts to support why you must back up your critical data, covers the different methods of backup, the methods for identifying the data you need to back up, and tells you how to reduce CPU, hardware, tape storage, and transportation costs.

The Problem with Power

We are in the 'Information Age'. Data is the livelihood of many businesses; the Internet has added pressure to provide more, and more current, data; and the visibility of not providing the data is high. If a blackout occurs, companies lose money, credibility, and possibly customers.

"Power outages interrupt operations at 72 percent of U.S. businesses."²

"31% of computer outages are the result of power failures."³

"According to the U.S. Department of Energy, a past power disruption cost a major online bookseller \$1 million a minute in downtime. A large retailer lost \$6 million in sales after a major financial institution's credit card authorization system failed due to power loss."⁴

"Due to the vulnerability of central power plants and transmission lines, power interruptions cost the United States as much as \$80 billion annually (2000)."⁵

The Cost of Downtime

A 2001 Contingency Planning Research study asked companies to identify how much each hour of downtime would cost.

"46% said: up to \$50K"

"28% said: between \$51K and \$250K"

"18% said: between \$251K and \$1M"

"8% said: more than \$1M per hour"³

The same study also revealed the costs of lost business due to server downtime:

"\$108,000 a minute in lost brokerage operations."

"\$43,000 a minute in lost credit card operations."

"\$1,500 a minute in lost airline reservation operations."

"\$1,200 a minute in lost telephone ticket sales operations." ³

A study of downtime costs by the Yankee Group showed that:

"50% of U.S. corporations rate their Internet downtime costs at more than \$1,000 per hour."

"9% of U.S. corporations rate Internet downtime costs at over \$50,000 per hour." ⁶

Given these statistics, how can you not afford to recover all your data in a timely manner?

Traditional Methods of Creating Backups

Here are the traditional methods used to create backups and their inherent shortcomings:

Full Volume Dump – Long Copy

- Outage to complete is generally too long.
- Multi-volume data sets are a problem.
- Must be synchronized and backed up at the same time.

Full Volume Dump – Fast Data Replication

- Copy time is fast but all data has to be synchronized.
- Requires duplicate DASD.
- Requires writing all that data to tape.

Mirror Offsite (less than 10% of U.S. sites mirror)

- Need to make sure you also mirror tape and ML2 (migrated tape).
- Need to be able to restart applications.

Logical Dumps by Application

- Generally masks are used, which include the new data every day, but unfortunately also include the old data every day.
- When you restore the application, you restore all data backed up, which may cause you to miss your recovery time objective.

Essentially, there are three major issues surrounding backups created with traditional methods: 1) the time it takes to back up the data initially, 2) the resources required to store and transport the backed up data, and 3) recovering the data in a timely manner. Specifically, the problems are:

You don't have enough downtime to back up all data every day.

It takes too long to restore the data and get up and running.

Most likely, you will miss your RTO – the time it takes to recover.

You waste time restoring data you may never need.

You waste resources: CPU, tape storage, and transportation costs on backed-up data you don't need.

Traditional methods of creating backups are not efficient.

To solve these problems, you need to reduce the amount of data you back up. So how much data are you currently backing up? How can you back up less data, and how will you identify the data you really need? Most importantly, is there a simple solution?

How Much Data Is Backed Up?

All data cannot possibly be backed up every day or recovered in the event of a disaster. Here's approximately how much data is backed up:

- Full Volume Dump – Long Copy: 100%
- Full Volume Dump – Fast Replication: 100%
- Remote Mirroring: up to 100%
- Logical Dump (e.g., 'pay.**'): 70 to 80%
- Forward Recovery: 35 to 45%
- Rerun Recovery: 30%

There is simply not enough downtime to back up data using a mask on a daily basis. Considerable cost savings can be achieved by not backing up data sets that aren't critical. If you want to back up less data, you need to do it – by APPLICATION – when it is used!

Application backup avoids volume-centric problems, multi-volume data set restoration issues, improves recoverability, provides a unit of backup and recovery, provides synchronization, and allows for prioritization.

Which Data Sets Comprise an Application?

The largest obstacles to overcome when determining critical data are identifying which data sets comprise an application and maintaining the correct list of data sets as applications change.

Problems occur because most application programmers have not worked with the application since its inception. Problems can also arise because programmers often work on a subset of an application, so they don't know every single data set.

Issues with identifying and maintaining the list:

- Applications can be large, making it easy to miss data sets.
- Even if data is identified, do you build one list and back it up every day?
- Who maintains changes over time?
- What if an important data set is missed?

How to Identify Critical Data

If you want to back up less data, and want assurance that all required and critical data is identified on an ongoing basis as applications change, then you need a sophisticated software tool that monitors applications as they execute.

Mainstar: ASAP is that tool!

As an application executes, **ASAP** constructs a 'picture' to identify which data needs to be backed up as it is used. This picture allows the user to determine which data sets are used by an application and which of those data sets are critical. **ASAP** automates the creation of the critical list of data set names to drive a variety of backup processes such as ABARS, FDR, DFSMSdss, or CA-Disk.

Once **ASAP** is configured for an application's recovery objectives, the application can then be tracked and backed up in an automated fashion. **ASAP** will routinely analyze the data sets in use and create selection data sets just before the backup process is performed. This routine re-analysis of the application will identify and track a variety of time-related issues, such as application maintenance, varying application cycle behavior, and unusual data set naming techniques.

ASAP identifies critical data by using information from an array of OS/390 sources,

such as real-time SMF records, JCL, and job schedulers or job masks.

ASAP supports both Rerun and Forward Recovery methodologies and includes in-depth GDG support for each. It automatically tracks changes to these applications to ensure a complete and automated backup and recovery process.

Which Disaster Recovery Method to Use

There are two disaster recovery methods to consider: Forward Recovery and Rerun Recovery. Often, organizations do not realize there is a choice to be made. Yet this choice affects how the data will be identified, what needs to be backed up, and how it needs to be recovered. Failure to choose *explicitly* creates diverse internal expectations and is often a recipe for failure. The choice between the two methods should reflect your organization's expectations regarding:

- How much time you have to get up and running.
- The point you wish to recover to.
- Technical details of the applications.
- Daily, weekly, monthly cycles, etc., and the data required for them.
- Ease and repeatability for testing and for actual disaster recovery.
- Organizational readiness for disaster recovery.

There are substantial differences between the two methods to understand.

Forward Recovery

Forward Recovery is the simplest method of implementation. Forward Recovery assumes the backup is taken at the end of an application cycle or at various points within an application. At the disaster site, the application is restored and continues forward with the next processing cycle. This methodology requires that any input data sets or output data sets needed as input for the next cycle are included in the application backup.

Rerun Recovery

Rerun Recovery is generally more difficult to implement but is less expensive to create because less data is backed up. Rerun Recovery assumes the input data sets are backed up prior to the beginning of application cycle processing. At the recovery site, the application is restored

and the last cycle is rerun. Output data sets are recreated during the rerun of the application. This methodology requires that only critical data sets (inputs to the application) are included in the application backup. Non-critical data sets (outputs) are not included, because they consume backup time and resources.

Know the Facts – Don't Be Blindsided

Implementing a strategy to back up less data is not difficult; however there are several important factors to consider. *Don't be blindsided, read the facts!* Choosing the Forward Recovery or Rerun Recovery method is an important part of the disaster recovery strategy. Failure to choose, or insistence on a single global solution for both system and application data, creates additional work and more problems. Recognize that regardless of which disaster recovery strategy you choose, the following issues will apply.

Forward Recovery Considerations

Forward Recovery is the simplest, most thorough form of identifying and backing up critical data. While it will identify and back up more critical data than Rerun Recovery, Forward Recovery does not compromise your disaster recovery plan.

It includes the most current input and output data sets. To continue forward, output data sets required for the next application cycle need to be included and backed up.

Generation data sets need to be selected as critical based on the recovery method selected. If you select Forward Recovery, you want to continue forward with the most current generation of a generation data group. If you select Rerun Recovery, you want the '-1' generation of a generation data group as input to rerun the application cycle. The following example illustrates the cycle of a generation.

APPLICATION EXECUTES:	END OF CYCLE:	REQUIRED:
0 generation is read	-1	Required for Rerun Recovery
+1 generation is created	0	Required for Forward Recovery

Also consider 'missing generation data sets'. Often, applications read in older generations of a data set to compare numbers or totals, but won't read all the generations in between.

If all generations in between are not included in the backup, there will be problems. When the next application cycle runs at the recovery site, where an older version is referenced (perhaps four generations removed from the current generation), the wrong generation will be referenced if generations in between weren't included in the backup. These generations must be adjusted to the correct generations required by the 'Forward Recovery' setting in the data identification tool.

The following example illustrates missing generations.

APPLICATION EXECUTES:	END OF CYCLE:
-3 generation is read	-4
-2 never referenced	-3
-1 never referenced	-2
0 never referenced	-1
+1 generation is created	0

To Forward Recover this application, the 'End of Cycle' '0' through '-3' generation needs to be included for back up. It is the '-3' generation because a new generation was created.

Rerun Recovery Considerations

Rerun Recovery, considered the cheapest method of backup, is not without problems. The theory behind Rerun Recovery is that input data sets are backed up at the beginning of the application before changes are made to the input data sets and thus recovered at the disaster recovery site and rerun to recreate the output data sets.

No tool, however, can predict what is required in advance of the application executing. It must track the application as it executes. Therefore, the information collected becomes 'after the fact'. So what do you do with this 'after the fact' data? You need to make sure that it is included in the next backup. This requires monitoring the application changes to identify newly modified data sets and adding them to the backup list. **ASAP** provides 'Updated, Renamed and Deleted' (URD) information to identify these data sets. Now let's say you've done all of that, there are still more factors to consider.

Why don't you need to backup output data sets? The fact is that you might need them. Consider this very common scenario: you are at the recovery site and rerun last night's data. Most

likely, that is just one day's worth of data. Now, you start batch, but batch ABENDS because a file isn't there. You happen to be running a cycle that reads in output data sets from Monday to Thursday, but you only restored last night's data.

Rerun Recovery does let you match up numbers from last night's execution, but in the case of a real disaster, once that is done, you will need to continue forward after the rerun has been done.

Generation data sets need to be selected as critical based on the recovery method selected. If you select Rerun Recovery, you want the input to the application. That is the '-1' generation of a generation data group as input to rerun the application cycle. The following example illustrates the cycle of a generation.

APPLICATION EXECUTES:	END OF CYCLE:	REQUIRED:
0 generation is read	-1	Required for Rerun Recovery
+1 generation is created	0	Required for Forward Recovery

Missing generation data sets are a concern here as well; however, they need to be adjusted correctly for Rerun Recovery so the correct generations are backed up. The following example illustrates missing generations.

APPLICATION EXECUTES:	END OF CYCLE:
-3 generation is read	-4
-2 never referenced	-3
-1 never referenced	-2
0 never referenced	-1
+1 generation is created	0

To rerun this application, the '-1' through '-4' generation need to be backed up. The '0' generation at the end of the cycle will be recreated during Rerun Recovery.

Aggressive data truncation to save backup bandwidth may create new problems with recovery. Ask yourself, "If this disaster were real, would we be able to eventually run forward?"

How ASAP Works

First, you need to define an application to **ASAP**. This can be any name you wish. During this process, forward recovery or rerun recovery

options will be selected. Once defined, **ASAP** needs to know which jobs to track. This can be accomplished in a number of ways. The first is to let **ASAP** collect the job information for your scheduler based on trigger job, anchor job, table, etc. This will depend on the scheduler you use. Another popular way is to identify the jobs by jobname mask. If your company has good naming standards, this may be the way to go. The final way is through a user-defined list.

ASAP has a started task, Real-time Selection Process (RSP) that is up and running continuously to gather your important data.

As applications execute, **ASAP** identifies both JCL and SMF information in real time.

The JCL collection identifies all data sets, even:

- Data sets in steps not executed
- Data sets not opened
- Concatenated libraries not opened
- Explodes the PROCLIBS

The SMF collection identifies all data sets, including those:

- Opened and closed
- Deleted or renamed
- Allocated
- Dynamically allocated

At the end of application processing, and 'end of application' job, APPLEND should be executed. This program or job is normally triggered by the last job in the application, signaling the end of the application cycle. The APPLEND program marries the JCL and SMF records together to form a complete picture of the application, analyzes and processes all filters and/or overrides, adjusts generation data sets for either forward or rerun recovery, and builds the list of data sets to back up.

Conclusion

You no longer need to implement disaster recovery in a 'black box' if you understand all the issues. Recognize the distinction and issues between Forward Recovery and Rerun Recovery and create a successful disaster recovery plan.

Backing up less data is desirable in any data processing center, but even more important is the ability to recover that data quickly and reliably in the event of a disaster.

It's now possible to back up less data reliably. Substantial savings *can* be achieved by backing up only critical data, thereby reducing CPU, hardware, tape storage, and transportation costs.

If you are considering a tool other than ASAP, it is imperative that the tool provides both Forward Recovery and Rerun Recovery options. If Rerun Recovery is selected, the tool needs to provide 'Updated, Renamed, and Delete' (URD) information. In addition, the tool needs to be able to select the correct generations based on the method selected and provide the gap fill for 'missing GDGs' when older generation data sets are referenced.

Kelly Smith is a Product Manager for several Mainstar products. She is recognized for her expertise in disaster recovery strategies, storage management, data access, and mirroring techniques. Kelly Smith offers a depth of knowledge honed by 17 years of experience in the mainframe industry. Her deft understanding of diverse installations is the result of helping a wide variety of companies implement SMS, HSM, ABARS, disaster recovery, and data access solutions. Kelly regularly writes and speaks on mainframe issues.

Mainstar is a registered trademark and ASAP is a trademark of Mainstar Software Corporation.

IBM is a registered trademark of International Business Machines Corporation and the following terms are trademarks of International Business Machines Corporation in the United States, and/or other countries: DFSMSdss, OS/390.

1. Eagle Rock Alliance, LTD. All Rights Reserved. (<http://www.eaglerockalliance.com>)
2. Ernst & Young (<http://www.ey.com>)
3. 2001 Contingency Planning Research
© Contingency Planning Research (<http://www.contingencyplanningresearch.com>), a Division of Eagle Rock Alliance, LTD. All Rights Reserved (<http://www.eaglerockalliance.com>)
4. NEMW (<http://www.nemw.org/energy>)
5. Worldwatch Institute (<http://www.worldwatch.org>)
6. Yankee Group (<http://www.yankeegroup.com>)