# 6

# Backup and Recovery

As with all computer systems, your work on the AS/400 should be backed up periodically so you can recover lost data. Losing data is disastrous if you don't have a backup of everything you could possibly lose. It is better to be safe than sorry. Data can be lost for any of the following reasons:

- Someone accidentally deletes an item such as a file member, a program, an entire file, or a whole library.
- Someone intentionally and maliciously deletes an item. Pressing charges for sabotage and suing the person will not bring back the missing item.
- The mass update program you created did not work as expected, so your database is mangled.
- The new Accounting clerk selected the wrong menu option, ignored all the warning panels with bells and whistles you designed to make the clerk

stop, and closed the fiscal year instead of the current month. The clerk will be fired for gross negligence, but you still have to restore the general ledger.

- One of your AS/400 disk drives went down, and you will have to replace it. The manufacturer's warranty will get you a free replacement, but everything the old drive contained is irretrievable.

## INITIALIZING

You need to prepare the tape or diskette before you back up anything to tape or diskette. This process is called initialization.

### *INITIALIZING TAPES*

To initialize a tape, you need to run the Initialize Tape (INZTAP) command. Here is an example:

```
INZTAP DEV(TAP01) NEWVOL(BACKUP) NEWOWNID(ABC_CORP) +
VOL(*MOUNTED) CHECK(*NO) DENSITY(*DEVTYPE) +
CODE(*EBCDIC) ENDOPT(*REWIND) CLEAR(*NO)
```

- DEV(TAP01) indicates that the tape to be initialized has been mounted on tape device TAP01. You must make sure that the tape drive is online and that the tape is not write-protected.

- NEWVOL(BACKUP) gives the tape a new volume ID of BACKUP. Volume IDs can have a maximum of six characters.

- NEWOWNID(ABC_CORP) provides an owner name for this tape (such as your company name).

- VOL(*MOUNTED) means that the INZTAP will initialize any tape that happens to be mounted on the drive, regardless of its current volume ID. For example, if you specify VOL(XYZ), the tape would be initialized only if it already had a volume ID of XYZ.

- CHECK(*NO) indicates that the system is not to check for active files on the tape. If the tape contains data, the data is ignored and the initialization continues. You can leave the default value of *YES if you would rather be warned that the tape contains data.

- DENSITY(*DEVTYPE) indicates that the tape should be initialized to whatever density is supported by the tape drive. Sometimes you can force other values, indicated in bits per inch (bpi), such as 1,600, 3,200, or as high as 43,200.

- CODE(*EBCDIC) means that the tape is initialized for EBCDIC backups. You can use the value *ASCII if you will be using that tape to restore on another system that supports ASCII.

- ENDOPT(*REWIND) indicates that the tape should be rewound after the initialization is complete. You also can indicate *UNLOAD, in which case the tape is rewound and unloaded.

- CLEAR(*NO) states that the tape drive is not to delete the tape of its current contents, which saves time. You also can specify *YES if you want the tape drive to erase the tape for you, but the tape will have to go all the way to the end and back.

**TIP:** If you have a used tape and you want to erase it quickly, the best method is to pass an electromagnet by the tape a few times, then reinitialize the tape with INZTAP CLEAR(*NO). The initialization process takes just a few seconds. You can obtain electromagnets designed for this purpose from computer supply or mail-order stores.

## BACKING UP

Backing up is a very important part of daily AS/400 operations. OS/400 has a different save command for each type of saving operation. Most of these save commands are complicated and have many parameters. This chapter explains the basic operations in enough detail to get you started. Always remember to:

- Have enough initialized tapes to complete your save operation. The system lets you initialize during the save operation, but doing so can be complicated.

- Give *SAVSYS special authority to the users who need to perform any kind of save or restore operation.

Most save commands have a few parameters in common:

- DEV, where you enter the name of the tape device you want to use for this save operation. You can also enter *SAVF if you would rather save to a save file on disk. Save files are described later in this chapter.

- VOL, for the volume ID of the tape you are using. Although you can be specific, you should use *MOUNTED whenever possible. This special value literally means "whatever tape is currently mounted on the drive."

- TGTRLS. If the restore is going to be performed on another AS/400 that is running under an earlier OS/400 release, you must use TGTRLS to indicate which release. If you have OS/400 release V2R1M0 or before, you only can specify *CURRENT or *PRV. *PRV is the release immediately prior to yours. If you are at V2R1M1 or later, you can be specific and indicate something like V1R3M0 if you want to save something to be restored in a V1R3M0 machine. To use this option, you also must make sure that the object, when created in your system, was created with the same TGTRLS parameter in the appropriate CRTXXX command.

- ENDOPT controls what happens to the tape when the save operation is complete. You can choose between *REWIND (the default, which rewinds the tape but keeps it in the drive), *UNLOAD (rewinds the tape and unloads it) and *LEAVE (no rewind: the tape stays in the same spot where it ended saving).

- STG. Do not use STG (*FREE). The *FREE option clears the objects from your system as they are saved. If you save a file with the *FREE option,

your system copies the file to the tape and clears the file as it does so, which leaves you with an empty file.

**TIP:** Backing up is no guarantee that you will be able to continue operations if something goes wrong. You should have a disaster recovery plan in place (refer to the section titled "Disaster Recovery Planning in this chapter). At the very least, you should keep some of your backup tapes off-site, so if there is a fire or a flood, your tapes will still be usable.

### *SAVING LIBRARIES*

You save a library with the Save Library (SAVLIB) command. When you save a library, the system copies the entire library (directory and all objects contained in it) to the tape you have specified. Here are some examples:

```
SAVLIB LIB(FRED) DEV(TAP01)
```

This command saves library FRED using tape device TAP01.

```
SAVLIB LIB(ARLIB APLIB) DEV(TAP01 TAP02)
```

This command saves libraries ARLIB and APLIB, using tape device TAP01. When the tape in TAP01 is full, the system automatically switches to TAP02 while TAP01 rewinds.

```
SAVLIB LIB(*NONSYS) DEV(TAP01)
```

This command saves all nonsystem libraries to TAP01. You need to end all subsystems to run this command.

```
SAVLIB LIB(*ALLUSR) DEV(TAP01) ENDOPT(*UNLOAD)
```

This command saves all user libraries to TAP01. When done, it unloads the tape from the tape drive.

> **TIP:** You can also use Operational Assistant's Backup List to automate the process of backing up your libraries on tape. For more information about Operational Assistant, see chapter 7.

### *SAVING ACCESS PATHS*

You can greatly reduce the amount of time it takes to restore files if you also save the access paths of the files. If you don't save the access paths with the files, the system will rebuild the paths when the file is restored. The users cannot be using the file while the access path is being built. The save will take a little longer, but the time saved when restoring the file will definitely make up for the time lost.

Here is an example of a command to save a library and the access paths of the files in the library:

```
SAVLIB LIB(LIB1) DEV(TAP01) ACCPTH(*YES)
```

### *SAVING INDIVIDUAL OBJECTS*

If it is not necessary, you don't need to save an entire library. You can save individual objects with the Save Object (SAVOBJ) command. Here are some examples:

```
SAVOBJ OBJ(EMPLOYEES) LIB(ACCTG) DEV(TAP01) OBJTYPE(*FILE)
```

This command saves file EMPLOYEES from library ACCTG to tape. Note that the SAVOBJ command does not use qualified names in the OBJ parameter.

```
SAVOBJ OBJ(AR*) LIB(ARLIB) DEV(TAP01) OBJTYPE(*ALL)
```

This command saves all objects (regardless of type) that have names beginning in AR, in library ARLIB to tape.

```
SAVOBJ OBJ(AP001 AP002 AP003) LIB(APLIB) DEV(TAP01) +
OBJTYPE(*DTAQ *DTAARA)
```

This command saves objects AP001, AP002, and AP003 from library APLIB to tape, whether they are data queues or data areas.

```
SAVOBJ OBJ(QRPGSRC) LIB(PGMLIB) DEV(TAP01) OBJTYPE(*FILE) +
FILEMBR((QRPGSRC (INV001RG INV002RG)))
```

This command saves members INV001RG and INV002RG from source file PGMLIB/QRPGSRC to tape. The command doesn't save anything else and can be used for database files, as well.

### *SAVING CHANGED OBJECTS*

One way you can save time when doing your backups is by saving only the objects that have changed since the last time you performed a complete SAVLIB, or since a given date. These methods are referred to as *incremental* backup and *differential* backup, respectively. OS/400 supports these backups with the Save Changed Objects (SAVCHGOBJ) command.

The key to the SAVCHGOBJ command is in two parameters: REFDATE and REFTIME, where you enter the cutoff date and time. If an object has been changed after that date/time combination, it will be saved by the SAVCHGOBJ command.

REFDATE can contain either an actual date such as 050399 or *SAVLIB, which is the default. If you use *SAVLIB, SAVCHGOBJ saves all objects that have changed since the last complete SAVLIB.

REFTIME lets you pinpoint the cutoff time. You can enter an actual time such as 130000 for 1 o'clock in the afternoon, or use the default value, *NONE.

Here are two examples:

```
SAVCHGOBJ OBJ(*ALL) LIB(*ALLUSR) DEV(TAP01 TAP02) +
OBJTYPE(*ALL) REFDATE(*SAVLIB) REFTIME(*NONE)
```

This command saves all objects in all user libraries that have changed since the library was saved with a complete SAVLIB to tape. Note that because you use REFDATE(* SAVLIB), the reftime parameter is irrelevant, and the value is *NONE.

The above command also stipulates that the save operation be performed on the tape currently mounted on tape device TAP01. When the tape ends, the save continues on tape TAP02 while TAP01 rewinds.

```
SAVCHGOBJ OBJ(PR*) LIB(DATA) DEV(TAP01) OBJTYPE(*PGM) +
REFDATE(050399) REFTIME(083000)
```

This command saves all programs in library DATA that have names beginning in PR, if they have been changed since May 3, 1999 at 8:30 in the morning.

### *SAVING WHILE ACTIVE*

In today's business world it is not at all uncommon to need access to the computer 24 hours a day, 7 days a week. This requirement is usually at odds with another requirement of regularly backing up your files.

Normally, saving an object would mandate that you not use the object while you are trying to save it. The save-while-active ability lets you begin the save of the object while the object is still being used. The system will manage the object locks until it has locked all the objects it intends to save. It will then free up the objects for use while it is saving synchronized images.

The save-while-active function takes a picture of the object being saved at a single point in time. This point in time is called a checkpoint. It actually locks the object (*SHRNUP) until it has obtained this checkpoint, and objects cannot be changed until a checkpoint has been reached. The amount of time the objects are

locked to users is called the *save outage*. You use different features of the save-while-active function to either reduce or eliminate the save outage.

> **TIP:** You should end your applications that change objects until after the checkpoint pictures are obtained. The system will send a message when it's reached this point. This will reduce your save outage with the least negative impact to response time and disk storage.

If you are saving multiple objects, you have different options for synchronization. You can indicate that each object being saved is locked until all objects are locked to the same checkpoint time (full synchronization). You could also indicate that all objects in a library are locked at the same time (library synchronization). With this option, different libraries could be locked with different checkpoint times. The third option is called system-defined synchronization, in which different objects can have different checkpoint times. (This option is not for the faint of heart.)

> **TIP:** If you are saving nonjournaled with the save-while-active function, you should not use system-defined synchronization.

There are a number of restrictions placed on the use of the save-while-active functions. You should not use this save feature while the system is very busy or short on disk space. It is a resource-intensive function and will degrade response time of active users while the function is being performed. You shouldn't load, remove, or apply PTFs while running a save-while-active function.

Here is an example of using the save-while-active function embedded in the SAVLIB command. You should first end all jobs that update objects in the libraries being saved. (It is not mandatory that you do this, but it will tremendously reduce the amount of time the save operation takes.)

```
SAVLIB LIB(LIB1 LIB2) DEV(TAP01) SAVACT(*SYNCLIB)+
       SAVACTMSGQ(QSYSOPR) ACCPTH(*YES)
```

If you are considering using the save-while-active functions, you should refer to the *OS/400 Backup and Recovery* manual for an in-depth discussion of the ramifications. For effective and safe use of this feature, you need to plan out your save and restore strategy in advance.

### SAVING THE SYSTEM

It is very important to use the Save System (SAVSYS) command to save the system regularly. The SAVSYS command saves the following items:

- Library QSYS. The save is performed in a format that makes it possible to restore the library using the operating system installation option during an attended IPL.
- All user profiles and the private authorities to objects given to each user.
- All descriptions of devices, controllers, lines, modes, classes of service, connection lists, and network interfaces.
- All office distribution objects, authorization lists, and authority holders you have created.

**TIP:**

The SAVSYS command always should be executed on a blank tape. To avoid any problems, do not save anything else on the tape used by SAVSYS.

Running the SAVSYS command requires a dedicated system. A dedicated system requires that you are the only one signed on and that no other tasks are running. You must end all subsystems. Here is an example of SAVSYS:

```
SAVSYS DEV(TAP01)
```

### SAVING DOCUMENTS AND FOLDERS

Documents and folders reside in a special library called QDOC. You should use the Save Document Library Object (SAVDLO) command to save them to tape. Do not use SAVLIB. Here are a few examples:

```
SAVDLO DLO(*ALL) DEV(TAP01)
```

This command saves all documents and folders onto the tape mounted on drive TAP01.

```
SAVDLO DLO(SMITH.DOC) DEV(TAP01) FLR(DOCUMENT/LETTER)
```

This command saves document SMITH.DOC from folder DOCUMENT/LETTER to tape.

```
SAVDLO DLO(*SEARCH) DEV(TAP01) REFCHGDATE(050399) +
REFCHGTIME(144500)
```

This command searches and saves all documents that have been changed since 2:45 in the afternoon on May 3, 1999. It is similar to the SAVCHGOBJ command.

### *SAVING CONFIGURATION AND SYSTEM VALUES*

Although configuration objects (such as device, controller, and line descriptions) are saved automatically every time you perform a SAVSYS, you can save them independently of SAVSYS with the Save Configurations (SAVCFG) command. You can save the configuration of your system if you have performed many changes (such as installing new display stations and printers) without executing a SAVSYS, which always takes a long time to process and requires a restricted system. Here is an example of SAVCFG:

```
SAVCFG DEV(TAP01) VOL(*MOUNTED) ENDOPT(*UNLOAD)
```

All configuration objects are saved to the tape currently mounted on the TAP01. When the save operation is complete, the tape is unloaded. There is no way to save individual configuration objects; it is an all-or-nothing affair.

System values, on the other hand, cannot be saved automatically by any means because they are not objects. There are two ways to solve this problem:

- Manually maintain a CL program (source code only) in a source file somewhere and call it something meaningful like SETSYSVAL for "set system values." Include a series of Change System Value (CHGSYSVAL) commands in this member. Include one command for each existing system value, even those you have never changed. As you change system values, you must change the corresponding statement in the CL program. You don't need to compile this program because you are not going to run it until you want to restore system values.

- Run the Work with System Values (WRKSYSVAL) command. Specify SYSVAL(*ALL) and OUTPUT(*PRINT). Then keep the printed listing in a safe place. Each time you change a system value, you must rerun the listing.

## RESTORING

Restoring is the process of copying the objects you previously saved to tapes back to the system. When you restore an object of any kind, you are recovering the object as it existed at the time it was saved. If you made some changes after you saved the object, the restored copy will not be up to date. This problem emphasizes the need to save as often as possible.

To restore an object, you use different commands depending on the type of object. This section presents the most important restore operations.

### *RESTORING LIBRARIES*

Restoring a library means restoring the library object (*LIB) itself and all the objects contained in the library at the time you saved the library. To restore a library, use the Restore Library (RSTLIB) command. Here are some examples:

```
RSTLIB SAVLIB(ARLIB) DEV(TAP01)
```

This command restores library ARLIB from the tape mounted in tape drive TAP01. Note that the name of the library, ARLIB, goes in parameter SAVLIB. Although confusing, this is the way the RSTLIB command has been designed. Remember that ARLIB is the name of the library that was saved.

```
RSTLIB SAVLIB(ARLIB) DEV(TAP01) RSTLIB(ARDATA)
```

This command is a variation of the previous one. The tape contains library ARLIB (the saved library, SAVLIB), but you want to restore the library into the system (RSTLIB) with the name ARDATA. In other words, library ARLIB is restored with a different name. When the RSTLIB command ends, you will have a library called ARDATA in your system.

```
RSTLIB SAVLIB(ARLIB) DEV(TAP01) OPTION(*ALL)
```

This command is identical to the first one because the OPTION parameter defaults to *ALL. All objects in the saved library, ARLIB, are restored to your system, whether or not your system's ARLIB library already contains the objects.

In other words, your system has a library named ARLIB on disk that contains file A but does not have file B. The version of ARLIB you have on the tape contains both files, A and B. When you restore ARLIB with OPTION(*ALL), both files are restored. File A is overwritten and file B is created.

```
RSTLIB SAVLIB(ARLIB) DEV(TAP01) OPTION(*NEW)
```

This command operates like the previous one except that only file B would be restored because A already exists in your system's ARLIB. The OPTION parameter also can have value *OLD (its meaning is opposite of *NEW) and *FREE. *FREE is used only when you saved a library with STG(*FREE), which is usually not recommended.

```
RSTLIB SAVLIB(*NONSYS) DEV(TAP01 TAP02)
```

This command restores all nonsystem libraries contained on the tape mounted in TAP01. When that tape is exhausted, the system automatically continues with TAP02 while TAP01 rewinds.

### RESTORING INDIVIDUAL OBJECTS

You don't have to restore a whole library if you need only a few of the objects contained in it. With the Restore Object (RSTOBJ) command you can restore individual objects.

Here are a few examples:

```
RSTOBJ OBJ(*ALL) SAVLIB(ARLIB) DEV(TAP01) OBJTYPE(*DTAARA)
```

This command restores all data areas that were saved from library ARLIB. The system uses the tape drive TAP01 for the restore.

```
RSTOBJ OBJ(AR001CL AR001RG) SAVLIB(ARLIB) DEV(TAP01) +
OBJTYPE(*ALL)
```

This command restores objects AR001CL and AR001RG (no matter what the type) from the tape mounted in TAP01.

```
RSTOBJ OBJ(QRPGSRC) SAVLIB(PGMLIB) DEV(TAP01) +
OBJTYPE(*FILE) FILEMBR((QRPGSRC (MBR1 MBR2))) +
MBROPT(*ALL)
```

This command restores members MBR1 and MBR2 from file QRPGSRC, as saved in library PGMLIB on tape device TAP01.

### RESTORING DOCUMENTS AND FOLDERS

Use the Restore Document Library Object (RSTDLO) to restore documents and folders from tape. Here are two examples:

```
RSTDLO DLO(*ALL) DEV(TAP01)
```

This command restores all documents and folders previously saved on the tape mounted in TAP01.

```
RSTDLO DLO(FRANKIE.DOC) DEV(TAP01) SAVFLR(*ANY) +
RENAME(FRANK.DOC) RSTFLR(MEMOS/PERSONAL)
```

This command assumes that the tape in TAP01 contains a document named FRANKIE.DOC (in any folder) that you want to restore. Once the document is restored, however, its name is changed to FRANK.DOC, and it is placed in folder MEMOS/PERSONAL.

### *RESTORING CONFIGURATION OBJECTS*

If you have a tape where you have saved the configuration objects (such as device descriptions and controllers), either with the SAVSYS or SAVCFG command, you can restore individual configuration objects from that tape using the Restore Configuration (RSTCFG) command. For example:

```
RSTCFG OBJ(DSP32 DSP35 PRT02) OBJTYPE(*DEVD) ENDOPT(*UNLOAD)
```

This command restores the device descriptions DSP32, DSP35, and PRT02. When the restore is completed, the tape is unloaded.

```
RSTCFG OBJ(*ALL) ENDOPT(*UNLOAD) OUTPUT(*PRINT)
```

This command restores all configuration objects from the tape. It also prints a report that lists all objects that were successfully restored, those that were not restored, and those that were excluded.

### *RESTORING SYSTEM VALUES*

Restoring system values is impossible. The previous section describes two methods to "save" system values: by writing a CL program that contains a series of CHGSYSVAL commands or by printing a list of the values and saving the printout.

- If you maintained the CL program, all you need to do is compile and execute it. When the program ends, you should consider IPLing the system to make sure that all system values have become effective. Some system values like QSECURITY require an IPL before they activate.

- If you opted for the printed list, run the WRKSYSVAL command and enter option 2 for all system values to change them. Then key in the values shown on the listing. When you are done, IPL the system.

## DISASTER RECOVERY PLANNING

No one is safe from disaster. A disaster can be a fire that burns your company building to the ground, a flood, an earthquake, a major system failure, or even sabotage.

Backing up is not enough to protect yourself from disaster. In the event of a fire or earthquake, having tapes with everything you ever had on the system doesn't do you any good if you don't have a system to restore them to. Therefore, you must devise a safe and sound disaster recovery plan.

A discussion of such a plan (or even a complete definition) is beyond the scope of this book. There are firms whose primary business is providing a disaster recovery plan for computer users. You should contact one of them immediately.

A well-implemented disaster recovery plan can offer you a replacement system that will be available if your system becomes unusable. The disaster recovery supplier charges a fee for this service, but you can look at it as a form of insurance. The provider works with you and helps you design a good disaster recovery plan.

**TIP:** Once you have a plan in place, follow the guidelines religiously and test it thoroughly. Sometimes you think the plan is foolproof, only to later discover that there are holes.

## USING SAVE FILES

When you perform save and restore operations, you can use save files instead of tapes. Save files are objects of type *FILE that reside on disk. They offer faster backups than those provided by tapes because a save operation to a save file is a copy from disk to disk. In addition, they don't require operator intervention to mount the next tape when one ends.

On the other hand, save files use up (at least temporarily) space on your disk drives.

### *USING SAVE FILES*

To use a save file instead of tape, you must create the save file first using the Create Save File (CRTSAVF) command. For example:

```
CRTSAVF FILE(MYLIB/MYSAVF) TEXT('Save file for backups')
```

To save to the save file, execute the usual save command, except specify DEV(*SAVF) and the name of the save file in the SAVF parameter. Note that the SAVSYS command does not accept DEV(*SAVF). For example:

```
LIB(ACCTG) DEV(*SAVF) SAVF(MYLIB/MYSAVF)
```

This command saves the entire ACCTG library in save file MYSAVF in library MYLIB. It goes into a compressed format; it is not a byte-by-byte copy.

At your leisure, some time after this SAVLIB command ends, you can save the save file to tape and clear the save file, which frees precious disk space:

```
SAVF(MYLIB/MYSAVF) DEV(SYSTAP01) FILE(MYLIB/MYSAVF)
```

If you need to restore the ACCTG library from this tape, you use the regular RSTLIB command. The system will not care that you used a save file in the process.

If you can afford the space on your disk, you can leave the save file full of data and never perform the backup to tape. In this case, the RSTLIB command would have to specify DEV(*SAVF) SAVF(MYLIB/MYSAVF). However, with this method you do not have an external backup that you can keep in a safe place. If your disk crashes and needs to be replaced, the backup in the save file will be lost too.

## AS/400 & S/36 MEDIA EXCHANGE

As explained in chapter 25, the AS/400 includes the System/36 Execution Environment (abbreviated S/36EE or S/36E), which lets you administer, operate, and program the AS/400 as if it were an S/36.

Even though the S/36E provides the real S/36 procedures to save and restore, you will not be able to save a file on the S/36 and restore it on the AS/400's S/36E with the predictable SAVE and RESTORE procedures. It will not work.

### *FROM S/36 TO AS/400*

On the S/36 side, use the usual commands such as SAVE and FROMLIBR. On the AS/400 side, however, you must use the native commands:

- RSTS36F: Restores an S/36 file that was backed up by the S/36 with the SAVE procedure.

- RSTS36LIBM: Restores one or more S/36 library members that were backed by the S/36 with the fromlibr procedure.

### *FROM AS/400 TO S/36*

The opposite also is true. You must save the files or library members with the AS/400 native commands:

- SAVS36F: Saves an AS/499 file to tape or diskette in a format that will make it possible for the S/36 to restore it with the RESTORE procedure.

- SAVES36LIBM: Saves one or more library members, as defined in the S/36E, to tape or diskette. This tape or diskette can then be processed by the S/36 with the TOLIBR procedure.

## FURTHER READING

IBM's *Backup and Recovery Guide.*

Midrange Computing's *The AS/400 Owner's Manual.*