System i

Backing up your system

Version 6 Release 1

System i

Backing up your system

Version 6 Release 1

Note

Before using this information and the product it supports, read the information in "Notices," on page 187.

This edition applies to version 6, release 1, modification 0 of i5/OS (product number 5761-SS1) and to all subsequent releases and modifications until otherwise indicated in new editions. This version does not run on all reduced instruction set computer (RISC) models nor does it run on CISC models.

Contents

Backing up your system

The method that you use to back up your system depends on your backup strategy. If you do not have a strategy, you need to plan a backup and recovery strategy. After reviewing the information, determine how you should save your data. Use the GO SAVE menu commands or individual Save commands to back up your system.

Simple strategy

If you choose a simple strategy you can use the GO SAVE command to back up your system. The Save menu options of the GO SAVE command provide an easy method to back up your system. These Save menu options include option 21 to save your entire system, option 22 to save your system data, and option 23 to save your user data. Each of these options requires that your system be in a restricted state. This means that no users can access your system, and the backup is the only thing that is running on your system.

Use the GO SAVE command, menu option 21, to save your entire system. Then you can use the other GO SAVE command menu options to save the parts of your system that change regularly. In addition, you can use a variety of other save commands to save individual parts of your system.

If you choose a simple save strategy, review Overview of the GO SAVE command to see what parts of your system GO SAVE command, menu options 21, 22, or 23 save. Then skip to the topic, Preparing your media to save your system.

Medium and complex strategy

To help you get started with a medium or complex strategy follow these steps:

1. Draw a picture of your system similar to the one in Save commands and menu options. In your picture, break the section called **User Libraries** into smaller segments that match the way you plan to save user libraries.
2. Study the information in the Overview of the GO SAVE command and in Manually saving parts of your system topics.
3. Determine how and when you plan to save each part of your system.

If you do not have time to do a full save operation, you can save your system while it is active. However, you must have a complete backup of your entire system (which requires a restricted state) before you use these advanced functions.

Note: By using the code examples, you agree to the terms of the “Code license and disclaimer information” on page 186.

Related information

Backup and recovery FAQ

Planning a backup and recovery strategy

What’s new for V6R1

Read about new or significantly changed information for the Backing up your system topic collection.

Encrypted backups

You can encrypt backups to tape media to prevent the loss of personal customer information or confidential data if the media is lost or stolen. You can perform encrypted backups using either method:

- "Encrypting tape drive" on page 23 using save/restore commands or Backup, Recovery, and Media Services (BRMS).
- "Software encryption using BRMS" on page 24.

Master keys are used to encrypt other keys. If a master key is lost, all keys encrypted under that master key, and consequently all data encrypted under those keys, are lost. Back up the master keys both by saving the passphrases, and by using a Save System (SAVSYS) or GO SAVE Option 21 or Option 22 backup operation. To protect the master keys while on the save media, they are encrypted with the save/restore master key. For more information, see "Encrypted backups" on page 154.

Saving and restoring user-defined file systems

Previously, you had to unmount user-defined file systems (UDFSs) before performing a save operation if you wanted to save the file-system attributes that defined the file systems. Because the file-system attributes are saved with a mounted UDFS, it is easier to save and restore mounted UDFSs now.

You can use the OBJ or PATTERN parameter on the SAV and RST commands to omit objects from unmounted UDFSs. For more information, see these topics:
- "Saving an unmounted UDFS" on page 88.
- "Saving a mounted UDFS" on page 88.

Saving and restoring private authorities

You can now save and restore private authorities for an object by specifying the PVTAUT(*YES) parameter on the SAV*xx* and RST*xx* commands. For more information, see "Saving security data" on page 57.

Synchronizing multiple save-while-active operations

Use the STRSAVSYNC command to fully synchronize multiple save-while-active operations. Because the backup data is synchronized, all of the data is saved at a single point in time and represents a consistent view of all the data. See these topics for more information:
- "Full synchronization" on page 133.
- "Using save-while-active to synchronize the saved data" on page 119.

Integrated server backup enhancements

You can use the SAV command to back up integrated Windows® or Linux® server files and directories. See these topics for more information:
- "Saving individual files on integrated servers" on page 107.
- "Saving Linux data on a logical partition" on page 107.

Use any of the following methods to back up data from an integrated server:
- Using a program from the operating system to save objects, such as a Windows or Linux program.
- Using i5/OS® to save configuration objects and network server storage spaces.
- Configuring file-level backups for integrated Windows or Linux servers.

See these topics for more information:
- "Saving data for iSCSI-attached integrated servers" on page 106.
- "Saving data for IXS and IXA-attached integrated Windows servers" on page 106.

You can save network server storage spaces for an integrated Windows or Linux server without having to shut down the system. This function enables your system to be available during the save operation. See "Methods for saving network server storage spaces" on page 97.

Miscellaneous save/restore enhancements

You can now save journaled libraries using the SAVLIB command and restore journaled libraries using the RSTLIB command. See these topics for more information:
- "Saving changed objects when you use journaling" on page 77.
- "Saving journaled objects and libraries" on page 76.

The maximum size of a save file has doubled from 1 TB (where TB equals 1 099 511 627 776 bytes) to approximately 2 TB. See "Size limits for save files" on page 7.

i5/OS no longer supports NetWare Enhanced Integration for backing up Novell data. Use IBM® Tivoli® Storage Manager to back up and restore your Novell data.

How to see what's new or changed

To help you see where technical changes have been made, the information center uses:
- The  image to mark where new or changed information begins.
- The  image to mark where new or changed information ends.

In PDF files, you might see revision bars (|) in the left margin of new and changed information.

To find other information about what's new or changed this release, see the Memo to users.

PDF file for Backing up your system

You can view and print a PDF file of this information.

To view or download the PDF version of this document, select Backing up your system (about 2.2 MB).

You can view or download these related topic PDFs:
- Backup and recovery frequently asked questions
- Backup, Recovery, and Media Services (BRMS) (about 584 KB)
- Disk management (about 2.7 MB)
- Planning a backup and recovery strategy (about 317 KB)
- Recovering your system (about 6.3 MB). This topic also is available as a printable manual (SC41-5304).
- Storage solutions (about 2 MB)

Other information

You can also view or print any of the following PDFs:
- Manuals:

  Backup, Recovery, and Media Services for i5/OS  (about 2559 KB). This manual provides information about how to install and use Backup, Recovery, and Media Services (BRMS) CL commands.
- IBM Redbooks™:

  A Practical Approach to Managing Backup, Recovery, and Media Services for OS/400® 

Saving PDF files

To save a PDF on your workstation for viewing or printing:

1. Right-click the PDF link in your browser.

2. Click the option that saves the PDF locally.
3. Navigate to the directory in which you want to save the PDF.
4. Click **Save**.

Downloading Adobe Reader

You need Adobe® Reader installed on your system to view or print these PDFs. You can download a free copy from the Adobe Web site (www.adobe.com/products/acrobat/readstep.html) .

Before you save any data

Read this information to prepare for saving data on your system.

Using the precheck option

Use the precheck option explains how to have the system check certain criteria on each object that you save on a library-by-library basis. This option is not required.

You can use the precheck (PRECHK) parameter when you save objects to ensure that all of the objects you intend to save can be successfully saved. If you specify `PRECHK(*YES)`, the system verifies that the following are true of each object that you are saving on a library-by-library basis:

- The object can be allocated during the save operation. No other job has a conflicting lock on the object.
- The object exists.
- The object is not marked as damaged. The precheck process looks only for damage that has already been detected. It does not detect new damage to the object header or damage to the contents.
- All members of an object can be allocated if the object is a database file.
- The person that requests the save operation has sufficient authority to save the object.

When you specify PRECHK(*YES), all of the objects you are saving in a library must meet the conditions. If they do not, no objects in the library are saved. If you specify more than one library on the save command, the failure of one library to meet the PRECHK tests does not typically prevent the system from saving other libraries. However, if you specify SAVACT(*SYNCLIB), the entire save operation stops if one object fails the precheck process.

When you specify PRECHK(*NO), the system performs the checking on an object-by-object basis. The system bypasses any object that does not meet the conditions, but the save operation continues with other objects in the library.

Related concepts

"Library synchronization" on page 133
All objects in a library reach a checkpoint at the same time. But different libraries reach checkpoints at different times. This option might be useful if all of the following are true.

Choosing compression type

You can use compression and other capabilities to improve save performance and also use less media for your save operation.

Data compression compresses data on the media when you perform the save operations. Data decompression reconstructs data when you perform a restore operation. The system ensures that information saved can be reconstructed exactly. No data is lost as a result of compression and decompression.

The two main types of compression are hardware compression and software compression. Most tape media devices use hardware compression, which is normally faster than software compression. Software compression takes considerable processing unit resources and might increase your save and restore time.

In addition to data compression, you can use compaction and optimum block size features to streamline your save operation. These features are available through parameters on all save commands:

- Data Compression (DTACPR)
- Data Compaction (COMPACT)
- Use Optimum Block Size (USEOPTBLK)

You can see examples of the parameter values in the SAVSYS command description.

If you save to save files or optical media, you also have three choices available for software compression: low, medium, and high. If you choose a higher form of compression, your save will take longer, but the resulting save data will usually be smaller. The following choices are available on the Data Compression (DTACPR) parameter of the save commands and through the Save Object (QsrSave) and Save Object List (QSRSAVO) APIs:

- **Low:** This is the default form of compression for save files and optical media. Low compression is usually faster than medium or high compression. The compressed data is usually larger than if medium or high compression is used.
- **Medium:** This is the default form of compression for optical-DVD media. Medium compression is usually slower than low compression but faster than high compression. The compressed data is usually smaller than if low compression is used and larger than if high compression is used.
- **High:** This form of compression is meant to be used when maximum compression is desired. High compression is usually noticeably slower than low and medium compression. The compressed data is usually smaller than if low or medium compression is used.

You can also find more information about compression, compaction, and optimum block size in System i™ Performance Capabilities Reference  . The “Saves and restores using save files” chapter contains information about the compression ratios for *LOW, *MEDIUM, and *HIGH compression.

Related concepts

“Save files” on page 12
Understand what save files are and how to use them in your save and restore operations.

Related information

Storage Solutions

Freeing storage when saving

Freeing storage when saving explains how to use the STG parameter to remove an object from your system after you save it. This only works with a limited number of commands.

Normally, saving an object does not remove it from the system. However, you can use the storage (STG) parameter on some save commands to free some of the storage that is used by saved objects.

If you specify STG(*FREE), the object description and search values remain on the system. The system deletes the contents of the object. You can perform operations such as moving and renaming an object whose storage you freed. However, you must restore the object to use it.

You can use the STG(*FREE) parameter for the object types in the following table:

Table 1. Object types that support freeing storage

Object Type	Description
*FILE[1,2]	Database files
*STMF[3]	Stream files
*JRNRCV[4]	Journal receivers
*PGM[5]	Programs
*DOC	Documents
*SQLPKG	SQL packages
*SRVPGM	Service programs
*MODULE	Modules

[1] When you free a database file, the system frees the storage that is occupied by the data portion of the object, but the object description remains on the system. If you save a database file that has already been freed and free its storage, the system does not save the object description and you receive the following message:

```
CPF3243 Member xxx already saved with storage freed
```

If you install the Media and Storage Extensions product on your system, and you save a database file and free its storage, the system saves the object description.

[2] The system does not free the storage occupied by logical file access paths.

[3] You can free storage for *STMF objects, but not during a save operation. Free the storage for *STMF objects with the Save Storage Free "Qp0lSaveStgFree() API".

You can save an *STMF object whose storage has already been freed, but you must restore the *STMF object before you can use it.

[4] You can free storage for a journal receiver if it is detached and all previous journal receivers are deleted or have their storage freed.

[5] Do not specify STG(*FREE) for a program that is running. This causes the program to end abnormally. For Integrated Language Environment® (ILE) programs, the program does not end abnormally. The system sends a message that indicates that the system did not save the ILE program.

You can also specify STG(*DELETE) on the Save Document Library Object (SAVDLO) command. This deletes any filed documents after the system saves them. This includes the object description, the document description, the search values, and the document contents.

Related concepts

"Methods for reducing disk space that is used by documents" on page 91
Documents tend to accumulate and require more and more storage. This information describes different methods that you can use to reduce disk space that is used for documents.

Related information

Qp0lSaveStgFree()

How object locking affects save operations

The system locks an object to prevent an update operation while the system saves it.

If the system cannot obtain a lock on an object within the specified time, the system does not save that object and the system sends a message to the job log. The save-while-active function shortens the time during which the system locks an object while saving.

Save-while-active object locking rules shows the type of lock the system must obtain successfully to save an object or to establish a checkpoint for the object for save-while-active processing.

When you specify multiple libraries for a save procedure, the system locks the libraries that you specified and the libraries are unavailable for use during the save operation. Some or all of the libraries might be unavailable for use at any given moment.

Size limitations when saving objects

This topic provides information about the size limitations when saving document library objects (DLOs).

When you perform a save operation, the system creates a list of the objects and their descriptions that it saves. The system saves this list with the objects for use when the system displays the save media or restores the objects. The system limits a single list of saved objects. Because the system creates multiple lists for each library that you save, the limits are rarely exceeded.

There are limits on the number of objects you can save from a single library. Because you normally store document library objects (DLOs) in libraries, this limit applies to the QDOC library in the system auxiliary storage pool (ASP) and the QDOC*nnnn* libraries in user ASPs.

If your save operation fails because you exceed any of these limits, you need to save objects using separate save commands instead of saving them with a single command.

The Save and restore limits topic shows the limits that apply to save and restore operations.

Related reference

"Saving objects with the SAVOBJ command" on page 67
Use the Save Object (SAVOBJ) command to save one or more objects on your system. You can also use the QSRSAVO API to save multiple objects.

Related information

Save and restore limits

Size limits for save files

Size limits for save files are 4 293 525 600 records. At 512 bytes per record, the maximum size of a save file is approximately 2 TB (where TB equals 1 099 511 627 776 bytes).

You can specify only one library when your output media for the save procedure is a save file. When saving document library objects (DLOs), you can specify only one auxiliary storage pool (ASP) when your output media is a save file.

Related information

Restrictions for current release-to-previous release support

Verifying what the system saved

Verify what the system saved explains techniques to audit your save strategy. You will learn which objects the system saved, which objects the system did not save, and when the system last saved an object.

You can use the job log or an output file to determine which objects the system saved successfully.

Determining objects that the system saved (save messages)

This information describes how save messages work and what information is available from the output files.

Save messages show the number of objects that the system saved. The message help of the completion message includes the volume identifiers of the first 75 volumes of save media that the system used. The system uses these identifiers to update the status information of each object that the system saved. The message data contains this information, the last volume ID, and either the last device that the system used or the save file that the system used.

Note: The system performs overlap processing during normal save operations. The system can write some libraries to the media while the system preprocesses other libraries. Occasionally the job log contains preprocessing and completion messages that appear in a different order than the order in which the system wrote libraries to the media.

If a single command saves multiple libraries, a final completion message (CPC3720 or CPC3721) also contains the last device that the system used.

Note: The output file that you specify is in use throughout the save operation. Therefore, the system cannot save it as part of the operation. Depending on how you perform your save operation, you might see a CPF379A message in the job log for the output file. If you want to save the output file after your save operation has completed, use the SAVOBJ command.

These are some messages that you might see during the verification process:

Message CPF3797: Objects from library <your library name> not saved. Save limit exceeded.

Message CPC3701: Sent for each library that is saved to media.

Message CPC3718 : Completion message for SAVSYSINF command.

Message CPC3722: Sent for each library that is saved to a save file.

Message CPC9410: Completion message for SAVDLO command to media.

Message CPC9063: Completion message for SAVDLO command to save file.

Message CPC370C: Completion message for SAV command to media.

Message CPC370D: Completion message for SAV command to save file.

Related concepts

"Interpreting output from save (SAV) and restore (RST)" on page 159
When you use the Save (SAV) command or the Restore (RST) command, you can direct output to a stream file or to a user space.

Related reference

"Interpreting output from save commands" on page 176
This topic contains a list of links to save commands or APIs that you can use to direct output to an output file.

Determining objects that are not saved

Determining the objects that are not saved is just as important as determining the objects that the system saved. The system might not save an object for two basic reasons.

- The object is not in your save plan. For example, you save libraries individually. You add a new application with new libraries, but forget to update your save procedures.
- The object is in your save plan, but the system did not successfully save it. The system might not save an object for any of the following reasons:
  - It is in use. If you use the save-while-active function, the system waits a certain amount of time to obtain a lock on the object. If you do not use the save-while-active function, the system does not wait.
  - The system marked the object as damaged.
  - You do not have the necessary authority to the object.

When the system cannot save an object, the system skips that object and writes an entry to the job log. Verifying the job logs that the system creates by your save procedures is very important. If you have very large save operations, you might want to develop a program that copies the job log to a file and analyzes it.

You can specify OUTPUT(*OUTFILE) INFTYPE(*ERR) on the SAVLIB, SAVOBJ, and SAVCHGOBJ commands. This creates an output file that only contains entries for those objects that the system did not save. Refer to the online command help for more information about the specific command.

Periodically verify your backup strategy by the following methods:

- Review when the system saves objects.
- Determine when the system saved the changes that were made to these objects.

Use the information in the object description to determine when the system last saved the object. Base your method for doing this according to your save strategy. If you save entire libraries, you can verify the save date for every library on the system. If you save individual objects, you need to verify the save date for objects in all user libraries.

To verify save dates for libraries, you can do the following:

1. Create an output file that has information about all the libraries by typing:

```
DSPOBJD OBJ(QSYS/*ALL) OBJTYPE(*LIB) +
        OUTPUT(*OUTFILE) +
        OUTFILE(library-name/file-name)
```

2. Use a query tool or a program to analyze the output file. The field ODSDAT contains the date that the object was last saved. You can sequence your report by this field or compare this field to some date in the past.

You can use a similar technique to check when the system last saved objects in a specific library.

Determining when an object was last saved

If a library contains an object, you can use the Display Object Description (DSPOBJD) command to find out when the system saved the object.

If the QSYS library contains an object, you can use the DSPOBJD command to display the appropriate data area that is shown in Data areas that contain save history.

You can also use the DSPOBJD command to obtain the save history for document library objects (DLO) in libraries. Use the Display Document Library Object Name (DSPDLONAM) command to find the system object name and the ASP ID of the DLO. On the DSPOBJD command, specify the system object name on the OBJ parameter. In the library name field, specify QDOCxxxx where xxxx is the ASP ID. For example, for auxiliary storage pool (ASP) 2 the library name is QDOC0002.

Note: For ASP 1, the system ASP, the library name is QDOC, not QDOC0001.

For objects that you store in directories, you can use the output from the SAV command to maintain save history information. To use the output, you must elect to keep the save history information when you issue the SAV command. To keep the save history information, specify either *PRINT or a stream file or user space path name on the OUTPUT parameter of the SAV command.

Note: The output from the SAV command does not store the last saved data for objects in directories. See Save changed objects in directories for instructions to save only changed objects.

The following commands do not update the save history information for the individual objects that the system saves:

- Save System (SAVSYS)

- Save Security (SAVSECDTA)
- Save Configuration (SAVCFG)
- Save Save File Data (SAVSAVFDTA)
- Save System Information (SAVSYSINF)

For some save operations, the system updates history information in a data area. In some cases, the system updates the data area instead of updating the individual objects. In other cases, the system updates the data area in addition to the individual objects.

When you install the operating system, the system will update the data areas. However, the data areas will appear as if you used RSTOBJ to restore them. The system does not support the QSAVDLOALL data area.

The QSRSAV21 data area in library QUSRSYS contains information about the last five most recent GO SAVE Option 21 (Save Entire System) operations. The information includes the starting date and time of each major step in the save operation, the step identifier, and the device used. The ending date and time of the save operation is identified with asterisks. You can use this information to help you plan how much time to estimate for your next GO SAVE Option 21 operation.

The following table shows these commands and the associated data areas:

Table 2. Data areas that contain save history

Command	Associated Data Area	Individual Objects Updated?
SAVCFG	QSAVCFG	No
SAVLIB *ALLUSR	QSAVALLUSR	Yes[1]
SAVLIB *IBM	QSAVIBM	Yes[1]
SAVLIB *NONSYS	QSAVLIBALL	Yes[1]
SAVSECDTA	QSAVUSRPRF	No
SAVSTG	QSAVSTG	No
SAVSYS	QSAVSYS, QSAVUSRPRF, QSAVCFG	No
SAVSYSINF	QSYSINF	No
GO SAVE Option 21	QSRSAV21	No

[1] If you specify UPDHST(*NO), the system does not update the *Date last saved* field in either the object or the data area.

The system uses the save history information when you save objects that have changed since the last save operation.

Related reference

"Saving only changed objects" on page 69
You can use the save changed object function to reduce the amount of save media that you use. You can also complete your save process in a shorter period of time.

How the system handles damaged objects during a save operation

When the system encounters a damaged object during a save operation, it does one of several things based on when it detected the damage. This information also describes error messages that you might see during a save operation.

Object that the system marked as damaged before the save operation

The system does not save an object that it marked as damaged, but the save operation continues with the next object. The operation completes with an indication of how many objects the system saved and how many it did not save. Diagnostic messages describe the reason that the system did not save each object.

Object that the save operation detects as damaged

The system marks the object as damaged, and the save operation ends. The save operation ends because the save media might contain part of the damaged object. If the media contains a damaged object, the save media cannot be used for restore operations. The system sends diagnostic messages.

Object that the system does not detect as damaged

In some unusual cases, a save operation does not detect a damaged object. The save operation might detect physical damage on the disk, but it might not detect all damage. For example, the system does not attempt to determine if all bytes within an object are valid and consistent (logical damage). For some cases, you will not be able to determine a damage condition unless you attempt to use the object (such as calling a program object). If this type of damage exists, the system restores the object normally.

Preparing your media to save your system

Use this information to select and manage the save media that you will use for all your save functions.

Managing your tapes and other media is an important part of your save operation. If you cannot locate the correct and undamaged tapes and other media that you need to do a recovery, your system recovery is more difficult. Here is a list of the save media types:

- Magnetic tape
- Optical media
- Virtual optical
- Save file
- Virtual tape

Successful media management involves making decisions about how to manage your media, writing down those decisions, and monitoring the procedures regularly.

Related information

BRMS

Choosing your save media

Learn about the different types of media that can be used for save and restore operations as well as which save and restore commands can be used with the different types of media.

Tape is the most common media that is used for save and restore operations. You can also save your user data and your system data to optical media.

The table below shows which save and restore commands support which types of media.

Table 3. Media used with the Save commands

Command	Tape	Virtual tape	Optical media	Virtual optical	Save file
SAVSYS	Yes	Yes[4]	Yes[1]	Yes[4]	No
SAVCFG	Yes	Yes	Yes	Yes	Yes
SAVSECDTA	Yes	Yes	Yes	Yes	Yes
SAVLIB	Yes	Yes	Yes[2]	Yes	Yes
SAVOBJ	Yes	Yes	Yes	Yes	Yes
SAVCHGOBJ	Yes	Yes	Yes	Yes	Yes
SAVDLO	Yes	Yes	Yes[3]	Yes	Yes
SAVSAVFDTA	Yes	Yes	Yes	Yes	No
SAVLICPGM	Yes	Yes[4]	Yes[1]	Yes[4]	Yes
SAVSTG	Yes	No	No	No	No

Table 3. Media used with the Save commands (continued)

Command	Tape	Virtual tape	Optical media	Virtual optical	Save file
SAV	Yes	Yes	Yes	Yes	Yes
RUNBCKUP	Yes	Yes	No	No	No
SAVSYSINF	Yes	Yes	Yes	Yes	Yes

[1] You cannot run this command on an optical media library device.

[2] You can specify SAVLIB LIB(*ALLUSR), SAVLIB LIB(*IBM), or SAVLIB LIB(*NONSYS) when you use optical media. However, you need to initialize your optical media to the *UDF format. You cannot use optical media that you initialized to *HPOFS format.

[3] You can save document library objects (DLO) from more than one auxiliary storage pool (ASP) to optical media with a single SAVDLO command. However, you need to initialize your optical media to the *UDF format. You cannot use optical media that you initialized to *HPOFS format.

[4] In a disaster recovery situation you must have physical media of the Licensed Internal Code and the operating system to begin your recovery.

Related information

Storage Solutions

Save files

Understand what save files are and how to use them in your save and restore operations.

Using a save file allows you to save and restore objects without first placing save media into your save media device. You can also use a save file to send objects from one System i environment to another over communications lines. You can use the save file as an online container to save the contents of a single library to run overnight. The next day, save the contents of the save file to storage media with the Save Save File Data (SAVSAVFDTA) command. Objects saved to media using the SAVSAVFDTA command can be restored directly from save media, using the RSTLIB, RSTOBJ, or RST command.

A few things to consider when saving to save files are:

- Only one library can be saved to a save file.
- You cannot save or send a save file that is larger than the target release allows.
- Performance can vary, depending on other disk activity. Save files can be created on or moved to an ASP for improved performance and additional protection from system disk unit failures.
- The maximum capacity of a save file is approximately 2 TB (where TB equals 1 099 511 627 776 bytes). You can specify the maximum size of the save file on the Create Save File (CRTSAVF) command.

Remember to specify data compression on the save commands to reduce the space for the save file and the amount of media needed for the SAVSAVFDTA command. (Data compression is not an option on the SAVSAVFDTA command.)

If you are using virtual I/O storage, which supports virtual disk, optical, and tape devices, you can write a save file to a virtual disk.

Related concepts

"Choosing compression type" on page 4
You can use compression and other capabilities to improve save performance and also use less media for your save operation.

Copying save files to media:

You can back up parts of your system to a save file on disk rather than removable save media. However, you should save the save file to removable media on a set schedule.

You can save the contents of your save file by two different methods. You can use the Save Save File Data (SAVSAVFDTA) command to save your save file data as if your objects were saved directly to media. Or, you can use the Save File Data (SAVFDTA) parameter to save the entire save file to media.

Save Save File Data (SAVSAVFDTA) command

Use the Save Save File Data (SAVSAVFDTA) command to save objects that appear on the media as if the system saved them directly to the media. For example, assume that you use the following commands to save a library:

```
SAVLIB LIB(LIBA) DEV(*SAVF) SAVF(LIBB/SAVFA)
SAVSAVFDTA SAVF(LIBB/SAVFA) DEV(media-device-name)
```

You can restore library LIBA either from the media volume or from the save file by using the RSTLIB command. When you use the SAVSAVFDTA command, the system does not save the save file object itself.

Save file data (SAVFDTA) parameter

Use the save file data (SAVFDTA) parameter on the SAVLIB command, the SAVOBJ command, or the SAVCHGOBJ command. When you specify SAVFDTA(*YES), the system saves the save file and its contents to save media. You cannot restore individual objects that are in the save file from the media copy of the save file. You must restore the save file and then restore the objects from the save file.

The following restrictions apply when specifying SAVFDTA(*YES):

- If you are saving the save file for a system at a previous release, the system saves the save file in a previous release format. The objects within the save file remain in the release format that was specified when they were saved to the save file.
- If the save media for the save operation is the same save file, the system only saves the description of the save file. The system sends message CPI374B, `SAVFDTA(*YES) ignored for file <your-file-name> in library <your-library-name>`, and the save operation continues.

Working with save files:

You can use the CL commands that are listed here with save files.

- The Create Save File (CRTSAVF) command creates a save file that can be used with save and restore commands to store data. The save file stores data that might otherwise be written to save media. You can use FTP to send a save file to another System i user on the network.
- The Change Save File (CHGSAVF) command changes one or more of the attributes of a save file, such as the maximum number of records.
- The Override with Save File (OVRSAVF) command overrides or replaces certain attributes of a save file, or overrides any file with a save file.
- The Display File Description (DSPFD) command displays the attributes of the save file.
- The Clear Save File (CLRSAVF) command clears the contents of a save file.
- The Display Save File (DSPSAVF) command displays the save and restore information in a save file, or the contents of the save file.
- You can use the Save Object (SAVOBJ) or the Save Library (SAVLIB) command to save the description of the save file. You can also save the data to tape, optical media, or another save file in a different library.
- The Save Save File Data (SAVSAVFDTA) command writes the contents of a save file to either tape or optical media.

Use the following API to work with save files:

The List Save File (QSRLSAVF) API returns the contents of the save file in a user space. The contents of the save file is returned at a user-selected level of library information, object information, member

information, or spooled files. The QSRLSAVF API returns the same information that is shown on a DSPSAVF command. In addition, when you specify the SAVF0200 format, the system includes the following:

- The serial number of the system on which the save operation was performed.
- The ASP from which the object was saved.

The QSYSINC library provides structures for the QSRLSAVF API formats in C, COBOL, and RPG.

About save file security:

The authority you grant for a save file is the same as for any file. Be careful when granting authority for save files. The authority you grant to the save file allows access to objects in the save file.

For example, the same file can be read from and written to by a high-level language program. The authority you grant for a particular save file should depend on what objects are in the file.

Consider the following factors when granting authorities to save files:

- A user with use (*USE) authority can read records and restore objects from the save file. This user can save the contents of the save file to tape or optical media.
- A user with use (*USE) and add (*ADD) authority can write records and save objects in a save file.
- A user with object operational (*OBJOPR) and object management (*OBJMGT) authority can clear the contents of a save file using the CLRSAVF command. The clear operation is required first when replacing existing records in a save file.
- A user with either save system (*SAVSYS) special authority or object existence (*OBJEXIST) authority for the file can save the description and contents.

Digital signature for a save file

The system verifies any digital signatures present on the save file each time you display the save file or use the save file in a restore operation. If the signature is not valid you cannot display or use the save file in a restore operation. The Verify Object on Restore (QVFYOBJRST) system value does not affect the verification of save files. Therefore, the system verifies the signature every time you display the save file or use the save file in a restore operation.

Related information

Object signing and signature verification

I/O operations on a save file:

Review these considerations which apply to input and output operations on a save file.

- Records are always read and written sequentially. The records read from a save file contain sequence and parity information that is validated when the records are written into another save file. This information ensures that the records are processed in sequence and have not been changed.

  You cannot write a record that has changed since it was retrieved from another save file. You cannot write a record that is not the next record in sequence. If you attempt either of these, an escape message is sent to report the error.
- A read of records from the save file can be done only if the entire file has been written.
- The force-end-of-data (FEOD) function is valid for both input and output.

  For an input file, FEOD signals end-of-file to the program that does the operation.

  To ensure buffered output records are not lost after an FEOD operation completes, they are written to the file. For an output file, buffered output records are not lost even if the job or system fails.

File-dependent attributes for a save file

- The following file-dependent attributes apply when the save file is open:
  - For input operations, the first record returned for a read operation is the one specified by the parameter POSITION when the file is opened. After the first record is read, all remaining records are returned sequentially to the end of the file.
  - For output operations, new records can be added to the end of records already in the file (specified using the EXTEND parameter). Each save file record contains sequencing information used by the system to ensure that a record is not skipped or written more than once.
  - If no record length is specified in the high-level language program that opens the file, a length of 528 bytes is assumed. If the program specifies a record length value, it must be 528 bytes.
- No file-dependent parameters (such as format name) can be specified for read or write operations with a save file. Any file-dependent parameters specified are ignored.

Damage to a save file:

A save file is marked partially damaged if an attempt to read a record or restore an object from the file encounters an auxiliary storage error. You can restore objects from a partially damaged save file other than the objects on the damaged part of auxiliary storage.

The objects on the damaged portion of the auxiliary storage within the save file cannot be restored. When a file is marked partially damaged, you cannot add more records to it until it is cleared.

Partial damage of the save file itself can occur that is unrelated to auxiliary storage errors. Sometimes a partial damage message is issued during a SAVSAVFDTA when the system is very busy. This can happen because an internal operation did not complete within a given time interval. It is most often seen when the SAVSAVFDTA job is running at a low priority and there is a heavy interactive load on the system. Although a SAVSAVFDTA can no longer be done from that save file, the objects in the SAVF can be restored to the system using RSTOBJ.

Sending network files:

The only objects you can send with the Send Network (SNDNETF) command are database file members or save files. The SNDNETF command creates a save file and copies the information into it.

The network file is not included in save operations on the destination system until the network file is received. Once the file is received using the Receive Network File (RCVNETF) command, the copy on the source system is not saved. Consider backing up the information about the destination system.

Other objects (such as programs or commands) must be saved in a save file before they can be sent using the SNDNETF command.

Note: Do not use save files to save objects on a system at the current release to distribute them to a system at a previous release unless TGTRLS(*PRV) is specified on the save command. You might also specify TGTRLS(VxRxMx) on the save command, where (VxRxMx) is the previous-release-value. The current release to previous release rules still apply.

Optical media

Optical media library devices allow you to archive information to optical media, and they provide backup and recovery capability similar to tape media.

If you want to substitute optical media for tape in some of your existing procedures, you need to evaluate how to assign saved objects to directories on the optical media and how to name the media. You need to consider how to use optical media in your save strategy.

Table 4. Consider using optical media as part of your save strategy

Characteristic	Comparison
Access to data	Optical devices are random access devices. File access is independent to the order in which the data is stored. Multiple users can access the same volume simultaneously.
Data transfer rates	Data transfer rates for tape tend to be higher than for optical, particularly if you use tape drive compression.
Durability	Optical media has a life span around 50 years.
Archiving	Write Once Read Many (WORM) optical media is made for archiving. Each sector on the media can only be written to once, when creating and updating files and directories. When a file is changed or deleted, a new version of the file gets written, and the old version still exists on the media. This unique characteristic of never rewriting the same sector is that it allows all previous versions of every file to remain on the media.
Transportability	DVD-RAM media created or written to on the system can be read from any platform that supports the Universal Disk Format (UDF) file system. UDF is an industry standard file system.

How random storage mode affects save functions

Optical devices use a random storage mode to save information. Optical devices use a hierarchical file structure when the system accesses files on the media.

You might specify a path name for the optical file in the save operation beginning with the root (/) directory. If you specify an asterisk (*), the system generates an optical file name in the root (/) directory. If you specify an `'optical_directory_path_name/*'`, the system generates an optical file name in the specified directory on the optical volume. If the directory does not exist, the system creates the directory.

For example, if you specify `SAVLIB LIB(MYLIB) DEV(OPT01) OPTFILE('MYDIR/*')`, the system creates the following optical file: `MYDIR/MYLIB`.

The system looks for active files on the optical media volume for the same file that you save currently. For example, you previously saved a SAVLIB to optical media. Now you run a new SAV command to the same media; the system ignores the SAVLIB files and does not report any active files for your SAV command.

In general, the save operation looks for an active file that matches the path name specified on the OPTFILE parameter. SAVSYS and options 21 and 22 of the SAVE menu look for any active file.

Table 5. Checking for active files on optical media

Consideration	General information
CLEAR(*NONE) parameter	If you specify CLEAR(*NONE) on the save command, the system checks the optical media volume for active optical files. The system looks for active files with the same name and path as the specified optical file. If the system **finds** an optical file that is identical to the specified optical file, the system displays an inquiry message. You might respond to the message by cancelling the process, writing over the existing file on the volume, or inserting a new cartridge. If the system **does not** find any active files and there is enough space on the optical volume, the system writes the files to the media. If the system does not find enough available space on the optical media volume, the system prompts you to insert a new media volume in the media device.

Table 5. Checking for active files on optical media (continued)

Consideration	General information
CLEAR(*ALL) parameter	The CLEAR(*ALL) parameter automatically clears all of the files on the optical media volume without prompting.
CLEAR(*AFTER) parameter	The CLEAR(*AFTER) parameter clears all the media volumes after the first volume. If the system encounters the specified optical file on the first volume, the system sends an inquiry message that allows you to either end the save operation or replace the file.
CLEAR(*REPLACE) parameter	The CLEAR(*REPLACE) parameter automatically replaces active data of the specified optical file on the media volumes.
Check for active files parameter on the GO SAVE command	During a GO SAVE command, menu option 21 or 22, or a SAVSYS command if the system detects an active file of the specified optical file, it displays message OPT1563 in the QSYSOPR message queue. During other save command operations, the system might display message OPT1260 depending on the value of the CLEAR parameter. If the system does not detect an active file of the specified optical file, the system checks for available space. If there is room to write the file, the system writes the file to the current volume in random mode. If there is not enough room, the system prompts you to insert another optical media volume into your optical device. During a GO SAVE command, menu option 21, you specify Y or N at the **Check for active files** prompt to see if there are active files on your media volume. • **Check for active files: N option** When you select the Check for active files: N option, the option forces the system to automatically overwrite all files on your DVD-RAM optical media. • **Check for active files: Y option** When you select the Check for active files: Y option, the option forces the system to check for active files on your DVD-RAM optical media.
SAVSYS command messages	When you run a SAVSYS command to an optical media volume, the system displays message **OPT1503 - Optical volume contains active files** if there are active files on the optical media volume. You can either initialize the media with the Initialize Optical (INZOPT) command or you can specify CLEAR(*ALL) on the SAVSYS command to run an unattended save operation.

Related tasks

"Performing a complete save using the GO SAVE checklist" on page 34
Use this checklist to perform a complete save operation.

Related information

Comparison of offline storage

Optical storage

Virtual optical media

Use this information to learn about virtual optical media in your save environment.

You can use virtual optical media to save images directly to system disk units for convenience, flexibility, and in some cases improved performance. The following scenarios will give you some examples of ways that you can utilize virtual optical in your save environment. Virtual optical is beneficial for unattended saves because it eliminates media errors that could stop an unattended save operation. If you do not allocate enough space in the image catalog to save the intended information, virtual optical will use the autoload feature to create additional images with the same capacity as the last image you loaded, provided the disk storage is available. You must specify automatic load in the reply list, MSGID(OPT149F), to avoid receiving a message that interrupts the unattended save operation.

Ability to duplicate to physical media

When a save is complete to virtual optical, you can transfer it to physical media at any time and not interfere with system operations. You also have the capability to send the stream files from the virtual optical save to another system via FTP. If you have multiple systems, your strategy could be to save each system to virtual optical and then FTP the stream files to a single system where the save to physical media could take place. You can save the virtual images to tape in optical format, or you can use the Duplicate Optical (DUPOPT) command to save the image to optical media.

Note: In a disaster recovery situation you must have physical media of the Licensed Internal Code and the operating system to begin your recovery. If you are saving to virtual optical as part of your disaster protection strategy, you must then save your Licensed Internal Code and operating system to physical media from the virtual images. You must also have access to all of your user data, either on a remote system or on physical media.

Save cumulative PTF record

If you receive fixes on CD-ROM, you can install your fixes from an image catalog. To maintain a complete record of all of the fixes that you apply, you can save these virtual PTF images to media. Then, in a recovery situation, you can restore all of the cumulative PTF images and automatically install them from the image catalog.

Saving data to virtual optical media

| Perform the following steps to save data to virtual optical media. The device and catalog names used
| here are examples.

1. Ensure that the system has enough disk space to hold all the virtual images you are going to create for your save operation.
2. Create a virtual optical device.

```
| CRTDEVOPT  DEVD(OPTVRT01) RSRCNAME(*VRT) ONLINE(*YES)
|            TEXT(text-description)
```

3. Vary on the virtual optical device.

```
| VRYCFG  CFGOBJ(OPTVRT01) CFGTYPE(*DEV) STATUS(*ON)
```

4. Create an image catalog for your save operation.

```
| CRTIMGCLG  IMGCLG(MYCATALOG) DIR('/MYCATALOGDIRECTORY') CRTDIR(*YES)
|            TEXT(image-description)
```

| 5. Add a new image catalog entry with a size of 48 MB to 16 GB. If you are performing a SAVSYS
| operation, the first volume must be at least 2048 MB to accommodate the Licensed Internal Code. If
| you plan to save the operating system, add a new image catalog entry with a size of 4 GB. If you plan
| to duplicate image catalogs to physical media, then ensure you select a virtual image size that
| matches the size of the media you plan to write to. Issue either the first or second set of commands:

```
| ADDIMGCLGE  IMGCLG(MYCATALOG) FROMFILE(*NEW) TOFILE(file-name)
|             IMGSIZ(*DVD4700) TEXT(text-description)
```

| or

```
| ADDIMGCLGE  IMGCLG(MYCATALOG) FROMFILE(*NEW) TOFILE(file-name)
|             IMGSIZ(*CD650) TEXT(catalog-description)
```

| Repeat this step for the number of desired images. You should add the images in the same order as
| you plan to restore from them. The virtual images provide spanning capability, with sequence
| numbers continuing from one volume to the next.

6. Load the image catalog. This step associates the virtual optical device to the image catalog. Only one image catalog at a time can be associated with a specific virtual optical device.

```
| LODIMGCLG  IMGCLG(MYCATALOG) DEV(OPTVRT01) OPTION(*LOAD)
```

7. Initialize the new volume.

```
INZOPT  NEWVOL(MYVOLUMEID) DEV(OPTVRT01) TEXT('volume text')
```

   Repeat this step for the number of new images you want to initialize. Use the WRKIMGCLGE (Work with image catalog entries) command to select the image to be initialized or use the LODIMGCLGE (Load or unload image catalog entry) command to continue to the next volume to be initialized.

```
LODIMGCLGE  IMGCLG(MYCATALOG) IMGCLGIDX(2) OPTION(*MOUNT)
LODIMGCLGE  IMGCLG(MYCATALOG) IMGCLGIDX(1) OPTION(*MOUNT)
```

   When you have completed initializing the new volumes, leave the first entry in mounted status.

8. Run the save command for your desired save operation, listing the virtual optical device in the DEV parameter.

Note: After you create virtual optical images, they will automatically be included when you perform a full system save using GO SAVE Option 21. The virtual optical images could significantly increase the time it takes to complete the Option 21 save operation, even if the image catalog entries do not contain data. If you want to exclude the virtual images from a full system save, use one of the following strategies:

- Use the Change Attribute (CHGATR) command to mark the image catalog directory as non-saveable. For example:

  ```
  CHGATR  OBJ('/MYINFO') ATR(*ALWSAV) VALUE(*NO)
  ```

- Use the Load Image Catalog (LODIMGCLG) command to make the image catalog ready. Image catalogs with a ready status will be omitted from the save operation.
- In an attended save, you can specify to omit the image catalog directories on the Save Object (SAV) command.

Related information

Virtual optical storage

Installing your fixes from an image catalog

CRTDEVOPT

VRYCFG

CRTIMGCLG

ADDIMGCLGE

LODIMGCLG

INZOPT

CHGATR

SAV

Tape media

Tape media might be a good option for your save and restore operations. Tape is the most common media that is used for save and restore operations. It has been around for some time, so it has been widely adopted and continues to be popular.

Tape provides several advantages over other storage methods, for the following reasons:

Table 6. Consider using tape media as part of your save strategy

Characteristic	Comparison
Capacity	As the amount of data you create grows, you can increase your capacity by adding additional tape volumes.
Security	It is easy to keep your data secure by securely storing backups or copies at an offsite location. This also guards against on-site data corruption from viruses, fire, natural disasters, accidental deletions, and other data-loss incidents.

Table 6. Consider using tape media as part of your save strategy (continued)

Characteristic	Comparison
Cost	Because you can store a larger amount of data on tape, it has a lower cost per gigabyte.
Reusability	You can rotate your tapes for backups, which means that you have more than one set of tapes. When one set expires, you can write over the data on it and use the media again.
Encryption	You can encrypt a backup if you use an encrypting tape drive. Encrypting a backup ensures data security by preventing authorized access to the data.

Related information

Tape

Virtual tape media

You can use virtual tape devices to save data directly to system disk units for convenience, flexibility, and in some cases improved performance. These scenarios will give you some examples of ways that you can utilize virtual tape in your save environment.

Virtual tape is beneficial for unattended saves because it eliminates media errors that could stop an unattended save operation. If you do not allocate enough space in the virtual volumes within the image catalog to save the intended information, virtual tape will use the auto-generate feature to create additional virtual tape volumes.

Ability to duplicate to physical media

When a save is complete to a virtual tape volume, you can duplicate the data to physical media at any time and not interfere with system operations. You also have the capability to send the stream files from the virtual tape save to another system via FTP SSL. If you have multiple systems, your strategy could be to save each system to virtual tape. Then, FTP the stream files to a single system where the duplication to the physical media could take place.

Note: In a disaster recovery situation you must have physical media to perform your recovery. If you are saving to virtual tape as part of your disaster recovery strategy, you must duplicate your virtual saves to physical media.

Saving data to virtual tape media

Perform the following steps to save data to virtual tape media. The device and catalog names used here are examples. Perform steps 2 through 5 only if you have not previously created a virtual tape device, image catalog, and images to use for the save operation.

1. Ensure that the system has enough disk space to hold all the virtual images you are going to create for your save operation.
2. Create a virtual tape device. (You can create up to 35 virtual tape devices.)

```
CRTDEVTAP  DEVD(TAPVRT01) RSRCNAME(*VRT) ONLINE(*YES)
           TEXT(text-description)
```

3. Vary on the virtual tape device.

```
VRYCFG  CFGOBJ(TAPVRT01) CFGTYPE(*DEV) STATUS(*ON)
```

4. Create an image catalog and virtual volumes for your save operation.

```
CRTIMGCLG  IMGCLG(MYCATALOG) DIR('/MYCATALOGDIRECTORY') CRTDIR(*YES)
           ADDVRTVOL(3) PREFIX(ABC) IMGSIZ(10000)
           TEXT(catalog-description) TYPE(*TAP)
```

5. Load the image catalog. This step associates the virtual tape device to the image catalog. Only one image catalog at a time can be associated with a specific virtual tape device.

```
LODIMGCLG  IMGCLG(MYCATALOG) DEV(TAPVRT01) OPTION(*LOAD)
```

6. Run a save command listing the virtual tape device in the DEV parameter. Virtual tape devices operate similar to tape media library devices so entering the volume names in the volume parameter automatically mounts the volumes.

Exclude virtual images from a full save

After you create virtual tape images, they will automatically be included when you perform a full system save using GO SAVE Option 21. The virtual tape images could significantly increase the time it takes to complete the Option 21 save operation, even if the image catalog entries do not contain data. If you want to exclude the virtual images from a full system save, use one of the following strategies:

- Use the Change Attribute (CHGATR) command to mark the image catalog directory as non-saveable. For example:

  ```
  CHGATR  OBJ('/Catalog-Path') ATR(*ALWSAV) VALUE(*NO)
  ```

- Use the Load Image Catalog (LODIMGCLG) command to make the image catalog ready. Image catalogs with a ready status will be omitted from the save operation.
- In an attended save, you can specify to omit the image catalog directories on the Save Object (SAV) command.

Related information

Virtual Tape

Rotating tapes and other media

Learn why rotating your media is a good save procedure practice. An important part of a good save procedure is to have more than one set of save media.

When you perform a recovery, you might need to go back to an old set of your media if one of the following is true:

- Your most recent set is damaged.
- You discover a programming error that has affected data on your most recent save media.

At a minimum, rotate three sets of media, as follows:

```
Save 1         Set A
Save 2         Set B
Save 3         Set C
Save 4         Set A
Save 5         Set B
Save 6         Set C
```

And so on.

Many installations find that the best approach is to have a different set of media for each day of the week. This makes it easy for the operator to know which media to mount.

Preparing media and tape drives

Understand why it is important to clean and initialize your tape drives.

You do not need to clean optical media devices as often as tape drives. You must clean your tape units on a regular basis. The read-write heads collect dust and other material that can cause errors when reading or writing to tape. In addition, you should also clean the tape unit if you are going to use it for an extended period of time or if you use new tapes. New tapes tend to collect more material on the read-write heads of the tape unit. For more specific recommendations, refer to the manual for the specific tape unit that you are using.

Initialize your tapes with the Initialize Tape (INZTAP) command or the **Format tape** function available in System i Navigator. Initialize your optical media with the Initialize Optical (INZOPT) command. These commands prepare your media, and the commands can physically erase all data on the media with the CLEAR parameter.

For tapes, you can specify the format (or density in bits per inch) before you write to tape. Do this by using parameters on the INZTAP command when you initialize the tape.

You can specify the format of your optical media. Several optical media types require a particular format. For erasable media, which allows a choice of format, you should use the *UDF format if you use the optical media for backup and recovery purposes.

You can use option 21 (Prepare tapes) on the GO BACKUP menu. This provides a simple method of initializing your media with a naming convention like the ones in Name and label media.

Related reference

"Naming and labeling media"
This information provides guidelines for naming and labeling your media.

Naming and labeling media

This information provides guidelines for naming and labeling your media.

When you initialize each media volume with a name, this helps to ensure that your operators load the correct media for the save operation. Choose media names that help determine what is on the media and in which media set it belongs. The following table shows an example of how you might initialize your media and label them externally if you use a simple save strategy. The INZTAP and the INZOPT commands create a label for each media volume. Each label has a prefix that indicates the day of the week (A for Monday, B for Tuesday, and so on) and the operation.

Note:

1. You can find more information about the different save strategies in the information about Planning a backup and recovery strategy.
2. You might use up to 30 characters to label optical media volumes.

Table 7. Media naming for simple save strategy

Volume Name (INZTAP)	External Label
B23001	Tuesday–GO SAVE command, menu option 23–Media 1
B23002	Tuesday–GO SAVE command, menu option 23–Media 2
B23003	Tuesday–GO SAVE command, menu option 23–Media 3
E21001	Friday–GO SAVE command, menu option 21–Media 1
E21002	Friday–GO SAVE command, menu option 21–Media 2
E21003	Friday–GO SAVE command, menu option 21–Media 3

Your media names and labels for a medium save strategy might look like those in the following table:

Table 8. Media naming for medium save strategy

Volume Name	External Label
E21001	Friday–GO SAVE command, menu option 21–Media 1
E21002	Friday–GO SAVE command, menu option 21–Media 2
AJR001	Monday–Save journal receivers–Media 1
AJR002	Monday–Save journal receivers–Media 2
ASC001	Monday–Save changed objects–Media 1

Table 8. Media naming for medium save strategy (continued)

Volume Name	External Label
ASC002	Monday–Save changed objects–Media 2
BJR001	Tuesday–Save journal receivers–Media 1
BJR002	Tuesday–Save journal receivers–Media 2
B23001	Tuesday–GO SAVE command, menu option 23–Media 1
B23002	Tuesday–GO SAVE command, menu option 23–Media 2

Put an external label on each media. The label should show the name of the media, and the most recent date that you used it for a save operation. Color-coded labels can help you locate and help you store your media: Yellow for Set A, red for Set B, and so on.

Related information

Planning a backup and recovery strategy

Storage solutions

Selecting the encryption media

You can encrypt the data that is stored on tape media to prevent the theft of personal customer information or confidential data if the media is lost or stolen. The first step is to select the encrypting media that you want to use.

You can encrypt backups on tape and virtual tape media only. However, you cannot encrypt backups on optical, virtual optical, or disk media.

You can perform an encrypted backup using software encryption with Backup, Recovery, Media, and Services (BRMS), or using a tape device that supports hardware encryption. Consider these factors in making your decision about your encryption media and method:

- Choose the software encryption method if you want a low-cost solution. This solution is ideal for backing up individual objects that contain customers' personal information or confidential data. Customers with sufficient system resources and a large enough backup window also can encrypt the backup without impacting their business. You can use any tape drive or tape library model with software encryption. However, the performance is not as good as using hardware encryption.
- Choose the hardware encryption method using an encrypting tape drive if you want the best performance for doing save and restore operations, especially a full-system save or restore operation. You do not need host-based encryption of data or the use of specialized encryption appliances to use the encrypting tape drive.

Related information

Tape encryption and decryption

Encrypting tape drive

Several tape library models, such as the IBM System Storage™ TS1120 and IBM Ultrium 4, provide data encryption and key management for backup data. The standalone tape drives do not support encryption. These tape drives must be part of a tape library with encryption capabilities.

You also can perform unencrypted save operations with tape libraries that support encryption.

The encrypting tape drive uses the IBM Encryption Key Manager (EKM) to manage the encryption keys. You can use the encrypting tape drive to save and restore encrypted data, or duplicate encrypted tapes. You can use save/restore commands or Backup, Recovery, and Media Services (BRMS) to back up the data using the encrypting tape drive. You can duplicate encrypted tapes.

For System i environments, the encrypting tape drive must reside in a tape library because the library handles communications with the EKM.

When you are planning your save strategy, consider the following factors:

- What data should or should not be encrypted. (For example, do not encrypt anything on the system or logical partition that is running the EKM, so that you can recover the encryption keys.)
- What encryption keystores are required, and how often should they be changed.
- How to keep the EKM up to date and available when needed for a recovery.

At least two instances of the EKM need to be available in the network so that encryption keys can be provided when needed. The EKM needs to run on a system or logical partition where the backups are not encrypted. That way, you can recover the EKM and its required objects and have the keys for the encrypted saves available.

In a disaster recovery situation, if you are using an encrypting tape drive, you need to access another encrypting tape drive and need to access the keystore and EKM configuration information at the recovery site.

For more information about using the EKM, see *IBM Encryption Key Manager Introduction, Planning, and User's Guide*, GA76-0418, in the IBM Publications Center. Each of the manuals is available from the IBM Publications Center as a printed hardcopy that you can order, in an online format that you can download at no charge, or both.

Related information

Data encryption

IBM Encryption Key Manager component for the Java Platform

Software encryption using BRMS

Backup, Recovery, and Media Services (BRMS) provides you with the ability to encrypt your data to a tape device. This encryption solution is hardware independent, meaning that you do not need to use an encrypting tape drive or other type of encryption device to encrypt the backup data. Only user data can be encrypted with BRMS.

BRMS uses cryptographic services to perform the encrypted backup. When you begin a backup, the BRMS interface asks you for the keys to use for encryption, and what items you want encrypted. You provide the name of the keystore file and the key label. BRMS saves the key information so that it knows what key information is needed to restore data.

The Tape Management exit program calls BRMS before each file is written. If encryption is requested, the Tape Management exit program determines if the data is to be encrypted, and which keystore file and record label to use. The Tape Management exit program does not verify what data is being encrypted.

Note: Currently, you cannot perform software encryption using save/restore commands. However, you can use save/restore commands to back up cryptographic services master keys and keystore files.

Related information

Managing cryptographic keystore files

Tape Management Exit Program

Backup, Recovery, and Media Services for i5/OS PDF

Considerations for encrypting backup data

Encryption of data enhances the data protection capabilities of the System i environment. Consider these important factors when encrypting backup data using either the software or hardware encryption method.

Considerations using the software encryption method

If you are using the software encryption method for a backup:

- You need *ALLOBJ or *SAVSYS special authority or *ALL authority for each file and directory to be saved.
- You might need more tapes for the save operation because encrypted data does not compress or compact as well as nonencrypted data.
- You cannot encrypt data that was saved with a SAVSYS operation (prevented by BRMS).
- You cannot encrypt BRMS-related data, such as QBRM, QUSRBRM, QMSE, and QUSRSYS.
- The encryption keys used for encrypting the data must be available for the life of the tape.
- You cannot encrypt a cryptographic services keystore file that contains the encryption key used for encrypting the tape data. If you restore the keystore file on another system that does not have the file and key already set up, you will not be able to decrypt the tape.
- The encryption keys used for restoring the data must be available on the restore system.
  - If the cryptographic services keystore file is sent to another system, the master key that is associated with the keystore must be the same on the other system.
  - You can export individual encryption keys from a keystore and import these keys into a keystore on another system. This keystore file is then protected with the master key.
- If the master key for a keystore is changed, you must translate the keystores. If this step is not done and the master key is changed a second time, an encrypted save that uses that keystores will fail.
- You can use the SAVSYS command to save the current master keys.
- Encrypting large amounts of data during a save/restore operation affects system performance and availability. Consider doing encryption and decryption during off-peak hours. If you are using a high availability solution, you can switch to the backup system while performing the encrypted backup to avoid affecting users.
- You cannot perform an encrypted save to a previous i5/OS release that does not support encrypted backups.

Considerations using the hardware encryption method

If you are using the hardware encryption method with an encrypting tape drive:

- Performance is fast with the encrypting tape drive, so save and restore operations might have minimal or no effect on users.
- If you use the SAVSYS command to encrypt all the data on tape, you must have the Encryption Key Manager (EKM) running on another system.
- It is recommended that you do *not* encrypt the system or logical partition where the EKM resides. If you use the EKM on the recovery system, you must not encrypt the following data:
  - SAVSYS data.
  - EKM keystore files and EKM configuration file.
  - System libraries.
  - System directories.
  - User libraries: QSYS2, QGPL, QUSRSYS, and QUSRBRM.
- If you are using the encrypting tape drive, you need access to another encrypting tape drive in a disaster recovery situation, along with access to the keystore and EKM configuration information.
- Before you can restore the encrypted data, you must be able to bring the system out of restricted state to start EKM. You also must be able to restore the keystore files and the EKM configuration file.

- If you have a digital certificate associated with the encrypting tape drive, it must be available for the life of the tape.

Verifying your media

Good save procedures ensure that you verify that you use the correct media. Depending on the size of your installation, you might choose to manually verify media, or you might have the system verify the media.

Manual checking
You can use the default of *MOUNTED for the volume (VOL) parameter on the save commands. This tells the system to use the currently mounted media. It is up to the operator to load the correct media in the correct order.

System checking
You specify a list of volume identifiers on the save or restore commands. The system makes sure that the operator loads the correct media volumes in the order specified on the command. If an error occurs, the system sends a message to the operator that requests the correct media volume. The operator can either load another media or override the request.

Expiration dates on the media files are another method that you can use to verify that you use the correct media. If you rely on your operators to verify the media, you might specify an expiration date (EXPDATE) of *PERM (permanent) for your save operations. This prevents someone from writing over a file on the media unintentionally. When you are ready to use the same media again, specify CLEAR(*ALL) or CLEAR(*REPLACE) for the save operation. CLEAR(*REPLACE) automatically replaces active data on the media.

If you want the system to verify your media, specify an expiration date (EXPDATE) that ensures that you do not use the media again too soon. For example, if you rotate five sets of media for daily saves, specify an expiration date of the current day plus 4 on the save operation. Specify CLEAR(*NONE) on save operations so the system does not write over unexpired files.

Avoid situations where the operator must regularly respond to (and ignore) messages such as `Unexpired files on the media`. If operators get in the habit of ignoring routine messages, they might miss important messages.

Storing your media

Store your media where it is safe but accessible. Make sure that they have external labels and that you organize them well so that you can locate them easily. Store a complete set of backup media at a safe, accessible location away from your system.

When choosing your offsite storage, consider how quickly you can retrieve the media. Also consider if you have access to your tapes on the weekends and during holidays. Offsite backup is essential in the case of a site loss.

Handling tape media errors

This information explains the three most common types of media errors and how to handle them.

When reading from or writing to tape, it is normal for some errors to occur. Three types of tape errors can occur during save and restore operations:

Recoverable errors
Some media devices support recovering from media errors. The system repositions the tape automatically and tries the operation again.

Unrecoverable errors–processing can continue
In some cases, the system cannot continue to use the current tape, but can continue processing on a new tape. The system requests you to load another tape. The tape with the unrecoverable error can be used for restore operations.

Unrecoverable errors–processing cannot continue
In some cases, an unrecoverable media error causes the system to stop the save process. How to recover from a media error during a SAVLIB operation describes what to do when this type of error occurs.

Tapes physically wear out after extended use. You can determine if a tape is wearing out by periodically printing the error log. Use the Print Error Log (PRTERRLOG) command and specify TYPE(*VOLSTAT). The printed output provides statistics about each tape volume. If you use unique names (volume identifiers) for your tapes, you can determine which tapes have excessive read or write errors. You should remove these bad tapes from your media library.

If you suspect that you have a bad tape, use the Display Tape (DSPTAP) or the Duplicate Tape (DUPTAP) command to check the integrity of the tape. These commands read the entire tape and detect objects on the tape that the system cannot read.

Related tasks

"Recovering from a media error during a SAVLIB operation" on page 54
This information describes the basic recovery steps for a save operation.

Overview of the GO SAVE command

Use the GO SAVE command to save your entire system or parts of your system that change regularly.

Using the GO SAVE command is a simple way to make sure that you have a good backup of your entire system. The GO SAVE command presents you with Save menus that make it easy to back up your system, no matter what backup strategy you decide to use. It is a good idea to use menu option 21 of the GO SAVE command right after you install your system.

Menu option 21 of the GO SAVE command is the basis for all save strategies. This option allows you to perform a complete save of all the data on your system. Once you have used menu option 21, you can use other menu options to save parts of the system, or to use a manual save process.

Another save method uses Backup, Recovery, and Media Services (BRMS), which automates your save processes. BRMS provides a comprehensive and easy solution for your backup and recovery needs.

Important: Be sure to permanently apply all Licensed Internal Code PTFs (fixes) before using the SAVSYS command, or the GO SAVE menu option 21 or 22.

The following figure illustrates the commands and menu options you can use to save the parts of the system and the entire system.

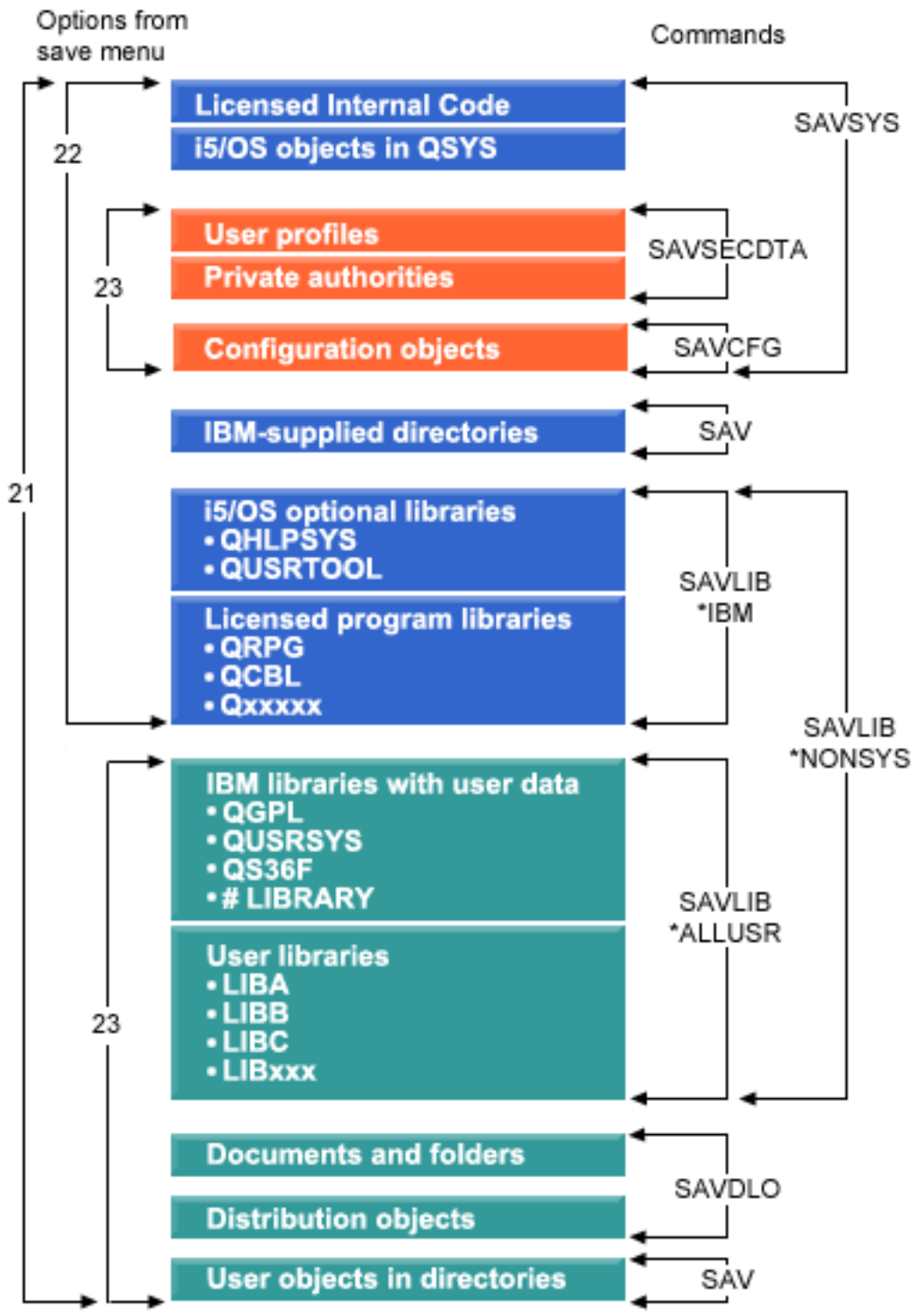


Figure 1. Save commands and menu options

The following information provides an overview and procedures on how to use menu options of the GO SAVE command:

- Overview of the GO SAVE command menu options explains how to start the GO SAVE command and provides more information about the various GO SAVE options.

- Customize your GO SAVE backup instructions allows you to create a list of GO SAVE steps tailored to your save environment.
- View entire GO SAVE checklist provides you with all of the steps for a GO SAVE operations. Some of the steps might not apply to your environment.

Related information

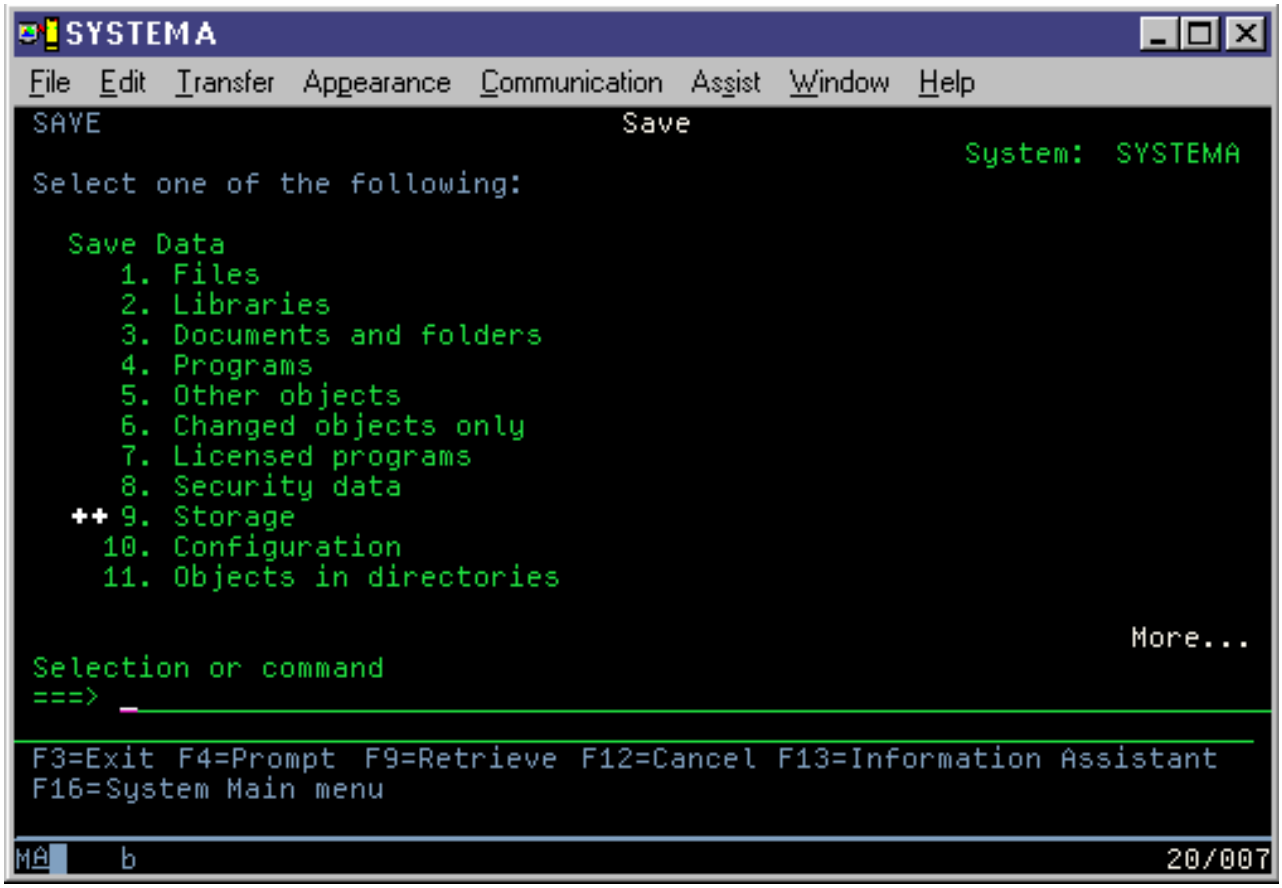
Backup, Recovery, and Media Services

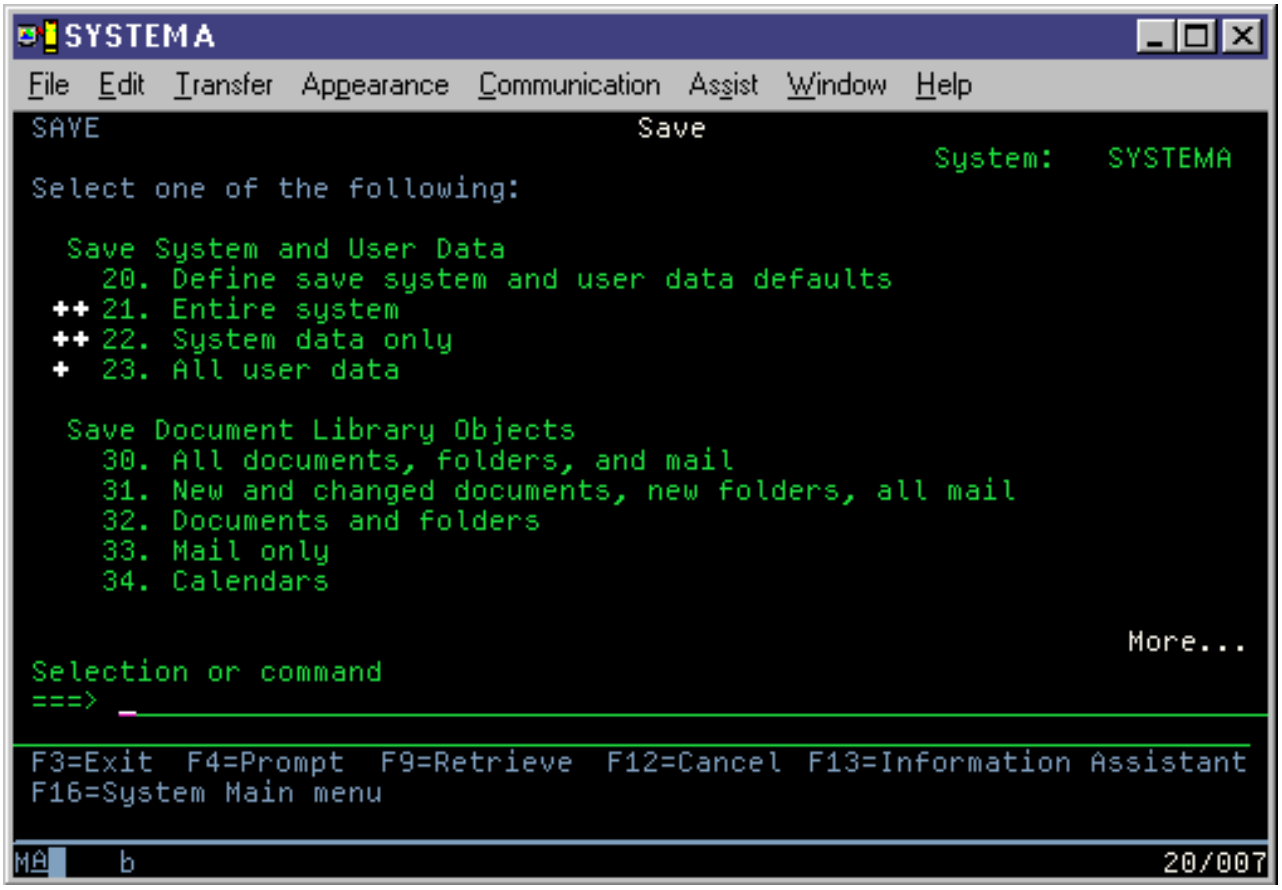
GO SAVE command menu options

This information describes the GO SAVE command and the most common menu options that you can use.

Access the GO SAVE command menu by typing GO SAVE from any command line. From the Save menu, you see option 21, option 22, and option 23 along with many more save options. A single plus sign (+) indicates that the option places your system into a restricted state if you run it without prompting for commands, which means that nothing else can be running on your system when the menu option is selected. If you do prompt for commands, you can skip the ENDSBS command that places the system in restricted state.

A double plus sign (++) indicates that your system must be in a restricted state to run this option. You cannot skip the ENDSBS command that runs as part of the option.





GO SAVE: Option 20 (changing the menu defaults)

You can use the Save menu option 20 to change the default values for the GO SAVE command, menu options 21, 22, and 23. This option simplifies the task of setting your save parameters.

In order to change the defaults, you must have *CHANGE authority for both the QUSRSYS library and the QSRDFLTS data area in the QUSRSYS library.

When you enter the GO SAVE command, then select menu option 20, the system displays the default parameter values for menu options 21, 22, and 23. If this is the first time you have used option 20 from the Save menu, the system displays the IBM-supplied default parameter values. You can change any or all of the parameter values to suit your needs. For example, you can specify additional tape devices or change the message queue delivery default. The system saves the new default values in data area QSRDFLTS in library QUSRSYS. The system creates the QSRDFLTS data area only after you change the IBM-supplied default values.

Once you define new values, you no longer need to worry about which, if any, options to change on subsequent save operations. You can review your new default options and then press Enter to start the save with the new default parameters.

If you have multiple, distributed system with the same save parameters on each system, this option provides an additional benefit. You can define the parameters from the Save menu, using option 20 on one system. Then, save the QSRDFLTS data area, distribute the saved data area to the other system, and restore it.

GO SAVE: Option 21 (saving the entire system)

Option 21 saves everything on your system and allows you to perform the save while you are not there.

Option 21 saves all of your data for additional licensed programs, such as a Domino® server or an integrated server that uses the IBM i5/OS Integrated Server Support option, when you select to vary off your network server. You also can back up data that is stored on a logical partition. If you have Linux or AIX® installed on a guest logical partition, you can back up that partition when you vary off your network server.

Option 21 puts your system into a restricted state. This means that when the save begins, no users can access your system and the backup is the only thing that is running on your system. It is best to run this option overnight for a small system or during the weekend for larger system. If you schedule an unattended save, make sure your system is in a secure location; after you schedule the save, you will not be able to use the workstation where the backup is initiated until the save is complete.

Note: If you are saving information on independent ASPs (also called independent disk pools in System i Navigator), make sure that you have varied on the independent ASPs that you want to save before using option 21.

If you have set the save/restore master key when you do a full system save, the master key also gets saved.

Tip: If you are using the Hardware Management Console (HMC), you can perform system saves using Option 21 while you are at a remote location. For example, you can open a shared console on your HMC from your office, travel to another location and link to the shared session from there.

Option Number	Description	Commands
21	Entire server (QMNSAVE)	ENDSBS SBS(*ALL) OPTION(*IMMED) CHGMSGQ MSGQ(QSYSOPR) DLVRY(*BREAK or *NOTIFY) SAVSYS SAVLIB LIB(*NONSYS) ACCPTH(*YES) SAVDLO DLO(*ALL) FLR(*ANY) SAV DEV('/QSYS.LIB/*media-device-name*.DEVD') + OBJ(('/*') ('/QSYS.LIB' *OMIT) + ('/QDLS' *OMIT))[1] UPDHST(*YES) STRSBS SBSD(*controlling-subsystem*)

[1]The command omits QSYS.LIB file system because the SAVSYS command and the SAVLIB LIB(*NONSYS) command both save it. The command omits the QDLS file system because the SAVDLO command saves it.

"Performing a complete save using the GO SAVE checklist" on page 34 provides you with step-by-step instructions on how to save your entire system with menu option 21 of the GO SAVE command.

Related tasks

"Saving independent ASPs" on page 55
You can save independent auxiliary storage pools (ASPs) in System i Navigator) separately, or you can save them as part of a full system save (GO SAVE Option 21), or when you save all user data (GO SAVE: Option 23). Independent ASPs are also known as *independent disk pools*.

Related reference

"Saving and restoring spooled files" on page 92
For i5/OS V5R4 or later, you can use any of the methods described here to save and restore spooled files. This information contains a table that lists the commands and APIs in order of preference.

"Manually saving parts of your system" on page 45
Use this information to use save commands to save your system manually. This information applies if you use a medium or complex save strategy.

"Methods for saving security data" on page 63
Use any of these methods for saving security data.

"Methods for saving configuration objects in QSYS" on page 64
Use any of these methods for saving configuration objects in QSYS.

"Methods for saving i5/OS optional libraries (QHLPSYS, QUSRTOOL)" on page 65
Use any of these methods for saving i5/OS optional libraries.

Related information

Independent disk pools

SAVLICPGM

GO SAVE: Option 22 (saving system data)

Option 22 saves only your system data. It does not save any user data. Option 22 puts your system into a restricted state. This means that no users can access your system, and the backup is the only thing that is running on your system.

Option Number	Description	Commands
22	System data only (QSRSAVI)	ENDSBS SBS(*ALL) OPTION(*IMMED) CHGMSGQ MSGQ(QSYSOPR) DLVRY(*BREAK or *NOTIFY) SAVSYS SAVLIB LIB(*IBM) ACCPTH(*YES) SAV DEV('/QSYS.LIB/*media-device-name*.DEVD') + OBJ(('/QIBM/ProdData') + ('/QOpenSys/QIBM/ProdData')) + UPDHST(*YES) STRSBS SBSD(*controlling-subsystem*)

"Performing a complete save using the GO SAVE checklist" on page 34 provides you with step-by-step instructions on how to save your system data with menu option 22 of the GO SAVE command.

Related reference

"Methods for saving security data" on page 63
Use any of these methods for saving security data.

"Methods for saving configuration objects in QSYS" on page 64
Use any of these methods for saving configuration objects in QSYS.

"Methods for saving i5/OS optional libraries (QHLPSYS, QUSRTOOL)" on page 65
Use any of these methods for saving i5/OS optional libraries.

Related information

SAVLICPGM

GO SAVE: Option 23 (saving user data)

Option 23 saves all user data. This information includes files, records, and other data that your users supply into your system.

Option 23 puts your system into a restricted state. This means that no users can access your system, and the backup is the only thing that is running on your system.

Note: If you are saving information about independent disk pools, make sure that you have varied on the independent disk pools that you want to save before using option 23. For more information see "Saving independent ASPs" on page 55.

Option Number	Description	Commands
23	All user data (QSRSAVU)	ENDSBS SBS(*ALL) OPTION(*IMMED) CHGMSGQ MSGQ(QSYSOPR) DLVRY(*BREAK or *NOTIFY) SAVSECDTA SAVCFG SAVLIB LIB(*ALLUSR) ACCPTH(*YES) SAVDLO DLO(*ALL) FLR(*ANY) SAV DEV('/QSYS.LIB/*media-device-name*.DEVD') + OBJ(('/*') ('/QSYS.LIB' *OMIT) + ('/QDLS' *OMIT) + ('/QIBM/ProdData' *OMIT) + ('/QOpenSys/QIBM/ProdData' *OMIT))[1] + UPDHST(*YES) STRSBS SBSD(*controlling-subsystem*)

[1]Menu option 23 omits the QSYS.LIB file system because the SAVSYS command, the SAVSECDTA command, the SAVCFG command, and the SAVLIB LIB(*ALLUSR) command save it. The command omits the QDLS file system because the SAVDLO command saves it. Menu option 23 also omits the /QIBM and /QOpenSys/QIBM directories because these directories contain IBM supplied objects.

"Performing a complete save using the GO SAVE checklist" on page 34 provides you with step-by-step instructions on how to save your user data with menu option 23 of the GO SAVE command.

Related tasks

"Saving independent ASPs" on page 55
You can save independent auxiliary storage pools (ASPs) in System i Navigator) separately, or you can save them as part of a full system save (GO SAVE Option 21), or when you save all user data (GO SAVE: Option 23). Independent ASPs are also known as *independent disk pools*.

Related reference

"Saving and restoring spooled files" on page 92
For i5/OS V5R4 or later, you can use any of the methods described here to save and restore spooled files. This information contains a table that lists the commands and APIs in order of preference.

"Methods for saving security data" on page 63
Use any of these methods for saving security data.

"Methods for saving configuration objects in QSYS" on page 64
Use any of these methods for saving configuration objects in QSYS.

"Methods for saving user data" on page 93
You can use these link references to learn how you can save user data in your system.

GO SAVE: Options 40, 41, 42, 43 (saving parts of your system)

You can use the GO SAVE menu options 40, 41, 42, or 43 to save parts of your system. You also can use CL commands to manually save parts of your system.

Option Number	Description	Commands
40	All libraries other than the system library (QMNSAVN)	ENDSBS SBS(*ALL) OPTION(*IMMED) CHGMSGQ MSGQ(QSYSOPR) DLVRY(*BREAK) CHGMSGQ MSGQ(QSYSOPR) DLVRY(*NOTIFY) SAVLIB LIB(*NONSYS) ACCPTH(*YES) STRSBS SBSD(*controlling-subsystem*)
41	All IBM libraries other than the system library	SAVLIB LIB(*IBM)
42	All user libraries	SAVLIB LIB(*ALLUSR)
43	All changed objects in user libraries	SAVCHGOBJ LIB(*ALLUSR)

Related reference

"Manually saving parts of your system" on page 45
Use this information to use save commands to save your system manually. This information applies if you use a medium or complex save strategy.

Performing a complete save using the GO SAVE checklist

Use this checklist to perform a complete save operation.

Use the following checklist for menu options 21, 22, and 23 of the GO SAVE command. When appropriate, select the option that you require. If you choose to, you can print system information during the procedure. The Printing system information topic contains detailed instructions on how to print system information if you do not want the Save menu option command to print your system information automatically.

Some of the steps in this checklist might not apply to your system configuration. See Identify optional features that affect your backup for help to determine whether you use optional features in your environment. If you are still unsure how your system is configured, contact your system administrator.

As an alternative to this checklist, use Customizing your GO SAVE backup to produce a set of instructions that is tailored to your save environment.

Attention: If you are using the Hardware Management Console (HMC), you must back up the HMC in addition to using the GO SAVE: Option 21 to obtain a complete save of your system. See Backing up and restoring the HMC.

1. If you are using software encryption for backup tapes and saving system data (GO SAVE option 21 or 22), set the save/restore master key before you perform the save operation. The save/restore master key is a special purpose master key used to encrypt all the other master keys when saving them in a SAVSYS or GO SAVE operation. For the instructions, see Loading and setting the save/restore master key.
2. Sign on with a user profile that has *SAVSYS and *JOBCTL special authorities, and also has sufficient authority to list different types of system resources. (The QSECOFR user profile contains all of these authorities.) This ensures that you have the authority that you need to place the system in the necessary state and to save everything.
3. If you want to exclude virtual images from a full-system save, use one of the following strategies. Virtual images can significantly increase the time it takes to complete an Option 21 save operation, even if the image catalog entries do not contain data:
   - Use the Change Attribute (CHGATR) command to mark the image catalog directory as nonsavable. For example:
     ```
     CHGATR OBJ('/MYINFO') ATR(*ALWSAV) VALUE(*NO)
     ```
   - Use the Load Image Catalog (LODIMGCLG) command to make the image catalog ready. Image catalogs with a ready status are omitted from the save operation.
   - In an attended save, you can specify to omit the image catalog directories on the Save Object (SAV) command.
4. If you have independent auxiliary storage pools (ASPs), make them available if you want them to be included in an Option 21 or 23 save operation.

   Note: If your system includes independent ASPs that are geographically mirrored, it is recommended that you eliminate them from this GO SAVE option by making them unavailable. You must save independent ASPs that are geographically mirrored separate from this GO SAVE operation.

If the geographically mirrored ASPs remain available during the GO SAVE operation, geographic mirroring is suspended when the system becomes restricted. When you resume mirroring after the save, a complete synchronization is required. Synchronization can be a lengthy process.

5. If you are operating in a clustered environment and want to save independent ASPs without causing a failover, or you want to save the cluster environment for a node, you must end the device cluster resource group and end clustering before you end subsystems.

   Use the End Cluster Resource Group (ENDCRG) command and the End Cluster Node (ENDCLUNOD) command.

6. If you have OptiConnect controllers, vary them off before the save operation. You must vary off OptiConnect controllers before ending subsystems and performing a save of the entire system, or before any save that ends the QSOC subsystem. If you do not vary off OptiConnect controllers before ending subsystems, they go into a failed status, the system marks them as damaged, and the system does not save them. For more information, see Networking for logical partitions.
7. If you have IBM WebSphere® MQ for Multiplatforms, V6 (5724-H72), you need to quiesce WebSphere MQ, V6 before you save the system. For information about WebSphere MQ documentation, see

   WebSphere MQ (www.ibm.com/software/integration/wmq/library/library53.html) .
8. If you plan to run the save procedure immediately, make sure that no jobs are running on the system, type the Work with Active Jobs (`WRKACTJOB`) command.

   If you plan to schedule the save procedure to run later, send a message to all users informing them when the system will be unavailable.
9. To perform an attended save of your system, go to step 11.
10. To perform an unattended save operation, continue with the following steps. An unattended save operation prevents your save operation from stopping because of unanswered messages:
    a. Display the reply list sequence numbers to find what numbers are available for use:

       ```
       WRKRPYLE
       ```

    b. If MSGID(CPA3708) is not already in your reply list, add it. For *xxxx*, substitute an unused sequence number from 1 through 9999:

       ```
       ADDRPYLE SEQNBR(xxxx) +
                MSGID(CPA3708) +
                RPY('G')
       ```

    c. If you are using virtual media for your save media, specify automatic load in the reply list, MSGID(OPT149F), to avoid receiving a message that interrupts the unattended save operation. If necessary, virtual media will use the autoload feature to create additional images with the same capacity as the last image you loaded, provided the disk storage is available.
    d. Change your job to use the reply list and to notify you of any break messages that are sent:

       ```
       CHGJOB INQMSGRPY(*SYSRPYL) BRKMSG(*NOTIFY)
       ```

    Note: You can also set up a default so that whenever you select menu options 21, 22, or 23, the system will always use the reply list. To set up the default, select menu option 20 from the Save menu. Specify `Yes` on the `Use system reply list` option.

| 11. Type `GO SAVE` at a command prompt to display the Save menu.
12. Select the option (21, 22, or 23) from the Save menu and press the Enter key.

    A prompt display describes the function of the menu option that you selected.
13. After reading the **Specify Command Defaults** prompt display, press the Enter key to continue.

```
                Specify Command Defaults

Type choices, press Enter.

  Devices . . . . . .    TAP01          Names
                         _________
                         _________
                         _________

Prompt for commands  . . . . .   Y           Y=Yes, N=No

Check for active file . . . . .  Y           Y=Yes, N=No

Message queue delivery . . . .   *BREAK      *BREAK, *NOTIFY

Start time . . . . . .   *CURRENT    *CURRENT, time

Vary off network servers . . . . *ALL        *NONE, *ALL

Unmount file systems . . . . . . Y           Y=Yes, N=No
```

```
                          Specify Command Defaults

Type choice, press Enter.

Print system information . . . . N           Y=Yes, N=No

Use system reply list  . . . . . N           Y=Yes, N=No

Spooled file data  . . . . . . . *NONE       *NONE, *ALL
```

14. Type your choices for the *Devices* prompt. You can specify as many as four tape media device names. If you specify more than one device, the system automatically switches to the next tape device when the current tape is full. You can select only one DVD-RAM optical media device, tape media library device, or virtual tape device.

    The first device for options 21 and 22 must be your alternate IPL device. If you are creating media to install on another system, the device must be compatible with the alternate IPL device for that system. This ensures that the system can read the SAVSYS media if you need to restore your Licensed Internal Code and the operating system.
15. Type your choice for the *Prompt for commands* prompt. Specify `N` (No) if you want to run an unattended save operation. Specify `Y` (Yes) if you want to change the defaults on the SAV*xx* commands.
16. Type your choice for the *Check for active files* prompt. Specify `Y` (Yes) if you want the system to warn you if active files exist on the save media. The warning you receive gives the following choices:
    - Cancel the save operation.
    - Insert new media and try the command again.
    - Initialize the current media and try the command again.

    Note: If you use DVD-RAM optical media for your save, the system sends inquiry messages to the QSYSOPR message queue when it encounters identical active files. The system sends the inquiry message for each identical active file that it finds.

    Specify `N` (No) if you want the system to write over any active files on the save media without warning you.
17. Type your choice for the *Message queue delivery* prompt. Specify *NOTIFY if you want to do an unattended save operation. This prevents communications messages from stopping the save operation. If you specify *NOTIFY, severity 99 messages that are not associated with the save operation are sent to the QSYSOPR message queue without interrupting the save process. For

example, messages that request that a new volume be loaded interrupt the save operation because they are associated with the job. You cannot continue until you reply to these messages.

Specify *BREAK if you want to be interrupted for severity 99 messages that require a reply.

18. Type your choice for the *Start time* prompt. You can schedule the start of the save operation up to 24 hours later. For example, assume that the current time is 4:30 p.m. on Friday. If you specify 2:30 for the start time, the save operation begins at 2:30 a.m. on Saturday.

    Note:

    a. The system uses the Delay Job (DLYJOB) command to schedule the save operation. Your workstation will be unavailable from the time you request the menu option until the save operation is completed.

    b. **Make sure that your workstation is in a secure location**. Your workstation remains signed on, waiting for the job to start. If the system request function is used to cancel the job, your workstation displays the Save menu. The workstation remains signed on with your user profile and your authority.

    c. Make sure that the value for the QINACTITV system value is *NONE. If the value for QINACTITV is other than *NONE, the workstation will vary off in the amount of time specified. If you changed the value to *NONE, write the old value down.

    d. If you specify a delayed start and want your save operation to run unattended, be sure you have done the following:
       - Set up the system reply list.
       - Specified *NONE on QINACTITV system value.
       - Specified *NOTIFY on message queue delivery.
       - Specify *NOTIFY for any break messages.
       - Responded `N` to the *Prompt for commands* prompt.
       - Responded `N` to *Check for active files*.

19. Type your choice for the *Vary off network servers* prompt. If you use integrated servers, you can optionally vary off the network server descriptions before beginning the save procedure. Examples of network servers include running Windows or Linux operating systems using IBM Extended Integrated Server Support for i5/OS, or running Linux or AIX in a guest partition.

    Select one of the following options to specify which hosted network servers need to be varied off before starting the save operation and varied on after completing the save operation:

    ***NONE**
    Does not vary off network servers. No data is saved for the network servers because the system is in a restricted state. Saving the individual objects requires an active connection to the system.

    ***ALL** Varies off all network servers before starting the save operation. The save operation takes less time but the network server data is not saved in a format that allows restoration of individual objects. You can only restore all of the data from the network servers.

20. Type your choice for the *Unmount file system* prompt. Specify `Y` (Yes) if you want to allow all dynamically mounted file systems to be unmounted. This step enables you to save user-defined file systems (UDFSs) and their associated objects.

    Note: After the save operation is completed, the system does not attempt to remount the file systems.

    Specify `N` (No) if you do not want all dynamically mounted file systems to be unmounted. The file system attributes for the UDFS are saved for all UDFSs that are saved when mounted. To rebuild these mounted file UDFSs during a restore, you must specify the RBDMFS(*UDFS) parameter on the RST command.

21. Type your choice for the *Print system information* prompt. Specify `Y` (Yes) if you want to print the system information. The system information might be useful for disaster recovery. The ″Printing

system information″ topic explains how to print your system information manually without using the automatic GO SAVE command menu option function.

22. Type your choice for the *Use system reply list* prompt. Specify `Y` (Yes) if you want to use the system reply list when the system sends an inquiry message.
23. Type your choice for the *Spooled file data* prompt. Specify whether this backup should save spooled file data for output queues that are saved. The possible choices are:

    ***NONE**
    No spooled file data is saved.

    ***ALL** For each output queue that is saved, all available spooled file data on the output queue is saved.

    Note: Saving spooled files might require more save media and will take additional time.
24. Press the Enter key. If you chose a later start time, your display shows message CPI3716. The message tells when the save operation was requested and when it will start. You cannot use the display until the save operation is completed. The input-inhibited indicator might appear. You have completed the steps for setting up the save operation.

    If you did not choose a later start time, continue with step 25. If the value for QSYSOPR message queue delivery is *BREAK with a severity level of 60 or lower, you must respond to the End Subsystem (ENDSBS) messages. This is true even if you plan to run an unattended save operation specifying a start time of *CURRENT.
25. If you responded `Y` to the system prompt, Prompt for commands, the End Subsystem display appears. Type any changes and press the Enter key. While the system is ending subsystems, you see the following messages. You must respond to them if the QSYSOPR message queue is set to *BREAK with a severity level of 60 or lower. Each message appears at least twice. Press the Enter key to respond to each message.

    a. `CPF0994  ENDSBS SBS(*ALL) command being processed`

    b. `CPF0968  System ended to restricted condition`

    If you responded `N` to the *Prompt for commands* prompt, skip to step 27.
26. When the system is ready to perform each major step in the save operation, you are shown the prompt display for that step. The time between prompt displays might be quite long.

    Note: If independent ASPs are available, you will see additional prompts for options 21 and 23, as described in the Saving independent ASPs topic.

    For option 21 (Entire system) these prompt displays appear:

```
ENDSBS SBS(*ALL) OPTION(*IMMED)
SAVSYS
SAVLIB LIB(*NONSYS) ACCPTH(*YES)
SAVDLO DLO(*ALL) FLR(*ANY)
SAV DEV('/QSYS.LIB/media-device-name.DEVD') +
    OBJ(('/*') ('/QSYS.LIB' *OMIT) +
        ('/QDLS' *OMIT)) +
    UPDHST(*YES)
STRSBS SBSD(controlling-subsystem)
```

    For option 22 (System data only) these prompt displays appear:

```
ENDSBS SBS(*ALL) OPTION(*IMMED)
SAVSYS
SAVLIB LIB(*IBM) ACCPTH(*YES)
SAV DEV('/QSYS.LIB/media-device-name.DEVD') +
    OBJ(('/QIBM/ProdData') +
        ('/QOpenSys/QIBM/ProdData')) +
    UPDHST(*YES)
STRSBS SBSD(controlling-subsystem)
```

    For option 23 (All user data) these prompt displays appear:

```
ENDSBS SBS(*ALL) OPTION(*IMMED)
SAVSECDTA
SAVCFG
SAVLIB LIB(*ALLUSR) ACCPTH(*YES)
SAVDLO DLO(*ALL) FLR(*ANY)
SAV DEV('/QSYS.LIB/media-device-name.DEVD') +
     OBJ(('/*') ('/QSYS.LIB' *OMIT) +
         ('/QDLS' *OMIT) +
         ('/QIBM/ProdData' *OMIT) +
         ('/QOpenSys/QIBM/ProdData' *OMIT)) +
     UPDHST(*YES)
STRSBS SBSD(controlling-subsystem)
```

Type your changes at each prompt display and press the Enter key.

27. When the system sends a message that asks you to load the next volume, load the next media and respond to the message. For example, if the message is the following, load the next volume and then enter `R` to try again (`C` cancels the operation):

    ```
    Device was not ready or next volume was
    not loaded (C R)
    ```

    Attention: If an unrecoverable media error occurs during the SAVLIB procedure, see Recovering from a media error during a SAVLIB operation.

28. Mount all other user-defined file systems at this point if you unmounted them for the save operation.
29. Change the QINACTITV system value back to its original value. You wrote this value down in step 18 c.
30. When the save operation is completed, print the job log. It contains information about the save operation. Use it to verify that the operation saved all objects. Type one of the following:

    ```
    DSPJOBLOG * *PRINT
    ```

    Or

    ```
    SIGNOFF *LIST
    ```

    You have completed the save operation. Make sure that you mark all of your media and store it in a safe, accessible place.

31. If you ended clustering before running the save operation, restart clustering on the save node from a node where clustering is already active.

    For more information, see the online help for Cluster Resource Services, or see i5/OS Clusters technology.

32. Restart the device cluster resource group to enable resiliency.
33. When your independent ASP was saved, the Qdefault.UDFS was unmounted, if you chose to unmount file systems. In order to use the independent ASP again, remount Qdefault.UDFS. Do this step for each independent ASP that you saved.

    ```
    MOUNT TYPE(*UDFS) MFS('/dev/iasp_name/Qdefault.UDFS') MNTOVRDIR('/iasp-name')
    ```

Related concepts

"Saving logical partitions and system applications" on page 102
With logical partitions, you can distribute resources within a single system to make it function as if it were two or more independent systems. You can back up each logical partition separately, or as a set of connected systems.

"Encrypted backups" on page 154
If you use an encrypting tape drive, you can use save commands or Backup, Recovery, and Media Services (BRMS) to perform an encrypted backup. However, if you use the software encryption method, you must use BRMS to perform the encrypted backup.

Related tasks

"Saving independent ASPs" on page 55
You can save independent auxiliary storage pools (ASPs) in System i Navigator) separately, or you can save them as part of a full system save (GO SAVE Option 21), or when you save all user data (GO SAVE: Option 23). Independent ASPs are also known as *independent disk pools*.

Related reference

"Saving and restoring spooled files" on page 92
For i5/OS V5R4 or later, you can use any of the methods described here to save and restore spooled files. This information contains a table that lists the commands and APIs in order of preference.

"Saving data for integrated servers" on page 105
You can back up and recover integrated server data from i5/OS, the integrated Windows server, the integrated Linux server, and VMWare.

"Optical media" on page 15
Optical media library devices allow you to archive information to optical media, and they provide backup and recovery capability similar to tape media.

Related information


Backing up critical HMC data

Storage Solutions

Making a disk pool available

Linux in a guest partition

Clusters

User-defined file systems (UDFSs)

Optional features that affect your backup:

Optional features that affect your backup might include user-defined files, virtual storage, encryption keys, independent disk pools, and network servers.

Are you using user-defined file systems:

A user-defined file system (UDFS) is a file system that a user creates and manages. To determine if you have any UDFS on your system, use one of the following methods.

Using System i Navigator:

Using **System i Navigator**, expand ***your_system*** → **File Systems** → **Integrated File System** → **Root** → **dev** → **QASP*xx*** or select the name of an independent disk pool. If UDFS objects exist, they appear in the right pane.

Using the character-based interface:

Perform the following steps to see if you have user-defined file systems on the system.

1. At a command line, specify `wrklnk '/dev'` .
2. On the Work with Object Links display, select option 5 (Display) to display the contents of the dev directory.
3. Locate object links beginning with QASP*xx* or the name of an independent disk pool, and select Option 5 to display the UDFS within the auxiliary storage pool (ASP).

Are you using software encryption for tapes:

If you are using software encryption for backup tapes and saving system data (GO SAVE option 21 or 22), set the save/restore master key before you perform the save operation. For the instructions, see the Loading and setting the save/restore master key topic.

Do you use virtual storage:

Virtual media simulates tape, CD, or DVD images that are stored directly on your system disk units. To determine if you store virtual images in image catalogs, do the following:

1. At a command line, specify `WRKIMGCLG`.

   Note: The Work with Image Catalogs (WRKIMGCLG) window displays the name of the image catalog, the status, and the virtual type.

Do you use independent disk pools:

An independent disk pool is a collection of disk units that can be brought online or taken offline independent of the rest of the storage on a system. If you have the necessary authority, you can check whether independent disk pools are configured on your system. Using **System i Navigator**, expand ***your_system*** → **Configuration and Service** → **Hardware** → **Disk Units** → **Disk pools** folder. Independent disk pools are numbered 33 – 255.

Have you configured independent disk pools to switch between systems in a cluster:

A System i cluster is a collection or group of one or more systems or logical partitions that work together as a single system. If you have the required authority, you can check to see if your independent disk pool is switchable between systems in a cluster.

1. Using **System i Navigator**, expand ***your_system*** → **Configuration and Service** → **Hardware** → **Disk Units** → **Disk pools** folder.
2. Independent disk pool are numbered somewhere between 33 and 255. Right-click the independent disk pool and select **Properties**.
3. On the **Disk Pool Properties** page the General tab displays the field **Switchable: Yes** if you have configured your independent disk pool to switch between systems.

Do you use WebSphere MQ, V6 on this system:

The IBM WebSphere MQ for Multiplatforms, V6 (WebSphere MQ, V6), 5724-H72, licensed program provides application programming services that enable you to code indirect program-to-program communications that use message queues. This enables programs to communicate with each other independently of their platforms, for example, between the z/OS® and i5/OS operating systems.

To check whether you have installed WebSphere MQ, V6, use one of the following methods:

Using System i Navigator:

Using **System i Navigator**, expand ***your_system*** → **Configuration and Service** → **Software** → **Installed Products**.

Using the character-based interface:

1. At a command line, specify GO LICPGM.
2. Specify option 10 (Display installed licensed programs) to display installed licensed programs.

   If WebSphere MQ, V6 is installed, 5724-H72 appears in the Description column for licensed program 5761-SS1.

   If WebSphere MQ is installed, the Work with Queue Managers (WRKMQM) command lets you see if you have configured any queue managers.

Do you use OptiConnect controllers:

OptiConnect is the system area network that provides high-speed interconnectivity between multiple systems in a local environment.

To check whether you have installed OptiConnect, use one of the following methods:

Using System i Navigator:

Using **System i Navigator**, expand ***your_system*** → **Configuration and Service** → **Installed Products** → **Software**. OptiConnect is option 23 of product 5761-SS1, i5/OS - OptiConnect.

Using the character-based interface:

1. At a command line, specify GO LICPGM.
2. Specify option 10 to display installed licensed programs.
3. If OptiConnect is installed, OptiConnect appears under the Description column for licensed program 5761-SS1.

Do you use network servers:

Several solutions are available that enable you to run other operating systems on your System i product. Examples include integrated server solutions that run an x86-based Linux or Windows operating system, and Linux or AIX running in a logical partition.

Do you use the Hardware Management Console:

If you have a System i5™ model 5*xx*, your system might be equipped with a Hardware Management Console (HMC). An HMC is required if you use Capacity on Demand or logical partitions.

Printing system information:

Printing the system information provides valuable information about your system that will be useful during a system recovery. It is especially useful if you cannot use your SAVSYS media to recover and must use your distribution media.

Printing this information requires *ALLOBJ, *IOSYSCFG, and *JOBCTL authority and produces many spooled file listings. You might not need to print this information every time you perform a backup. However, you should print it whenever important information about your system changes.

1. Print your current disk configuration. This is essential if you plan to do a model upgrade and you are using mirrored protection. This information is also vital if you need to recover an independent ASP. Do the following:
   a. Sign on with a user profile that has *SERVICE special authority.
   b. Type `STRSST` on a command line and press the Enter key.
   c. Specify the service tools user ID and service tools password. These are case-sensitive.
   d. Select option 3 **Work with disk units** on the System Service Tools (SST) display.
   e. Select option 1 **Display disk configuration** on the Work with Disk Units display.
   f. Select option 3 **Display disk configuration protection** on the Display Disk Configuration display.
   g. Print the displays there might be several using the PRINT key for each display.
   h. Press F3 until you see the Exit System Service Tools display.
   i. On the Exit System Service Tools display, press the Enter key.
2. If you are using logical partitions, print the logical partition configuration information.
   a. From the primary partition, type `STRSST` on a command line and press Enter.
   b. If you are using SST, select option 5 **Work with system partitions**, and press Enter. If you are using DST, select option 11 **Work with system partitions**, and press Enter.

c. From the Work With System Partitions menu, select option 1 **Display partition information**.
d. To display all system I/O resources from the Display Partition Information menu, select option 5.
e. At the Level of detail to display field, type *ALL to set the level of detail to ALL.
f. Press F6 to print the system I/O configuration.
g. Select option 1 and press Enter to print to a spooled file.
h. Press F12 to return to the Display Partition Information menu.
i. Select option 2 **Display partition processing configuration**.
j. From the Display Partition Processing Configuration display, Press F6 to print the processing configuration.
k. Press F12 to return to Display Partition Information display.
l. Select option 7 **Display communications options**.
m. Press F6 to print communication configuration.
n. Select option 1 and press Enter to print to a spooled file.
o. Return to a command line and print these three spooled files.

3. If you are operating in a clustered environment, print the cluster configuration information. Use the following commands to print cluster information:
   a. Display Cluster Information — DSPCLUINF DETAIL(*FULL) OUTPUT(*PRINT)
   b. Display Cluster Resource Group — DSPCRGINF CLUSTER(*cluster-name*) CRG(*LIST) OUTPUT(*PRINT)
4. If you have independent ASPs configured, record the relationship between the independent ASP name and number. You can find this information in System i Navigator. In the **Disk Units** folder, select **Disk Pools**.
5. Sign on with a user profile that has *ALLOBJ special authority, such as the security officer. The system lists information only if you have the correct authority. If you sign on as a user with less than *ALLOBJ authority, some of the listings in these steps might not be complete. You must also be enrolled in the system directory before you can print a list of all the folders on the system.
6. If you use the history log or if you have a requirement to keep it, do the following:
   a. Display the system log QHST. This automatically brings it up to date. Type:
      ```
      DSPLOG LOG(QHST) OUTPUT(*PRINT)
      ```
   b. Display all copies of the system log:
      ```
      WRKF FILE(QSYS/QHST*)
      ```
      Look at the list to verify that you saved all copies of the log that you might need later.

      Note: The history (QHST) log contains information such as date created, and the last change date and time. To get more information about the history (QHST) log, select option 8 (Display file description) on the Work with Files display.
   c. To prevent confusion about the date of the log, select the `Delete` option on the Work with Files display. Delete all but the current copies of the system log. This step improves the performance of the SAVSYS command.
7. Print the system information. You can do this by two different methods:
   a. Using the GO SAVE command, on the Specify Command Defaults display, select `Y` at the *Print system information* prompt.
   b. Use the `PRTSYSINF` command.

   The following table describes the spooled files that the system creates. The PRTSYSINF command does not create empty spooled files. If some objects or types of information do not exist on your system, you might not have all of the files listed below.

Table 9. Spooled files created by the system

Spooled File Name	User Data	Description of Contents
QPEZBCKUP	DSPBCKUPL	List of all user libraries
QPEZBCKUP	DSPBCKUPL	List of all folders
QSYSPRT	DSPSYSVAL	Current settings for all system values
QDSPNET	DSPNETA	Current settings for all network attributes
QSYSPRT	DSPCFGL	Configuration lists
QSYSPRT	DSPEDTD	User-defined edit descriptions (a separate spooled file for each)
QSYSPRT	DSPPTF	Details of all fixes that are installed on your system
QPRTRPYL	WRK RPYLE	All reply list entries
QSYSPRT	DSPRCYAP	Settings for access path recovery times
QSYSPRT	DSPSRVA	Settings for service attributes
QSYSPRT	DSPNWSSTG	Network server storage spaces information
QSYSPRT	DSPPWRSCD	Power on/off schedule
QSYSPRT	DSPHDWRSC	Hardware configuration reports (a separate spooled file for each resource type, such as *CMN or *LWS)
QSYSPRT	WRKOPTCFG	Optical device descriptions (if your system has an optical device and optical support is started when you run the command)
QSYSPRT	DSPRJECFG	Remote job entry configurations
QPDSTSRV	DSPDSTSRV	SNADS configuration
QPRTSBSD	DSPSBSD	Subsystem descriptions (a separate spooled file for each subsystem description on your system)
QSYSPRT	DSPSFWRSC	Installed licensed programs (Software Resources List)
QPRTOBJD	DSPOBJD	A list of all the journals on your system
QPDSPJNA	WRKJRNA	The journal attributes for each journal that is not in the QUSRSYS library (a separate file for each journal). Typically, journals in the QUSRSYS library are IBM-supplied journals. If you have your own journals in the QUSRSYS library, you need to manually print information about those journals.
QSYSPRT	CHGCLNUP	Settings for automatic cleanup
QPUSRPRF	DSPUSRPRF	Current values for the QSECOFR user profile
QPRTJOBD	DSPJOBD	Current values for the QDFTJOBD job description
QPJOBLOG	PRTSYSINF	The job log for this job[1]

[1] On your system, this spooled file might be in the QEZJOBLOG output queue.

8. Print a list of directories in the root (/) directory.
   ```
   DSPLNK OBJ('/*') OUTPUT(*PRINT)
   ```
9. Print any IBM-supplied objects that you have modified, such as the QSYSPRT print file.
10. If you maintain a CL program that contains your configuration information, use the Retrieve Configuration Source (RTVCFGSRC) command to ensure that the CL program is current.
    ```
    RTVCFGSRC CFGD(*ALL) CFGTYPE(*ALL) +
              SRCFILE(QGPL/QCLSRC) +
              SRCMBR(SYSCFG)
    ```
11. Print these spooled files. Keep this information with your backup log or your save system media for future reference. If you choose not to print the lists, use the Copy Spooled File (CPYSPLF) command

to copy them to database files. See Save spooled files for information about how to do this. Make sure that the database files are in a library that is saved when you perform the Save menu option.

Manually saving parts of your system

Use this information to use save commands to save your system manually. This information applies if you use a medium or complex save strategy.

Use the information that follows if you are saving your system with a medium or complex save strategy.

You can save the information automatically with the GO SAVE command menu options, or you can save the information manually with individual save commands.

You must save your entire system with menu option 21 of the GO SAVE command before you save parts of your system. You should also periodically save your entire system after you install prerequisite program temporary fixes (PTFs) or before a migration or upgrade.

Related tasks

"GO SAVE: Options 40, 41, 42, 43 (saving parts of your system)" on page 33
You can use the GO SAVE menu options 40, 41, 42, or 43 to save parts of your system. You also can use CL commands to manually save parts of your system.

"GO SAVE: Option 21 (saving the entire system)" on page 31
Option 21 saves everything on your system and allows you to perform the save while you are not there.

Related information

Save strategy

Commands for saving parts of your system

This table groups the data that you need to save on your system. Three sections divide the information.

- System data
- System data and related user data
- User data

For detailed information in each section, select the appropriate link the in table.

Table 10. Saving the parts of your system

Part of your system	GO SAVE command menu option	Save commands
System data is IBM-supplied data that runs your system hardware and software		
Licensed Internal Code	Option 21 or 22	SAVSYS
i5/OS objects in QSYS	Option 21 or 22	SAVSYS
System data and related user data is a combination of system data and related user data		
User profiles	Option 21, 22 or 23	SAVSYS or SAVSECDTA
Private authorities	Option 21, 22 or 23	SAVSYS or SAVSECDTA
Configuration Objects	Option 21, 22, or 23	SAVSYS or SAVCFG
IBM-supplied directories	Option 21 or 22	SAV
i5/OS optional libraries	Option 21 or 22	SAVLIB *NONSYS or SAVLIB *IBM

Table 10. Saving the parts of your system (continued)

Part of your system	GO SAVE command menu option	Save commands
Licensed program libraries	Option 21 or 22	SAVLIB *NONSYS or SAVLIB *IBM
User data is data that you input to the system		
IBM libraries with user data	Option 21 or 23	SAVLIB *NONSYS or SAVLIB *ALLUSR
User libraries	Option 21 or 23	SAVLIB *NONSYS or SAVLIB *ALLUSR
Documents and folders	Option 21 or 23	SAVDLO
User objects in directories	Option 21 or 23	SAV
Distribution objects	Option 21 or 23	SAVDLO

Commands to save specific object types provides you with detailed information about which save command you can use to save specific types of objects.

Related concepts

"Saving system data" on page 49
System data is IBM-supplied data that runs the hardware and software for your system. System data includes the Licensed Internal Code and i5/OS objects in QSYS, libraries, and directories.

Related reference

"Saving system information" on page 59
Use the Save system information (SAVSYSINF) command to perform a partial save of the data saved by the Save system (SAVSYS) command.

"Saving system data and related user data" on page 51
System data and related user data includes information that the system needs to operate and information that allows you to use the system.

"Saving user data in your system" on page 67
User data includes any information that you enter into the system, including the items that are listed in this topic.

"Commands for saving specific object types"
This information contains a table that shows you which commands you can use to save each object type.

Related information

SAVSYS

SAVSECDTA

SAVCFG

SAV

SAVLIB

SAVDLO

Commands for saving specific object types

This information contains a table that shows you which commands you can use to save each object type.

An `X` appears in the column for the SAV command if you can use the SAV*xx* command to individually save an object of that type. When you specify `SAV OBJ('/*')`, the system saves all objects of all types.

Table 11. Objects Saved by Commands According to Object Type

Object Type	System Object Type	SAV*xx* Command: OBJ	LIB	SECDTA	SYS	CFG	DLO	SAV
Alert table	*ALRTBL	X	X		X[1]			X
Authority holder	*AUTHLR			X[6]	X[6]			
Authorization list	*AUTL			X[6]	X[6]			
Bind directory	*BNDDIR	X	X		X[1]			X
Block special file	*BLKSF[10]							X
C locale description	*CLD	X	X		X[1]			X
Character special file	*CHRSF							X
Chart format	*CHTFMT	X	X		X[1]			X
Change request descriptor	*CRQD	X	X		X[1]			X
Class	*CLS	X	X		X[1]			X
Class-of-service description	*COSD				X[3]	X		
Cluster resource group	*CRG	X	X					X
Command definition	*CMD	X	X		X[1]			X
Communications side information	*CSI	X	X		X[1]			X
Configuration list[3,4]	*CFGL				X[3]	X		
Connection list[3]	*CNNL				X[3]	X		
Controller description	*CTLD				X[3]	X		
Cross-system product map	*CSPMAP	X	X		X[1]			X
Cross-system product table	*CSPTBL	X	X		X[1]			X
Data area	*DTAARA	X	X		X[1]			X
Data queue[2]	*DTAQ	X	X		X[1]			X
Data dictionary	*DTADCT		X					X
Device description [11]	*DEVD				X[3]	X		
Directory	*DIR							X
Distributed directory	*DDIR							X
Distributed stream file	*DSTMF							X
Distributions	*MAIL[8]						X	
Document	*DOC						X	X
Double-byte character set dictionary	*IGCDCT	X	X		X[1]			X
Double-byte character set sort table	*IGCSRT	X	X		X[1]			X
Double-byte character set font table	*IGCTBL	X	X		X[1]			X
Edit description[4]	*EDTD	X	X		X			X
Exit registration	*EXITRG	X	X		X			X
File[2,5]	*FILE	X	X		X[1,7]			X
Filter	*FTR	X	X		X[1]			X
First-in-first-out special file	*FIFO							X
Folder	*FLR						X	X
Font mapping table	*FNTTBL	X	X		X[1]			X
Font resource	*FNTRSC	X	X		X[1]			X
Forms control table	*FCT	X	X		X[1]			X
Forms definition	*FORMDF	X	X		X[1]			X
Graphics symbol set	*GSS	X	X		X[1]			X
Internet packet exchange description	*IPXD				X[3]	X[3]		
Job description	*JOBD	X	X		X[1]			X
Job queue[2]	*JOBQ	X	X		X[1]			X
Job scheduler	*JOBSCD	X	X		X[1]			X
Journal[2]	*JRN	X	X		X[1]			X

Table 11. Objects Saved by Commands According to Object Type (continued)

Object Type	System Object Type	SAV*xx* Command: OBJ	LIB	SECDTA	SYS	CFG	DLO	SAV
Journal receiver	*JRNRCV	X	X		X[1]			X
Library [9]	*LIB		X[7]					X
Line description	*LIND				X[3]	X		
Locale	*LOCALE	X	X		X[1]			X
Management collection	*MGTCOL	X	X		X[1]			X
Media definition	*MEDDFN	X	X		X[1]			X
Menu	*MENU	X	X		X[1]			X
Message file	*MSGF	X	X		X[1]			X
Message queue[2]	*MSGQ	X	X		X[1]			X
Mode description	*MODD				X[3]	X		
Module	*MODULE	X	X		X[1]			X
NetBIOS description	*NTBD				X[3]	X		
Network interface description	*NWID				X[3]	X		
Network server configuration	*NWSCFG	X	X		X[1]			X
Network server description	*NWSD				X[3]	X		
Node group	*NODGRP	X	X		X[1]			X
Node list	*NODL	X	X		X[1]			X
Output queue[2], [11]	*OUTQ	X	X		X[1]			X
Overlay	*OVL	X	X		X[1]			X
Page definition	*PAGDFN	X	X		X[1]			X
Page segment	*PAGSEG	X	X		X[1]			X
PDF map	*PDFMAP	X	X					
Panel group	*PNLGRP	X	X		X[1]			X
Printer description group	*PDG	X	X		X[1]			X
Product availability	*PRDAVL	X	X		X[1]			X
Program	*PGM	X	X		X[1]			X
PSF configuration object	*PSFCFG	X	X		X[1]			X
Query definition	*QRYDFN	X	X		X[1]			X
Query form	*QMFORM	X	X		X[1]			X
Query manager query	*QMQRY	X	X		X[1]			X
Reference code translation table	*RCT	X	X		X[1]			X
System/36™ machine description	*S36	X	X		X[1]			X
Search index	*SCHIDX	X	X		X[1]			X
Server storage	*SVRSTG	X	X		X[1]			X
Service program	*SRVPGM	X	X		X[1]			X
Session description	*SSND	X	X		X[1]			X
Spelling help dictionary	*SPADCT	X	X		X[1]			X
SQL package	*SQLPKG	X	X		X[1]			X
Stream file	*STMF							X
Subsystem description	*SBSD	X	X		X[1]			X
Symbolic link	*SYMLINK							X
System object model object	*SOMOBJ							X
System resource management data	*SRMDATA[8]				X[3]	X		
Table	*TBL	X	X		X[1]			X
Time zone description	*TIMZON	X			X			
User defined SQL type	*SQLUDT	X	X		X[1]			X
User index	*USRIDX	X	X		X[1]			X
User profile	*USRPRF			X[6]	X[6]			
User queue[2]	*USRQ	X	X		X[1]			X
User space	*USRSPC	X	X		X[1]			X

Table 11. Objects Saved by Commands According to Object Type (continued)

Object Type	System Object Type	SAV*xx* Command: OBJ	LIB	SECDTA	SYS	CFG	DLO	SAV
Validation list	*VLDL	X	X		X[1]			X
Workstation customization	*WSCST	X	X		X[1]			X

Notes:

1. If the object is in library QSYS.
2. Save files have the option of saving only the description SAVFDTA(*NO) or the content SAVFDTA(*YES). Data Queues have the option of saving only the description QDTA(*NONE) or the content QDTA(*DTAQ). Output Queues have the option of saving only the description SPLFDTA(*NONE) or the content SPLFDTA(*ALL).
3. Use the RSTCFG command to restore these objects.
4. Edit descriptions and configuration lists reside only in library QSYS.
5. The SAVSAVFDTA command saves only the contents of save files.
6. Use the RSTUSRPRF command to restore user profiles. Use the RSTAUT command to restore authorities after you restore the objects that you need. The system restores authorization lists and authority holders when you use the RSTUSRPRF USRPRF(*ALL) command and parameter.
7. If there are save files in the library, the system saves the save file data by default.
8. Mail and SRM data consists of internal object types.
9. Special values for SAVLIB command: LIB parameter shows which IBM-supplied libraries that you cannot save with the SAVLIB command.
10. You can only save block special files when they are not mounted. These files are unmounted user-defined file systems.
11. When a printer device description is saved, the associated output queue located in library QUSRSYS is not saved.

Related reference

"Commands for saving parts of your system" on page 45
This table groups the data that you need to save on your system. Three sections divide the information.

"Objects whose contents are not saved" on page 68
For some object types, the system saves only object descriptions, not the contents of the objects.

Saving system data

System data is IBM-supplied data that runs the hardware and software for your system. System data includes the Licensed Internal Code and i5/OS objects in QSYS, libraries, and directories.

The easiest way to save your system data is with menu option 22 of the GO SAVE command. This saves all of your system data as well as security data and configuration data.

To manually save your system data, use the SAVSYS commands. You can use the same device that you use for the SAVSYS command to perform an initial program load (IPL) of your system. You can also use the SAVSYS save media to perform the IPL.

```
SAVSYS
SAVLIB LIB(*IBM) ACCPTH(*YES)
SAV DEV('/QSYS.LIB/media-device-name.DEVD') +
    OBJ(('/QIBM/ProdData') +
    ('/QOpenSys/QIBM/ProdData')) +
    UPDHST(*YES)
```

Important: Be sure to permanently apply all Licensed Internal Code PTFs (fixes) before using the SAVSYS command, or the GO SAVE menu option 21 or 22.

Related reference

"Commands for saving parts of your system" on page 45
This table groups the data that you need to save on your system. Three sections divide the information.

Related information

SAVSYS command in CL reference

Methods for saving Licensed Internal Code

Use any of these methods to save Licensed Internal Code.

Table 12. Licensed Internal Code information

Item description	When changes occur	Contains user data or changes?	IBM-supplied data?
Licensed Internal Code	Your Licensed Internal Code changes when you apply Program Temporary Fixes (PTFs) or when you install new releases of the operating system.	No	Yes

Common save method for system information	Requires restricted state?
SAVSYS	Yes
GO SAVE command, menu option 21	Yes
GO SAVE command, menu option 22	Yes

Note: **DO NOT** use a tape that you created through DST with option 5=Save Licensed Internal Code from the IPL or Install the System menu. Only do this if Software Services instructs you to use this type of tape. This process creates a tape that does not contain the Licensed Internal Code PTF Inventory information or the i5/OS Operating System. If you recover your system with this type of tape, you need to re-install the Licensed Internal Code from either SAVSYS tapes or from your distribution media. After you re-install the Licensed Internal Code, you can load PTFs onto your system.

Methods for saving system information

Use any of these methods to save system information.

Table 13. System information

Item description	When changes occur	Contains user data or changes?	IBM-supplied data?
System information	System information, such as system values and access path recovery times change regularly.	Yes	Yes

Common save method for system information	Requires restricted state?
SAVSYS	Yes
SAVSYSINF	No
GO SAVE command, menu option 21	Yes
GO SAVE command, menu option 22	Yes

Methods for saving operating system objects

Use any of these methods to save operating system objects.

Table 14. Operating system objects information

Item description	When changes occur	Contains user data or changes?	IBM-supplied data?
Operating system objects	Operating system objects change under two circumstances. First, when you apply Program Temporary Fixes (PTFs). Second, when you install a new release of the operating system.	No[1]	Yes

Note: [1] You should not change objects or store user data in these IBM-supplied libraries or folders. When you install a new release of the operating system, the installation might destroy these changes. If you make changes to objects in these libraries, note them carefully in a log for future reference.

Common save method for system information	Requires restricted state?
SAVSYS	Yes
SAVSYSINF	No
GO SAVE command, menu option 21	Yes
GO SAVE command, menu option 22	Yes

Saving system data and related user data

System data and related user data includes information that the system needs to operate and information that allows you to use the system.

This information includes:

- User profiles
- Private authorities
- Configuration objects
- IBM-supplied directories
- i5/OS optional libraries (QHLPSYS and QUSRTOOL)
- Licensed program libraries (QRPG, QCBL, and Qxxxx)

Related reference

"Commands for saving parts of your system" on page 45
This table groups the data that you need to save on your system. Three sections divide the information.

Saving libraries with the SAVLIB command

Save one or more libraries. You can use this information to save your i5/OS optional libraries. This information also includes special SAVLIB parameters and how to select libraries on your system.

Use the Save Library (SAVLIB) command or menu option 21 of the GO SAVE command to save one or more libraries. When you specify libraries by name on the SAVLIB command, the system saves the libraries in the order in which you list them. You might specify generic values for the LIB parameter.

Related reference

"Methods for saving i5/OS optional libraries (QHLPSYS, QUSRTOOL)" on page 65
Use any of these methods for saving i5/OS optional libraries.

Special values for the SAVLIB command:

The Save Library (SAVLIB) command enables you to use the special values *NONSYS, *ALLUSR, and *IBM to specify groups of libraries.

The system saves libraries that begin with 'Q' as part of *NONSYS and *IBM. Other libraries are saved as part of *NONSYS and *ALLUSR. This table shows which IBM-supplied libraries the system saves for each special value.

Table 15. Comparison of special values for SAVLIB command: LIB parameter. The system saves all of the libraries that are marked with an X.

Library Name	*NONSYS	*IBM	*ALLUSR
	Both user and IBM-supplied libraries	**All IBM-supplied libraries that do not contain user data**	**All user libraries and IBM supplied libraries that contain user data**
QDOC*xxxx*[1]			
QDSNX	X		X
QGPL[7]	X		X
QGPL38	X		X
QMGTC	X		X
QMGTC2	X		X
QMPGDATA	X		X
QMQMDATA	X		X
QMQMPROC	X		X
QPFRDATA	X		X
QRCL	X		X
QRCL*xxxxx*[6]	X		X
QRCY*xxxxx*[6]			
QRECOVERY[3]			
QRPLOBJ[3]			
QRPL*xxxxx*[6]			
QSPL[3]			
QSPL*xxxx*[1,3]			
QSRV[3]			
QSRVAGT	X		X
QSYS[2]			
QSYS*xxxxx*[6]			
QSYS2[7]	X		X
QSYS2*xxxxx*[6], [7]	X		X
QS36F	X		X
QTEMP[3]			
QUSER38	X		X
QUSRADSM	X		X
QUSRBRM	X		X
QUSRDIRCF	X		X
QUSRDIRCL	X		X
QUSRDIRDB	X		X
QUSRIJS	X		X
QUSRINFSKR	X		X
QUSRNOTES	X		X
QUSROND	X		X

Table 15. Comparison of special values for SAVLIB command: LIB parameter (continued). The system saves all of the libraries that are marked with an X.

Library Name	*NONSYS	*IBM	*ALLUSR
	Both user and IBM-supplied libraries	**All IBM-supplied libraries that do not contain user data**	**All user libraries and IBM supplied libraries that contain user data**
QUSRPYMSVR	X		X
QUSRPOSGS	X		X
QUSRPOSSA	X		X
QUSRRDARS	X		X
QUSRSYS[7]	X		X
QUSRVI	X		X
QUSRV*x*R*x*M*x*[4]	X		X
Q*xxxxxx*[5]	X	X	
#CGULIB	X	X	
#COBLIB	X	X	
#DFULIB	X	X	
#DSULIB	X	X	
#LIBRARY	X		X
#RPGLIB	X	X	
#SDALIB	X	X	
#SEULIB	X	X	

[1] Where *xxxx* is a value from 0002 to 0032, corresponding to an auxiliary storage pool (ASP).

[2] Use the SAVSYS command to save information in the QSYS library.

[3] These libraries contain temporary information. They are not saved or restored.

[4] A different library name, format QUSRV*x*R*x*M*x*, might have been created by the user for each previous release supported by IBM. This library contains user commands to be compiled in a CL program for a previous release. For the QUSRV*x*R*x*M*x* user library, the V*x*R*x*M*x* is the version, release, and modification level of a previous release that IBM continues to support.

[5] Q*xxxxxx* refers to any other library that starts with the letter Q. These libraries are intended to contain IBM-supplied objects. They are not saved when you specify *ALLUSR.

[6] Where *xxxxx* is a value from 00033 to 00255, corresponding to an independent auxiliary storage pool (ASP).

[7] The SAVLIB LIB(*NONSYS), SAVLIB LIB(*ALLUSR), and SAVCHGOBJ LIB(*ALLUSR) functions save libraries QSYS2, QGPL, QUSRSYS, and QSYS2*xxxxx* libraries first on the media if they are located on the ASPs specified by the ASPDEV parameter. The other libraries follow in alphabetical order by ASP device name. Libraries on independent ASPs are saved before libraries on the system and basic user ASPs. The IBM libraries are restored first and contain the prerequisite objects necessary for other libraries that follow in the restore process.

Related information

CL finder

OMITLIB parameter and OMITOBJ parameter for the SAVLIB command:

This information explains two parameters for the SAVLIB command.

OMITLIB parameter for the SAVLIB command:

You can exclude one or more libraries by using the OMITLIB parameter. The system does not save libraries that you exclude. You might specify generic values for the OMITLIB parameter.

Here is an example of omitting a group of libraries from a SAVLIB operation:

```
SAVLIB LIB(*ALLUSR) OMITLIB(TEMP*)
```

An example of using the OMITLIB parameter along with generic library naming appears as: `SAVLIB LIB(T*) OMITLIB(TEMP)`. The system saves all libraries that begin with the letter 'T' except for the library that is named TEMP.

You can also use the OMITLIB parameter with generic naming while performing concurrent save operations to different media devices:

```
SAVLIB LIB(*ALLUSR) DEV(first-media-device) OMITLIB(A* B* $* #* @*...L*)
SAVLIB LIB(*ALLUSR) DEV(second-media-device) OMITLIB(M* N* ...Z*)
```

Tips and restrictions for the SAVLIB command:

This information describes considerations that you should keep in mind when using the SAVLIB command

When you save a large group of libraries, you should place your system in a restricted state. This ensures that the system saves all of the important objects. For example, if subsystem QSNADS or directory shadowing is active, the system does not save files whose names begin with QAO in library QUSRSYS. The QAO* files in library QUSRSYS are **very** important files. If the system does not save the QAO* files, you should end the QSNADS subsystem (End Subsystem (ENDSBS) command or End Directory Shadow System (ENDDIRSHD) command). Then you can save the QAO* files.

| You might also need to end the QSYSWRK, QSERVER, and ENDTCPSVR(*MGTC *DIRSRV) subsystems
| to save the QAO* files.

Be sure that you regularly save the QGPL library and the QUSRSYS library. These IBM-supplied libraries contain information that is important to your system and it changes regularly.

Restrictions for the SAVLIB command::

1. You can only specify one library if you save to a save file.
2. You might not run multiple concurrent SAVLIB commands that use the same library. A SAVLIB and Restore Library (RSTLIB) command might not run concurrently using the same library.

Recovering from a media error during a SAVLIB operation:

This information describes the basic recovery steps for a save operation.

If an unrecoverable media error occurs when you save multiple libraries, restart the procedure with the Start Library (STRLIB) parameter on the SAVLIB command.

The basic recovery steps for a save operation are:

1. Check the job log to determine the library where the previous save operation failed. Find the last library saved, which is indicated by a successful completion message.
2. Load the next media volume and ensure that you initialized the media volume. If you were using menu option 21, 22, or 23 when the save operation failed, skip to step 4.
3. Type the SAVxxx command you were using with the same parameter values. Add the STRLIB and OMITLIB parameters and specify the last library that was saved successfully. For example, if you were running a SAVLIB *ALLUSR and CUSTLIB was the last library that was successfully saved, you might type:

   ```
   SAVLIB LIB(*ALLUSR) DEV(media-device-name) +
          STRLIB(CUSTLIB) OMITLIB(CUSTLIB)
   ```

   This starts the save operation on the library after the last successfully saved library. You have completed restarting the SAVLIB operation.

4. If you were using a menu option, select that menu option again.
5. On the Specify Command Defaults display, type `Y` for the *Prompt for commands* prompt. When the system displays prompts for commands that you have completed successfully, press F12 (cancel). When the system displays the prompt for the SAVLIB command, specify the STRLIB and OMITLIB parameters as shown in step 3.

Note: Restoring the system using this set of media requires two RSTLIB commands to restore the libraries.

Related reference

"Handling tape media errors" on page 26
This information explains the three most common types of media errors and how to handle them.

Saving independent ASPs

You can save independent auxiliary storage pools (ASPs) in System i Navigator) separately, or you can save them as part of a full system save (GO SAVE Option 21), or when you save all user data (GO SAVE: Option 23). Independent ASPs are also known as *independent disk pools*.

In either case, you must make the independent ASPs available before you perform the save operation. Refer to the following scenarios and choose the option that best fits your needs.

Related tasks

"GO SAVE: Option 21 (saving the entire system)" on page 31
Option 21 saves everything on your system and allows you to perform the save while you are not there.

"GO SAVE: Option 23 (saving user data)" on page 32
Option 23 saves all user data. This information includes files, records, and other data that your users supply into your system.

"Performing a complete save using the GO SAVE checklist" on page 34
Use this checklist to perform a complete save operation.

"Backing up encrypted auxiliary storage pools" on page 156
Disk encryption enables you to encrypt data stored in user auxiliary storage pools (ASPs) and independent ASPs. You back up an encrypted ASP in the same way as for an unencrypted ASP. However, if the data in the system ASP or independent ASP is lost, you need to perform additional recovery steps.

Related information

Backup, Recovery and Media Services

Making a disk pool unavailable

Scenario: Saving the current ASP group:

Perform the following commands to save the current independent ASP group (the primary ASP and any associated secondary ASPs).

Note: If you are saving independent ASPs that are geographically mirrored, it is recommended that you save the production copy. Quiesce any applications that affect the data in the independent ASP before the save operation. You might also want to consider Backup, Recovery, and Media Services.

1. `SETASPGRP ASPGRP(primary-ASP-name)`
2. `SAVSECDTA ASPDEV(*CURASPGRP)`
3. `SAVLIB LIB(*ALLUSR) ASPDEV(*CURASPGRP)`
4. Unmount any QDEFAULT user-defined file systems in the current independent ASP group.
5. `SAV OBJ(('/dev/*')) UPDHST(*YES) ASPDEV(*CURASPGRP)`
6. Mount any QDEFAULT user-defined file systems (UDFSs) that were unmounted in an earlier step.

Scenario: Saving UDFS ASP:

Perform the following commands to save an available UDFS ASP.

1. `SAVSECDTA ASPDEV(ASP-name)`
2. Unmount any QDEFAULT user-defined file systems in the UDFS ASP that you are saving.
3. `SAV OBJ(('/dev/*')) UPDHST(*YES) ASPDEV(ASP-name)`
4. Mount any QDEFAULT user-defined file systems that were unmounted in an earlier step.

Scenario: Saving independent ASPs as part of a full-system save (Option 21):

If you make independent ASPs available, they will be included in an Option 21 save operation.[1] Follow the checklist in Use GO SAVE: Option 21, 22, and 23, and note extra requirements if you are operating in a clustered environment. Before you end subsystems and restrict your system, make sure that your current job does not use integrated file system objects in the independent ASP. Also, do not perform a SETASPGRP command; Option 21 will perform the necessary commands to save the independent ASPs that you have made available. In addition to the commands listed in Save your whole system with GO SAVE: Option 21, the system performs the following commands for each available ASP group during an Option 21 save:

- `SETASPGRP ASPGRP(asp-group-name)`
- `SAVLIB LIB(*NONSYS) ASPDEV(*CURASPGRP)`
- `SAV OBJ(('/dev/*')) UPDHST(*YES) ASPDEV(*CURASPGRP)`

The system then performs the following command for each available user-defined file system (UDFS) ASP:

`SAV  OBJ(('/dev/*')) UPDHST(*YES) ASPDEV(udfs-asp-name)`

The system also performs a CHKTAP ENDOPT(*UNLOAD) command after the last SAV command it processes.

Saving independent ASPs when you save all user data (Option 23):

If you make independent ASPs available, they will be included in an Option 23 save operation.[1] Follow the checklist in Use GO SAVE: Option 21, 22, and 23, and note extra requirements if you are operating in a clustered environment. Before you end subsystems and restrict your system, make sure that your current job does not use integrated file system objects in the independent ASP. Also, do not perform a SETASPGRP command; Option 23 will perform the necessary commands to save the independent ASPs that you have made available. In addition to the commands listed in Save user data with GO SAVE: Option 23, the system performs the following commands for each available ASP group during an Option 23 save:

- `SETASPGRP ASPGRP(asp-group-name)`
- `SAVLIB LIB(*ALLUSR) ASPDEV(*CURASPGRP)`
- `SAV OBJ(('/dev/*')) UPDHST(*YES) ASPDEV(*CURASPGRP)`

The system then performs the following command for each available user-defined file system (UDFS) ASP:

- `SAV OBJ(('/dev/*')) UPDHST(*YES) ASPDEV(udfs-asp-name)`

The system also performs a CHKTAP ENDOPT(*UNLOAD) command after the last SAV command it processes.

1 If your system includes independent ASPs that are geographically mirrored, it is recommended that you eliminate them from this GO SAVE option by making them unavailable. You should save independent ASPs that are geographically mirrored separately, as described in Save the current ASP group. If the geographically mirrored ASPs remain available during the GO SAVE

operation, geographic mirroring is suspended when the system becomes restricted. When you resume mirroring after the save, a complete synchronization is required. Synchronization can be a very lengthy process.

Example of save order for independent ASPs with GO SAVE: Option 21 or 23:

When you choose to perform a full-system save (Option 21) or to save all user data (Option 23), independent disk pools are saved alphabetically. Secondary ASPs are saved along with their primary.

Save order	Independent ASP name	Independent ASP type	What is saved	Command
1	Apples	Primary	Libraries	SAVLIB LIB (*NONSYS or *ALLUSR)
	Cantaloupe	Secondary		
2	Apples	Primary	User-defined file systems	SAV OBJ(('/dev/*'))
	Cantaloupe	Secondary		
3	Bananas	UDFS	User-defined file systems	SAV OBJ(('/dev/*'))

Saving security data

This information describes the commands that save user profiles, private authorities, authorization lists, and authority holders.

Use the SAVSYS command or the Save Security Data (SAVSECDTA) command to save the following security data:

- User profiles
- Private authorities
- Authorization lists
- Authority holders

The system stores additional security data with each object. The system saves this security data when it saves the object, as follows:

- Public authority
- Owner and owner authority
- Primary group and primary group authority
- Authorization list linked to object

To save security data, the command does not require that your system be in a restricted state. However, you cannot delete user profiles while the system saves security data. If you change user profiles or grant authority while you save security data, your saved information might not reflect the changes.

To reduce the size of a large user profile, do one or more of the following:

- Transfer ownership of some objects to another user profile.
- Remove the private authority to some objects for that user profile.

| Your system stores authority information for objects in the /QNTC file systems for an integrated server.

Note: If you use authorization lists to secure objects in library QSYS, you should write a program to produce a file of those objects. Include this file in the save operation. This is because the association between the object and the authorization list is lost during a restore operation due to QSYS being restored before user profiles.

Saving private authorities

You can save private authorities for objects using either of the following methods:

- Use the SAVSYS or SAVESECDTA command. When you restore the data, specify the Restore User Profiles (RSTUSRPRF) and Restore Authority (RSTAUT) commands to restore the private authorities along with the data. This method is recommended for recovering an entire system.
- Use any of the SAV*xx* or SAVRST*xx* commands with the PVTAUT(*YES) parameter to save private authorities for objects. When you restore the objects, specify PVTAUT(*YES) on the RST*xx* command to restore the private authorities for those objects. Although saving private authorities increases the amount of time it takes to save the objects, it simplifies the recovery of the objects. Using the PVTAUT(*YES) parameter is recommended for restoring specific objects, but it is *not* recommended for recovering the entire system or a large-scale recovery of user data.

Remember: You need save system (*SAVSYS) or all object (*ALLOBJ) special authority to save private authorities. You need *ALLOBJ special authority to restore private authorities.

QSRSAVO API

You can use the Save Objects List (QSRSAVO) API to save user profiles.

Related reference

"Methods for saving security data" on page 63
Use any of these methods for saving security data.

Related information

Save Security Data (SAVSECDTA)

Restore Authority (RSTAUT)

Saving security information

What you should know about restoring user profiles

Saving configuration information

This information describes when to use the SAVCFG command and the SAVSYS command and what object types are saved.

Use the Save Configuration (SAVCFG) command or the SAVSYS (Save System) command to save configuration objects. The SAVCFG command does not require a restricted state. However, if your system is active, the SAVCFG command bypasses the following configuration objects:

- Devices that the system is creating.
- Devices that the system is deleting.
- Any device that is using the associated system resource management object.

When you save your configuration by using the SAVCFG command or the SAVSYS command, the system saves the following object types:

Object types saved

*CFGL	*CTLD	*NWID
*CNNL	*DEVD	*NWSD
*CIO	*LIND	*SRM
*COSD	*MODD	
*CRGM	*NTBD	

Note: You might think of system information, such as system values and network attributes, as configuration information. However, the system does not store this type of information in

configuration objects. The SAVCFG command does not save system information. The SAVSYS command saves it because the system stores it in the QSYS library.

Related reference

"Methods for saving configuration objects in QSYS" on page 64
Use any of these methods for saving configuration objects in QSYS.

Saving system information

Use the Save system information (SAVSYSINF) command to perform a partial save of the data saved by the Save system (SAVSYS) command.

Note:

1. The SAVSYSINF command increases the time and complexity it takes to recover your system.
2. Do not use the SAVSYSINF command as a replacement for the SAVSYS command, and do not use it for a system upgrade or migration. You must have performed a successful SAVSYS before using this command.
3. The SAVSYSINF command is only intended for customers who cannot bring their system to restricted state and take the necessary downtime it takes to perform a SAVSYS command. A SAVSYSINF should be considered a ″partial″ of a complete SAVSYS. During a complete system recovery the SAVSYSINF save will also need to be recovered along with the complete SAVSYS.
4. If you are using the SAVSYSINF command in your backup strategy, the PTF save files must remain on the system until the next SAVSYS command is run. For the Restore System Information (RSTSYSINF) command to recover the system to the current state, SAVSYS requires the PTF save files for all operating system PTFs that have been applied after the last SAVSYS command was run. Do not run the Delete Program Temporary Fix (DLTPTF) command unless you run it just before or after the SAVSYS command. For more information see Clean up fixes.

*SAVSYS or *ALLOBJ special authority is required to use the SAVSYSINF command. You cannot restore a SAVSYSINF to another existing system. You can use the SAVSYSINF for system recovery when you are recovering a system using the SAVSYS and SAVSYSINF media. The data saved by the SAVSYSINF is cumulative from the last SAVSYS.

When you save your system information by using the SAVSYSINF command, the system saves the following object types from QSYS:

Object types that are saved

*JOBD	*JOBQ	*EDTD
*JRN	*MSGF [1]	*SBSD
*CLS	*MSGQ	*TBL
*IGCTBL	*DTAARA	*CMD [1]

[1] objects changed since the last SAVSYS

Additional items that are saved include the following:

Additional items that are saved

System reply list	Service attributes	Environment variables
Most system values	Network attributes	PTFs applied since the last SAVSYS operation [1] for 5761-SS1[2] and 5761-999

[1] If you load PTFs, you must copy them into *SERVICE. This enables SAVSYSINF to find the save files of the PTFs. The Copy PTFs (CPYPTF) service attribute specifies whether to copy PTF save files into *SERVICE when PTFs are loaded from a tape or optical device. Use the CHGSRVA CPYPTF(*YES) command to change the service attribute on your system to copy PTF save files when loading PTFs from media.

Additional items that are saved

[2] The SAVSYSINF command saves PTFs for all licensed programs, including 5761-SS1 and 5761-999, which have been temporarily or permanently applied since the last SAVSYS operation. In addition, for loaded PTFs, the IPL action is checked to determine if the PTF should be included. Loaded PTFs, scheduled to be applied at the next IPL, (IPL action 1 or 3) are saved. PTFs scheduled to be removed at the next IPL, (IPL action 2 or 4) are not saved.

Items that are not saved as part of SAVSYSINF command include the following:

Items that are not saved

Licensed Internal Code	QSYS library	System values that are not saved
Configuration objects (use the SAVCFG command)	Security data (use the SAVSECDTA command)	

The SAVSYSINF command might be incorporated into a save strategy once a base SAVSYS in restricted state is successful. It is recommended that a save of the entire system including a SAVSYS be done in restricted state. This can be accomplished by performing a Go Save Option 21, a combination of an Option 22 and 23, or by using the equivalent functions within BRMS.

Once you have a base SAVSYS, you might perform some or all of these save commands to capture changed or updated information:

```
SAVLIB LIB(*IBM)
SAV OBJ(('/QIBM/ProdData')('/QOpenSys/QIBM/ProdData')) UPDHST(*YES)
SAVSYSINF
```

These are other save commands that should be used on a daily basis to save user data:

```
SAVESECDTA
SAVCFG
SAVLIB LIB(*ALLUSR)
SAVDLO DLO(*ALL) FLR(*ANY)
SAV OBJ(('/*')('/QSYS.LIB'*OMIT)('/QDLS'*OMIT)) UPDHST(*YES)
SAVSYSINF
```

Example SAVSYSINF:

This command saves the system information to the save file named SAVF in library QGPL. The save file will be cleared automatically. Information about what was saved will be written to the first member of the file name OUTPUT in library QGPL. The file and member will be created if they do not exist.

```
SAVSYSINF DEV(*SAVF) SAVF(QGPL/SAVF) CLEAR(*ALL)
          OUTPUT(*OUTFILE) OUTFILE(QGPL/OUTPUT)
```

Related concepts

"Save-while-active function" on page 113
The save-while-active function allows you to use your system during all or part of the save process, that is, save your system while it is active.

Related reference

"Commands for saving parts of your system" on page 45
This table groups the data that you need to save on your system. Three sections divide the information.

Related information

Restoring system information

System values that are not saved:

Most system values are saved when you use the Save System Information (SAVSYSINF) command, or restored with the Restore System Information (RSTSYSINF) command. However, certain system values are not saved as part of the SAVSYSINF command.

Table 16. System values that are not saved as part of SAVSYSINF

System values that are not saved as part of SAVSYSINF	
QABNORMSW	Previous end of system indicator. This system value cannot be changed.
QADLSPLA	System value no longer used by the operating system.
QAUTOSPRPT	System value no longer used by the operating system.
QBOOKPATH	System value no longer used by the operating system.
QCENTURY	Date and time related system values are not saved or restored.
QCONSOLE	Specifies the name of the display device that is the console. You cannot change this system value. The system changes this system value when the console is varied on.
QDATE	Date and time related system values are not saved or restored.
QDATETIME	Date and time related system values are not saved or restored.
QDAY	Date and time related system values are not saved or restored.
QDAYOFWEEK	Date and time related system values are not saved or restored.
QHOUR	Date and time related system values are not saved or restored.
QIGC	Double-byte character set (DBCS) version installed indicator. Specifies whether the DBCS version of the system is installed. You cannot change QIGC; it is set by the system.
QIPLSTS	Initial program load (IPL) status. Indicates which form of IPL has occurred. You can refer to this value in your recovery programs, but you cannot change it.
QJOBMSGQTL	System value no longer used by the operating system.
QJOBMSGQSZ	System value no longer used by the operating system.
QMINUTE	Date and time related system values are not saved or restored.
QMODEL	The number or letters used to identify the model of the system. You cannot change QMODEL.
QMONTH	Date and time related system values are not saved or restored.
QPRCFEAT	This is the processor feature code level of the system. You cannot change QPRCFEAT.
QPWDLVL	To avoid possible security-related problems, QPWDLVL is not saved or restored. See Restoring user profiles for considerations when moving from one password level to another.
QSECOND	Date and time related system values are not saved or restored.

Table 16. System values that are not saved as part of SAVSYSINF (continued)

System values that are not saved as part of SAVSYSINF	
QSRLNBR	This value cannot be changed. It is retrieved from the data fields by the system when installing the i5/OS licensed program.
QSTRPRTWTR	Start print writers at IPL. Specifies whether print writers were started. This value is either set by the system at IPL time or by the user on the IPL Options display. This value can only be displayed or retrieved.
QSVRAUTITV	System value no longer used by the operating system.
QTIME	Date and time related system values are not saved or restored
QUTCOFFSET	Cannot change this system value, it is set during a change to system value QTIMZON.
Password related system values.	All of the password-related system values might not be restored. Refer to Chapter 7 of the Security Reference manual for more information
QYEAR	Date and time related system values are not saved or restored.

Saving licensed programs

Save licensed programs for backup purposes or to distribute licensed programs to other systems in your organization. Use this information to save Licensed program libraries.

You can use the SAVLIB command or the Save Licensed Program (SAVLICPGM) command to save licensed programs. These methods work well for two different purposes:

- If you are saving licensed programs in case you need them for a recovery, use the SAVLIB command. You can save just the libraries that contain licensed programs by specifying SAVLIB LIB(*IBM). Or, you can save the libraries that contain licensed programs when you save other libraries by specifying SAVLIB LIB(*NONSYS).
- If you are saving licensed programs to distribute them to other systems in your organization, use the SAVLICPGM command. You can use a save file as the output for the SAVLICPGM command. You can then send the save file over your communications network.

Related information

Central Site Distribution

SAVLICPGM

Methods for saving system data and related user data

This information provides you with several different methods to save your system data and related user data. These methods include the GO SAVE command and manual save commands and APIs.

The easiest way to save all of your user data and system data is with menu option 21 of the GO SAVE command. This saves all of your system data as well as the related user data.

The following commands enable you to manually save your system and user data:

- SAV (Save Object in the integrated file system or in directories)
- SAVCFG (Save Configuration)
- SAVDLO (Save Document Library Object)
- SAVLIB (Save Library)
- SAVLICPGM (Save Licensed Programs)
- SAVSECDTA (Save Security Data)

| • SAVSYS (Save System)

The following links provide you with detailed information about various save commands and save APIs:
- QSRSave API
- QSRSAVO API
- SAV command
- SAVCFG command
- SAVCHGOBJ command
- SAVDLO command
- SAVLIB command
- SAVOBJ command
- SAVSAVFDTA command
- SAVSECDTA command
- SAVSYS command
- SAVLICPGM command

The following information explains the various methods that you can use to save your system data and related user data:

Methods for saving security data:

Use any of these methods for saving security data.

Table 17. Information about security data

Item description	When changes occur	Contains user data or changes?	IBM-supplied data?
Security data	Security data—user profiles, private authorities, and authorization lists—change regularly as you add new users and objects or if you change authorities.	Yes	Some

Common save method for security data	Requires restricted state?
SAVSYS[1]	Yes
SAVSECDTA[1]	No
GO SAVE command, menu option 21	Yes
GO SAVE command, menu option 22	Yes
GO SAVE command, menu option 23	No[2]
QSRSAVO API (for saving user profiles)	No[3]

Note:

[1] SAVSYS and SAVSECDTA do not save authority information for objects in the QNTC file systems. The system saves authority information with the Windows server objects.

[2] When you use option 23 from the GO SAVE command menu, the default is to place your system in a restricted state. If you choose the prompting option, you can cancel the display that puts your system in a restricted state.

Important: For procedures where the system does not require a restricted state, you must ensure that the system can get the locks necessary to save the information. You should place your system in a restricted state whenever you save multiple libraries, documents, or directories, unless you use the save-while-active function.

[3] You must have *SAVSYS special authority to save user profiles with the QSRSAVO API

Save security data contains information about how to back up the authority data for your users and objects.

Related concepts

"Save-while-active function" on page 113
The save-while-active function allows you to use your system during all or part of the save process, that is, save your system while it is active.

Related tasks

"GO SAVE: Option 21 (saving the entire system)" on page 31
Option 21 saves everything on your system and allows you to perform the save while you are not there.

"GO SAVE: Option 22 (saving system data)" on page 32
Option 22 saves only your system data. It does not save any user data. Option 22 puts your system into a restricted state. This means that no users can access your system, and the backup is the only thing that is running on your system.

"GO SAVE: Option 23 (saving user data)" on page 32
Option 23 saves all user data. This information includes files, records, and other data that your users supply into your system.

Related reference

"Saving security data" on page 57
This information describes the commands that save user profiles, private authorities, authorization lists, and authority holders.

"QSRSAVO API" on page 68
You can use the Save Objects List (QSRSAVO) application programming interface (API) to save multiple objects.

Related information

SAVSYS

SAVSECDTA

Methods for saving configuration objects in QSYS:

Use any of these methods for saving configuration objects in QSYS.

Table 18. Configuration objects in QSYS information

Item description	When changes occur	Contains user data or changes?	IBM-supplied data?
Configuration objects in QSYS	Configuration objects in QSYS change regularly. This happens when you add or change configuration information with commands or with the Hardware Service Manager function. These objects might also change when you update licensed programs.	Yes	No

Common save method for configuration objects in QSYS	Requires restricted state?
SAVSYS	Yes
SAVCFG	No[1]
GO SAVE command, menu option 21	Yes
GO SAVE command, menu option 22	Yes
GO SAVE command, menu option 23	No[2]

[1] **Important:** For procedures where the system does not require a restricted state, you must ensure that the system can get the locks necessary to save the information. You should place your system in a restricted state whenever you save multiple libraries, documents, or directories, unless you use the save-while-active function.

[2] When you use option 23 from the GO SAVE command menu, the default is to place your system in a restricted state. If you choose the prompting option, you can cancel the display that puts your system in a restricted state.

Save configuration information contains information about how to save your configuration objects.

Related concepts

"Save-while-active function" on page 113
The save-while-active function allows you to use your system during all or part of the save process, that is, save your system while it is active.

Related tasks

"GO SAVE: Option 21 (saving the entire system)" on page 31
Option 21 saves everything on your system and allows you to perform the save while you are not there.

"GO SAVE: Option 22 (saving system data)" on page 32
Option 22 saves only your system data. It does not save any user data. Option 22 puts your system into a restricted state. This means that no users can access your system, and the backup is the only thing that is running on your system.

"GO SAVE: Option 23 (saving user data)" on page 32
Option 23 saves all user data. This information includes files, records, and other data that your users supply into your system.

Related reference

"Saving configuration information" on page 58
This information describes when to use the SAVCFG command and the SAVSYS command and what object types are saved.

Related information

SAVSYS

SAVCFG

Methods for saving i5/OS optional libraries (QHLPSYS, QUSRTOOL):

Use any of these methods for saving i5/OS optional libraries.

Table 19. i5/OS optional libraries (QHLPSYS, QUSRTOOL) information

Item description	When changes occur	Contains user data or changes?	IBM-supplied data?
i5/OS optional libraries (QHLPSYS, QUSRTOOL)	i5/OS optional libraries (QHLPSYS, QUSRTOOL) change when you apply Program Temporary Fixes (PTFs) or when you install new releases of the operating system.	No[1]	Yes

Common save method	Requires restricted state?
SAVLIB*NONSYS	Yes
SAVLIB *IBM	No[2, 3]
SAVLIB library-name	No[3]
GO SAVE command, menu option 21	Yes
GO SAVE command, menu option 22	Yes

1 You should avoid changing objects or storing user data in these IBM-supplied libraries or folders. You could lose or destroy these changes when you install a new release of the operating system. If you make changes to objects in these libraries, note them carefully in a log for future reference.

2 You do not need to put your system into a restricted state, but it is recommended.

3 **Important:** For procedures where the system does not require a restricted state, you must ensure that the system can get the locks necessary to save the information. You should place your system in a restricted state whenever you save multiple libraries, documents, or directories, unless you use the save-while-active function.

Save libraries with the SAVLIB command explains how to save one or more libraries. This information also includes special SAVLIB parameters and how to select libraries on your system.

Related concepts

"Save-while-active function" on page 113
The save-while-active function allows you to use your system during all or part of the save process, that is, save your system while it is active.

Related tasks

"GO SAVE: Option 21 (saving the entire system)" on page 31
Option 21 saves everything on your system and allows you to perform the save while you are not there.

"GO SAVE: Option 22 (saving system data)" on page 32
Option 22 saves only your system data. It does not save any user data. Option 22 puts your system into a restricted state. This means that no users can access your system, and the backup is the only thing that is running on your system.

Related reference

"Saving libraries with the SAVLIB command" on page 51
Save one or more libraries. You can use this information to save your i5/OS optional libraries. This information also includes special SAVLIB parameters and how to select libraries on your system.

Related information

SAVLIB

Methods for saving licensed program libraries (QRPG, QCBL, Qxxxx):

Use any of these methods for saving licensed program libraries.

Table 20. Licensed program libraries (QRPG, QCBL, Qxxxx) information

Item description	When changes occur	Contains user data or changes?	IBM-supplied data?
Licensed program libraries (QRPG, QCBL, Qxxxx)	When you update licensed programs	No[1]	Yes

Common save method for licensed program libraries (QRPG, QCBL, Qxxxx)	Requires restricted state?
SAVLIB *NONSYS	Yes
SAVLIB *IBM	No[2, 3]
SAVLICPGM	No[3]
GO SAVE command, menu option 21	Yes
GO SAVE command, menu option 22	Yes

1 You should avoid changing objects or storing user data in these IBM-supplied libraries or folders. You could lose or destroy these changes when you install a new release of the operating system. If you make changes to objects in these libraries, note them carefully in a log for future reference.

2 You do not need to put your system into a restricted state, but it is recommended.

3 **Important:** For procedures where the system does not require a restricted state, you must ensure that the system can get the locks necessary to save the information. You should place your system in a restricted state whenever you save multiple libraries, documents, or directories, unless you use the save-while-active function.

Related information

SAVLIB

Saving user data in your system

User data includes any information that you enter into the system, including the items that are listed in this topic.

- User profiles
- Private authorities
- Configuration objects
- IBM libraries with User Data (QGPL, QUSRSYS, QS36F, #LIBRARY)
- User libraries (LIBA, LIBB, LIBC, LIBxxxx)
- Documents and folders
- Distribution objects
- User objects in directories

Related reference

"Commands for saving parts of your system" on page 45
This table groups the data that you need to save on your system. Three sections divide the information.

Saving objects with the SAVOBJ command

Use the Save Object (SAVOBJ) command to save one or more objects on your system. You can also use the QSRSAVO API to save multiple objects.

Unless you specify that storage is to be freed, this command does not affect objects (other than having the change history updated). You might specify generic values for the LIB parameter with this command. You might run multiple concurrent SAVOBJ operations (including the QSRSAVO API) against a single library.

Related concepts

"Size limitations when saving objects" on page 7
This topic provides information about the size limitations when saving document library objects (DLOs).

Saving multiple objects with the SAVOBJ command:

The parameters of the SAVOBJ command can be used to specify multiple objects in many ways. This information describes some of the most useful parameters.

Parameter	Description
Object (OBJ)	Can be *ALL, a generic name, or a list of as many as 300 specific names and generic names.
Object type (OBJTYPE)	Can be *ALL or a list of types. For example, you can save all job descriptions and subsystem descriptions by specifying `OBJ(*ALL)` and `OBJTYPE(*JOBD *SBSD)`.
Library (LIB)	Can be a single library or a list of as many as 300 library names. You might specify generic values for this parameter.[1]
Omit object (OMITOBJ)	Allows you to specify up to 300 objects to exclude from the SAVOBJ command. You might specify generic values for this parameter. If you use generic values, or supply a specific object type, you can actually omit more than 300 objects.[1]
Omit library (OMITLIB)	Allows you to exclude from 1 to 300 libraries. You might specify generic values for this parameter.[1]

[1] Use the Command user space (CMDUSRSPC) parameter on the save commands to specify up to 32767 names for the parameters.

When you save from more than one library, you can specify one or more object types, but you must specify OBJ(*ALL) for the object name. Libraries are processed in the order that is specified in the library (LIB) parameter.

QSRSAVO API:

You can use the Save Objects List (QSRSAVO) application programming interface (API) to save multiple objects.

The QSRSAVO API is similar to the SAVOBJ command except that you can associate a specific object type with each object name that you specify. This provides more granularity in what you save with a single command. The QSRSAVO API also enables you to save one or more user profiles.

Related reference

"Methods for saving security data" on page 63
Use any of these methods for saving security data.

Related information

API finder

QSRSAVO API

Objects whose contents are not saved:

For some object types, the system saves only object descriptions, not the contents of the objects.

The following table shows these object types:

Table 21. Object Types Whose Contents Are Not Saved

Object Type	Contents Not Saved
Job queues (*JOBQ)	Jobs
Journals (*JRN)	List of currently journaled objects. List of associated journal receivers.
Logical files (*FILE)	Physical files making up logical files are not saved when the logical file is saved. Access paths owned by logical files are saved with the physical file if access path (*YES) is specified on the save command.
Message queues (*MSGQ)	Messages
Output queues (*OUTQ)[1]	Spooled files
Save file (*SAVF)	When SAVFDTA(*NO) is specified.
User Queue (*USRQ)	User queue entries

[1] The default value for parameter does not save spooled files. To save spooled files specify SPLFDTA (*ALL). This will allow you to save all of your spooled files.

Note: Data queue (DTAQ) contents are not saved if QDTA(*NONE) is specified or if it is a DDM data queue.

Related reference

"Commands for saving specific object types" on page 46
This information contains a table that shows you which commands you can use to save each object type.

Saving only changed objects

You can use the save changed object function to reduce the amount of save media that you use. You can also complete your save process in a shorter period of time.

Related reference

"Determining when an object was last saved" on page 9
If a library contains an object, you can use the Display Object Description (DSPOBJD) command to find out when the system saved the object.

Related information

Lotus Domino reference library

Saving Changed Objects (SAVCHGOBJ) command:

Use the Save Changed Objects (SAVCHGOBJ) command to save only those objects that have changed since a specified time.

The options for specifying objects, object types, and libraries are similar to those for the SAVOBJ command:

- You can specify up to 300 different libraries by using the LIB parameter. You can use specific or generic values.
- You can omit up to 300 libraries by using the OMITLIB parameter. You can specify generic values for this parameter.
- You can omit up to 300 objects by using the OMITOBJ parameter. You can specify generic values for this parameter.

Note: Use the Command user space (CMDUSRSPC) parameter on the save commands to specify up to 32767 names for the parameters.

You can perform multiple concurrent SAVCHGOBJ operations against a single library. This can be helpful if you need to save different parts of a library to different media devices simultaneously, as shown in the following example:

```
SAVCHGOBJ OBJ(A* B* C* $* #* @* ...L*) DEV(media-device-name-one) LIB(library-name)
SAVCHGOBJ OBJ(M* N* O* ...Z*) DEV(media-device-name-two) LIB(library-name)
```

Saving changed objects in directories:

This information describes how to use the CHGPERIOD parameter of the SAV command to save objects that have changed.

You can use the change period (CHGPERIOD) parameter on the Save (SAV) command to save objects that changed since a specified time, objects that last changed during a specific time period, or objects that were changed since they were last saved.

If you specify `CHGPERIOD(*LASTSAVE)`, you get any object that changed since **any** save operation you performed for that object with UPDHST(*YES) specified. If you use this method several times during a week, the resulting media will look like Table 23 on page 72.

To perform a save operation that includes all objects that changed since the last complete save of a directory (similar to what is shown in Table 22 on page 71), do one of the following:

- Specify a date and time for the CHGPERIOD parameter.
- Specify UPDHST(*YES) for a complete save operation. Specify UPDHST(*NO) and CHGPERIOD(*LASTSAVE) when you save changed objects.

You can also use the SAV command to save objects that have **not** changed since a particular time by specifying `CHGPERIOD(*ALL *ALL date time)`. This might be useful to archive old information before you remove it.

The system keeps a record of when it last changed the object. It also records whether it changed the object since the last save or not. The system does not store data for when it last saved the object.

Select option 8 on the Work With Object Links (WRKLNK) display to view the attributes that describe whether an object in a directory changed since you last saved it.

Note: If you use the operating system of a client workstation to save an object, the PC archive indicator will be set to 'No'. Since file systems accessed through the network system do not distinguish between save operations, the system archive indicator for those file systems will always match the PC archive indicator. Therefore, changed objects in the file systems accessed through the network system that have been saved by a client workstation save operation will not be saved by a save operation until they have been changed again.

The UPDHST parameter value controls updating of the system save history and PC save history:

- *NO - The system does not update the save history. The PC archive attribute and the system archive attribute do not change.
- *YES - The system updates the save history. For file systems that you access through the network server, the PC archive attribute is set to 'No'. For other file systems, the system archive attribute is set to 'No'.
- *SYS - The system updates the system save history. The system archive attribute is set to 'No'.
- *PC - The system updates the PC save history. The PC archive attribute is set to 'No'.

Related concepts

"How the system updates changed object information with the SAVCHGOBJ command" on page 72
The changed object information kept by the system is a date and a timestamp. When the system creates an object, the system places a timestamp in the changed field. Any change to the object causes the system to update the date and timestamp.

"Using the Save (SAV) command" on page 78
This information explains how to use the SAV command with the OBJ parameter.

Saving changed document library objects:

You can use the Save Document Library Object (SAVDLO) command to save DLOs that have changed since a particular time.

When you specify `SAVDLO DLO(*CHG)`, the default setting saves DLOs that changed since you saved all DLOs for that user ASP (SAVDLO DLO(*ALL) FLR(*ANY)). When you save changed DLOs, the system also saves the distribution objects in the QUSRSYS library, which are called **unfiled mail**.

Note: The system saves documents that a distribution (unfiled mail) refers to if they have changed since the last time that you saved them. If you have Version 3 Release 1 or later, the system does not save these documents when you specify `DLO(*MAIL)`.

Related reference

"Methods for saving IBM-supplied document library objects and folders" on page 99
This information describes common save methods for saving IBM-supplied document library objects.

Additional considerations for SAVCHGOBJ:

If you need to save changed objects as part of your save strategy, you must ensure that any partial save activity that occurs between your full save operations does not affect what you save with the SAVCHGOBJ command.

If users occasionally save individual objects, you might want them to specify UPDHST(*NO). That prevents their save activity from having an impact on the overall SAVCHGOBJ strategy.

Note: The most common way to use the SAVCHGOBJ command is to specify REFDATE(*SAVLIB). If you have a new library that has never been saved, it is not saved when you specify SAVCHGOBJ REFDATE(*SAVLIB).

Using SAVCHGOBJ–Example:

In a typical environment, you might use the SAVLIB command once a week and the SAVCHGOBJ command every day. Because the default for SAVCHGOBJ is from the last SAVLIB operation, the media that the SAVCHGOBJ command produces tends to grow during the week.

What follows shows an example of using SAVCHGOBJ during a typical week. Assume that you save the entire library on Sunday night and the SAVCHGOBJ command is used each evening during the week:

Table 22. SAVCHGOBJ Command: Cumulative

Day	Files That Changed That Day	Media Contents
Monday	FILEA, FILED	FILEA, FILED
Tuesday	FILEC	FILEA, FILEC, FILED
Wednesday	FILEA, FILEF	FILEA, FILEC, FILED, FILEF
Thursday	FILEF	FILEA, FILEC, FILED, FILEF
Friday	FILEB	FILEA, FILEB, FILEC, FILED, FILEF

If a failure occurred on Thursday morning, you would:

1. Restore the library from Sunday evening.
2. Restore all the objects from Wednesday's SAVCHGOBJ media volumes.

When you use this technique of saving everything that changed since the last SAVLIB, recovery is easier. You need to restore only the media volumes from the most recent SAVCHGOBJ operation.

Changing the reference date and time: The default for the command is to save objects that have changed since the library was last saved using the SAVLIB command. You can specify a different reference date and time by using the reference date (REFDATE) and reference time (REFTIME) parameters on the SAVCHGOBJ command. This enables you to save only objects that have changed since the last SAVCHGOBJ operation.

This might reduce the amount of media and the time for the save operation. Here is an example:

Table 23. SAVCHGOBJ Command–Not Cumulative

Day	Files That Changed That Day	Media Contents
Monday	FILEA, FILED	FILEA, FILED
Tuesday	FILEC	FILEC
Wednesday	FILEA, FILEF	FILEA, FILEF
Thursday	FILEF	FILEF
Friday	FILEB	FILEB

You can restore the SAVCHGOBJ media from earliest to latest. Or you can display each media volume and restore only the latest version of each object.

How the system updates changed object information with the SAVCHGOBJ command:

The changed object information kept by the system is a date and a timestamp. When the system creates an object, the system places a timestamp in the changed field. Any change to the object causes the system to update the date and timestamp.

Use the DSPOBJD command and specify DETAIL(*FULL) to display the date and time of the last change for a specific object. Use the Display File Description (DSPFD) command to display the last change date for a database member.

To display the last change date for a document library object, do the following:

1. Use the Display DLO Name (DSPDLONAM) command to display the system name for the DLO and the ASP where it is located.
2. Use the DSPOBJD command, specifying the system name, the name of the document library for the ASP (such as QDOC0002 for ASP 2), and DETAIL(*FULL).

Some common operations that result in a change of the date and time are:

- Create commands
- Change commands
- Restore commands
- Add and remove commands
- Journal commands
- Authority commands
- Moving or duplicating an object

These activities do not cause the system to update the change date and time:

- *Message queue*. When the system sends a message or when the system receives a message.
- *Data queue*. When the system sends an entry or when the system receives and entry.

When you IPL, the system changes all of the job queues and output queues.

Change Information for Database Files and Members: For database files, the SAVCHGOBJ command saves the file description and any members that changed.

Some operations change the change date and time of the file and all of its members. Examples are the CHGOBJOWN, RNMOBJ, and MOVOBJ commands. If you save a file with 5 or more members, the system updates the change date for the library because it creates a recovery object in the library to improve save performance.

Operations that affect only the content or attributes of a member change only the date and time of the members. Examples are:

- Using the Clear Physical File Member (CLRPFM) command
- Updating a member by using source entry utility (SEU)
- Updating a member with a user program.

The SAVCHGOBJ command can be useful for backing up typical source files. Normally, a source file has many members, and only a small percentage of members change every day.

Related concepts

"Saving changed objects in directories" on page 70
This information describes how to use the CHGPERIOD parameter of the SAV command to save objects that have changed.

Saving database files

This information describes what the system does when you save a database file.

Use the SAVOBJ command to save individual database files. You can use the FILEMBR (file member) parameter to save:

- A list of members from one database file.
- The same group of members from multiple files.

Here is what the system does when you save a database file:

Table 24. Saving database files

Type of File	What is saved
Physical file, TYPE(*DATA), keyed access path[1]	Description, data, access path
Physical file, TYPE(*DATA), access path not keyed	Description, data
Physical file, TYPE(*SRC), keyed access path	Description, data
Logical file[2]	Description

[1] The following types of access paths are included as keyed access paths: keyed access paths, primary key constraints, unique constraints, referential constraints.

[2] You can save the access path for a logical file by saving the associated physical files using the SAVLIB, SAVOBJ, or SAVCHGOBJ command and specify the ACCPTH parameter..

The description for a file might include the following:

- Definitions of triggers and the programs that are associated with the file, but not the programs themselves. You must save the programs separately.
- Definitions of any constraints for the file.

Special considerations apply when you restore a file that has trigger programs or referential constraints defined.

Related concepts

"Saving journaled objects and libraries" on page 76
When you save a journaled object or journaled library, the system writes an entry to the journal for each object that you save.

Related information

SAVOBJ command

How the system restores files with referential constraints

How the system restores files with triggers

Saving access paths:

When you restore a database file, but you did not save the access path to the database, the system rebuilds the access path. You can significantly reduce the amount of time it takes you to recover if you save the access paths. However, the process that saves access paths increases the time for the save operation and the amount of media that you use.

To save access paths that are owned by logical files, specify ACCPTH(*YES) on the SAVCHGOBJ, SAVLIB, and SAVOBJ commands when you save the physical files. The system saves access paths when you save the physical file because the physical file contains the data that is associated with the access path. When you save the logical file, you are saving only the description of the logical file.

When a save command (SAVLIB, SAVOBJ, SAVCHGOBJ, SAVRSTLIB, SAVRSTOBJ, or SAVRSTCHG) is performed, the save access paths parameter value is determined by the QSAVACCPTH system value when ACCPTH(*SYSVAL) is specified. When ACCPTH(*YES) or ACCPTH(*NO) is specified, this system value is ignored. If access paths are to be saved, the process that saves access paths increases the time for the save operation and the amount of media that you use. However, by having the access paths saved, you significantly reduce the amount of time it takes to recover a system because the access paths do not need to be rebuilt.

The system saves access paths that logical files own, and that are not used for referential constraints if all of the following are true:

- You specify ACCPTH(*YES) on the save command for the physical files.
- All based-on physical files under the logical file are in the same library and are being saved at the same time on the same save command.
- The logical file is MAINT(*IMMED) or MAINT(*DLY).

In all cases, the systemsaves an access path only if it is valid and not damaged at the time of the save operation.

When you save a physical file that is not a source file, the system saves the following types of access paths with it, even if you do or do not specify ACCPTH(*YES):

- Keyed access paths that are owned by the physical file
- Primary key constraints
- Unique constraints
- Referential constraints

If the based-on physical files and the logical files are in different libraries, the system saves the access paths. However, the system might not restore these access paths.

Related information

How the system restores access paths

EXAMPLE - Saving files in a network:

This information describes a physical file and how logical files have access paths over the physical file.

The following figure shows a physical file, FILEA in the LIB1 library. Logical file FILEB in LIB1 and logical file FILEC in LIB2 have access paths over physical file FILEA in LIB1.

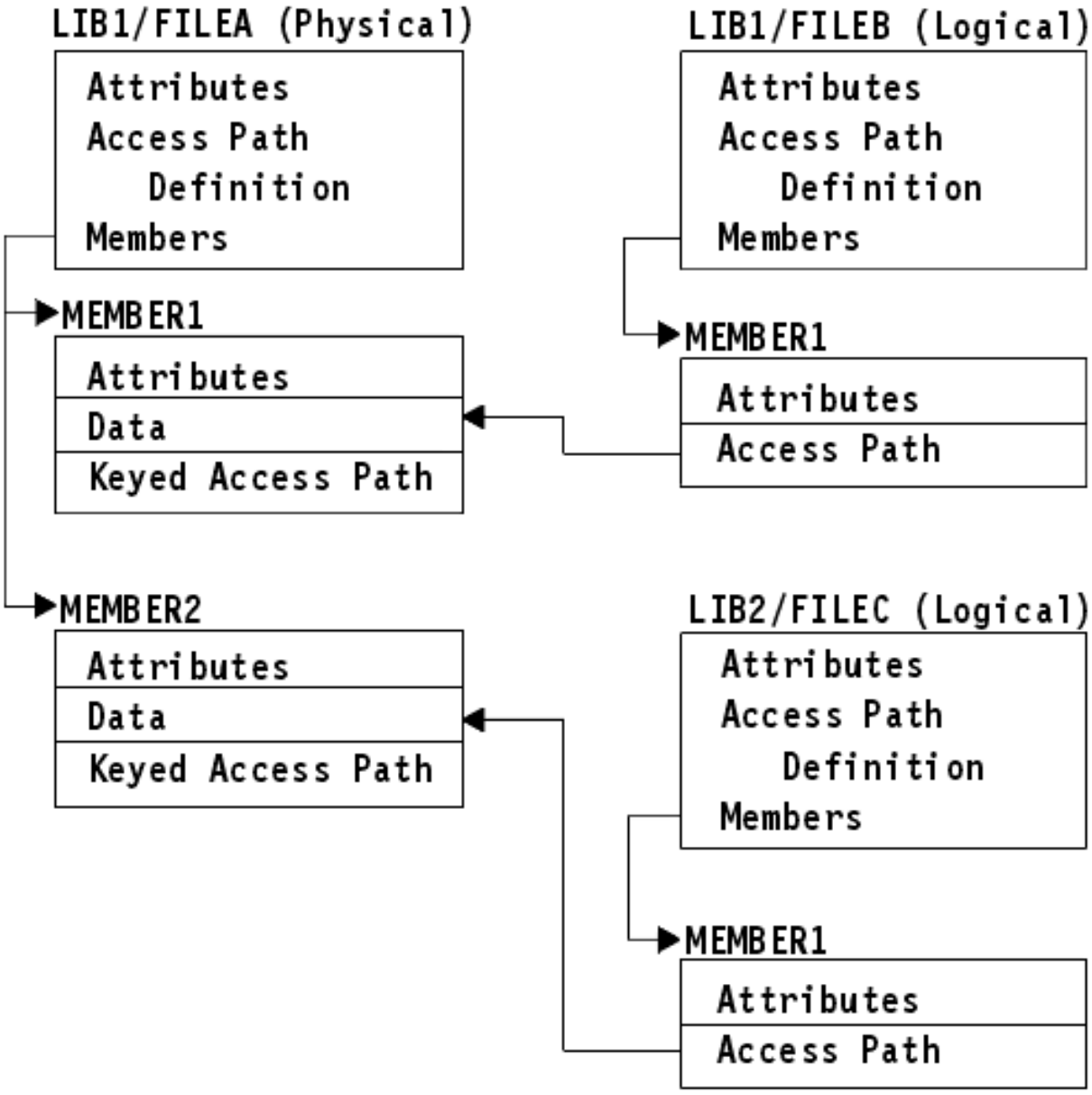


Figure 2. Saving Access Paths

The following table shows which parts of this file network different commands save:

Table 25. Saving a File Network

Command	What is saved
`SAVLIB LIB(LIB1) ACCPTH(*YES)`	FILEA: description, data, keyed access path
	FILEB: description, access path
	FILEC: access path
`SAVOBJ OBJ(FILEA) LIB(LIB1) ACCPTH(*YES)`	FILEA: description, data, keyed access path
	FILEB: access path
	FILEC: access path

Table 25. Saving a File Network (continued)

Command	What is saved
`SAVLIB LIB(LIB2) ACCPTH(*YES)`	FILEC: description

Saving files with referential constraints:

Referential constraints link multiple files together in a network, similar to the network for access paths. You might think of this as a relationship network. If possible, you should save all the files in a relationship network in a single save operation.

If you restore files that are in a relationship network during separate restore operations, the system must verify that the relationships are still valid and current. You can avoid this process and improve restore performance if you save and restore relationship networks in a single operation.

Related information

How the system restores files with referential constraints

Saving journaled objects and libraries

When you save a journaled object or journaled library, the system writes an entry to the journal for each object that you save.

Keep the following considerations in mind when you save journaled objects:

- When you start journaling an object, save that object after you start journaling it.
- After you add a new physical file member to a journaled database file, you should save that database file.
- Save an integrated file system object after it is added to a directory which has the inherit journaling attribute on.
- When you are journaling libraries, objects created, moved, or restored into a journaled library are also journaled.

You can journal the objects that are listed below:

- Access paths
- Database files
- Data areas
- Data queues
- Materialized query tables
- Byte stream files
- Directories
- Symbolic links
- Libraries

You can journal libraries like any other object. Journaling can start automatically for objects such as database files, data areas, and data queues that are created, moved into, or restored into a journaled library. The library's inheritance rules determine which objects to start journaling automatically and with what journaling attributes. You can replay changes to journaled libraries by using the Apply Journaled Changes (APYJRNCHG) command. To start journaling for a library, use the Start Journal Library (STRJRNLIB) command.

Related reference

"Saving database files" on page 73
This information describes what the system does when you save a database file.

Related information

Start Journal Library (STRJRNLIB) command

Saving changed objects when you use journaling:

Journal Management describes how to set up journaling. When you use journaling, the system uses one or more journal receivers to keep a record of changes that occur to the journaled objects.

If you are journaling data areas, data queues, or database files, you probably do not want to save those journaled objects when you save changed objects. You should save the journal receivers rather than the journaled objects. The journal receivers are a record of all the changes to the journaled objects.

The journaled objects (OBJJRN) parameter of the SAVCHGOBJ command controls whether the system saves journaled objects or not. If you specify *NO, which is the default, the system does not save an object if both of these conditions are true:

- The system journaled the object at the time specified for the REFDATE and REFTIME parameters on the SAVCHGOBJ command.
- The object is currently being journaled.

The OBJJRN parameter applies only to journaled data areas, data queues, and database files. It does not apply to journaled integrated file system objects.

If you save a journaled library using the SAVLIB command, the journaled changes are also saved. All the journaled objects in that library are also saved. Use the RSTLIB command to restore a journaled library.

Related information

Journal management

Saving journals and journal receivers:

This information describes the commands that you should use to save journals and journal receivers. It also contains some special considerations for you to keep in mind when using these commands.

Use the SAVOBJ, SAVCHGOBJ, or SAVLIB command to save journals and journal receivers that are in user libraries. Use the SAVSYS command to save the journals and journal receivers that are in the QSYS library.

You can save a journal or journal receiver even when you journal objects to it. The save operation always starts at the beginning of the journal receiver. If you save a journal receiver that is currently attached, you receive a diagnostic message.

If you specified MNGRCV(*USER) for a journal on the CRTJRN command or the CHGJRN command, save the detached receiver immediately after running the CHGJRN command.

If you specified MNGRCV(*SYSTEM), do one of the following:

- Set up a regular procedure for saving detached receivers. Use this procedure to determine which detached journal receivers that you need to save:
  1. Type `WRKJRNA JRN(`*`library-name/journal-name`*`)`
  2. On the Work with Journal Attributes display, press F15 (Work with receiver directory).
- Create a program to monitor for message CPF7020 in the journal's message queue. This saving sends this message when you detach the receiver. Save the receiver that the message identifies.

Related information

Journal management

Saving file systems

The **integrated file system** is a part of the i5/OS program that supports stream input/output and storage management similar to personal computers and UNIX® operating systems. The integrated file system also provides an integrating structure over all information that you store in the system.

You can view all objects on the system from the perspective of a hierarchical directory structure. However, in most cases, you view objects in the way that is most common for a particular file system. For example, you usually view objects in the QSYS.LIB file system from the perspective of libraries. You usually view objects in the QDLS file system as documents within folders.

Similarly, you should save objects in different file systems with the methods that are designed for each particular file system. You can find several good examples of how to use the SAV command in the CL reference information in the i5/OS Information Center.

Related information

SAV command in the CL reference information

Using the Save (SAV) command:

This information explains how to use the SAV command with the OBJ parameter.

The SAV command enables you to save the following data:

- A specific object
- A directory or subdirectory
- An entire file system
- Objects that meet search value

You can also save the items in this list by using the QsrSave API. For more information, refer to the API finder.

The Objects (OBJ) parameter on the SAV command supports the use of wildcard characters and the directory hierarchy. When you have a specific subset of similar objects within a directory subtree that you want to save, you can use the Name pattern (PATTERN) parameter to further define the objects that are identified in the (OBJ) parameter. For example, you could have a directory '/MyDir' that contains 100 subdirectories, Dir1 through Dir100, that each contain 100 .jpg files, Photo1.jpg through Photo100.jpg, with corresponding backup files, Photo1.bkp through Photo100.bkp. To save all of the .jpg files in '/MyDir', but omit the backup files, you could issue the following command:

```
SAV OBJ(('/MyDir')) PATTERN(('*.bkp' *OMIT))
```

When you use the SAV command to save the current directory **SAV OBJ('*')** and the current directory is empty (it has no files or subdirectories), the system does not save anything. The command does not save the one *DIR object that represents the current directory. However, when you explicitly specify the directory by name **SAV OBJ('/mydir')** you include the *DIR object in your save operation. The same applies to the home directory.

Another feature that the SAV command offers is the Scan objects (SCAN) parameter for purposes such as virus protection. If exit programs are registered with any of the integrated file system scan-related exit points, you can specify whether objects will be scanned while being saved. This parameter also enables you to indicate whether objects that previously failed a scan should be saved.

When you use the SAV command, you can specify OUTPUT(*PRINT) to receive a report of what the system saved. You can also direct the output to a stream file or to a user space. The SAV command does

not provide the option to create an output file. The Interpreting output from save (SAV) and restore (RST) topic describes output file format information from the SAV and RST commands.

Related concepts

"Interpreting output from save (SAV) and restore (RST)" on page 159
When you use the Save (SAV) command or the Restore (RST) command, you can direct output to a stream file or to a user space.

"Saving changed objects in directories" on page 70
This information describes how to use the CHGPERIOD parameter of the SAV command to save objects that have changed.

Related information

Integrated File System Scan on Close API (Exit Program)

Integrated File System Scan on Open API (Exit Program)

Integrated file system

Specifying the device name:

When you use the SAV command, you use a path name to specify objects to be saved. The path name consists of a sequence of directory names that are followed by the name of the object.

You also use the path name for the values of other parameters, such as the device (DEV) parameter. For example, on the SAVLIB command, you specify `DEV(TAP01)`. To use device TAP01 on the SAV command, you specify:

```
DEV('/QSYS.LIB/TAP01.DEVD')
```

To use a save file name MYSAVF in library QGPL on the SAV command, you specify:

```
DEV('/QSYS.LIB/QGPL.LIB/MYSAVF.FILE')
```

You might want to create symbolic links for devices that you specify with the SAV command to simplify keying and to reduce errors. For example, you can create a symbolic link for the media device description that is called either `TAP01` or `OPT01`. If you want to use symbolic links, it is recommended that you perform a one-time setup of symbolic links in the root (/) directory. For each tape device on your system, type the following:

```
ADDLNK  OBJ('/qsys.lib/media-device-name.devd') NEWLNK('/media-device-name') +
        LNKTYPE(*SYMBOLIC)
```

If the current directory is the root (/) directory, then an example of the SAV command using the symbolic link can be the following:

```
SAV  DEV('/media-device-name')
     OBJ(('/*') ('/QDLS' *OMIT) ('/QSYS.LIB' *OMIT))
```

All subsequent path names on the command need to begin from the root (/) directory.

Saving objects that have more than one name:

You can give more than one name to objects on the system. An additional name for an object is sometimes called a link. This information describes how linking works.

Some links, referred to as hard links, point directly to the object. Other links are more like a nickname for an object. The nickname does not point directly to the object. Instead, you can think of the nickname as an object that contains the name of the original object. This type of link is referred to as a soft link, or a symbolic link.

If you create links for objects, study the examples that follow to ensure that your save strategy saves both the contents of objects and all their possible names.

The following figure shows an example of a hard link: The root (/) directory contains UserDir. UserDir contains JCHDIR and DRHDIR. JCHDIR contains FILEA that has a hard link to Object A. DRHDIR contains FILEB which also contains a hard link to Object A.

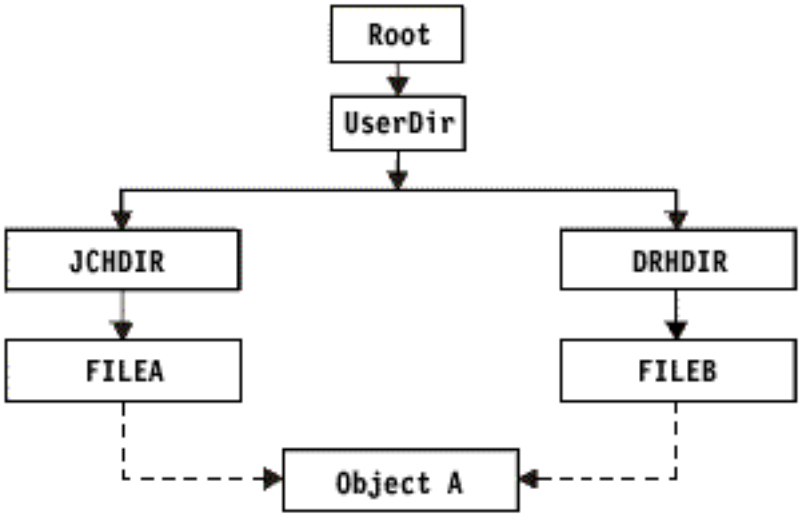


Figure 3. An Object with Hard Links–Example

You can save Object A with either of the following commands. For both commands, you get the description of the specified object and the contents of object.

- `SAV OBJ('/UserDir/JCHDIR/FILEA')`
- `SAV OBJ('/UserDir/DRHDIR/FILEB')`

If you use only the first command (JCHDIR), only the FILEA link name is saved for 'Object A'. The FILEB link name is not saved in this case. FILEB will not be found on the media, if it is specified on subsequent restore operations.

You can use any of the following commands to get the data once and both names (hard links) for the file:

- `SAV OBJ(('/UserDir'))`
- `SAV OBJ(('/UserDir/JCHDIR') ('/UserDir/DRHDIR'))`
- `SAV OBJ(('/UserDir/JCHDIR/FILEA') ('/UserDir/DRHDIR/FILEB'))`

The following figure shows an example of a symbolic link: The root (/) directory contains QSYS.LIB and Customer. QSYS.LIB contains CUSTLIB.LIB. CUSTLIB.LIB contains CUSTMAS.FILE. Customer is a symbolic link to CUSTMAS.FILE.

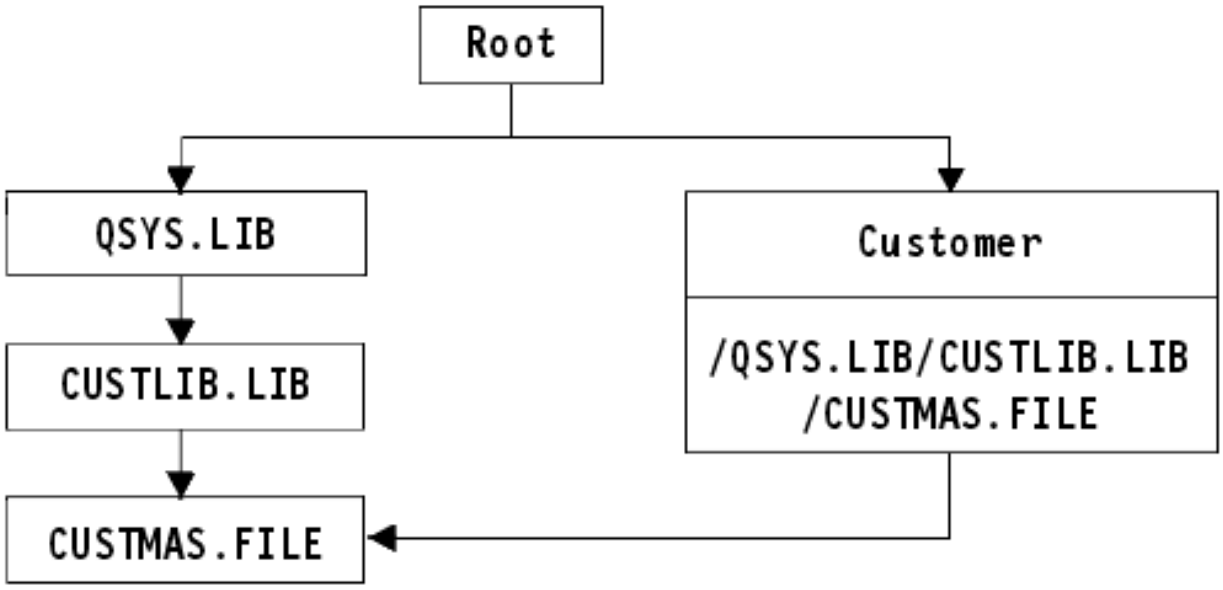


Figure 4. An Object with a Symbolic Link–Example

Following are several commands you can use to save the CUSTMAS file (both description and data):

- `SAVLIB LIB(CUSTLIB)`
- `SAVOBJ OBJ(CUSTMAS) LIB(CUSTLIB)`
- `SAV ('/QSYS.LIB/CUSTLIB.LIB/CUSTMAS.FILE')`
- `SAV ('/QSYS.LIB/CUSTLIB.LIB')`

None of these commands saves the fact that the CUSTMAS file has an alias of `customer` in the root (/) directory.

If you specify SAV OBJ('/customer'), you save the fact that customer is an alias for the CUSTMAS file. You do not save the description of the CUSTMAS file or its contents.

Saving across different types of file systems:

This information describes restrictions that apply when you use the SAV command to save objects from more than one file system at the same time.

- Different file systems support different types of objects and different methods of naming objects. Therefore, when you save objects from more than one file system with the same command, you cannot specify object names or object types. You can save all objects from all file systems, or you can omit some file systems. These combinations are valid:
  - Saving all objects on the system: `OBJ('/*')`

    Note: Using this command is not the same as using option 21 from the GO SAVE command menu. Following are the differences between SAV OBJ('/*') and option 21:
    - SAV OBJ('/*') does not put the system in a restricted state.
    - SAV OBJ('/*') does not start the controlling subsystem when it finishes.
    - SAV OBJ('/*') does not provide prompting to change default options.
  - Saving all objects in all file systems except the QSYS.LIB file system and the QDLS file system: `OBJ(('/*') ('/QSYS.LIB' *OMIT) ('/QDLS' *OMIT))`

  - Saving all objects in all files systems except the QSYS.LIB file system, the QDLS file system, and one or more other file systems: `OBJ(('/*') ('/QSYS.LIB' *OMIT) ('/QDLS' *OMIT) ('/other values' *OMIT))`
- Values for other parameters of the SAV command are supported only for some file systems. You must choose values that are supported by all file systems. Specify the following parameters and values:

  CHGPERIOD
  Default

  PRECHK
  *NO

  UPDHST
  *YES

  LABEL
  *GEN

  SAVACT
  *NO

  OUTPUT
  *NONE

  SUBTREE
  *ALL

  SYSTEM
  *LCL

  DEV Must be a tape device or an optical device
- The SAV OBJ('/*') command parameters require the following:
  - The system must be in a restricted state.
  - You must have *SAVSYS or *ALLOBJ special authority.
  - You must specify VOL(*MOUNTED).
  - You must specify SEQNBR(*END).

  Note: SAV OBJ('/*') is **not** the recommended method for saving the entire system. Use menu option 21 of the GO SAVE command to save the entire system.

When saving objects from the QSYS.LIB file system:

This information lists restrictions that apply when you use the SAV command to save objects from the QSYS.LIB (library) file system.

- The OBJ parameter must have only one name.
- The OBJ parameter must match the way that you can specify objects on the SAVLIB command and the SAVOBJ command:
  - You can save a library: `OBJ('/QSYS.LIB/library-name.LIB')`
  - You can save all the objects in a library: `OBJ('/QSYS.LIB/library-name.LIB/*')`
  - You can save all objects of a particular type in a library: `OBJ('/QSYS.LIB/library-name.LIB/*.object-type')`
  - You can save a specific object name and object type in a library: `OBJ('/QSYS.LIB/library-name.LIB/object-name.object-type')`
  - You can save all the members in a file by using either of the following:
    - `OBJ('/QSYS.LIB/library-name.LIB/file-name.FILE/*')`
    - `OBJ('/QSYS.LIB/library-name.LIB/file-name.FILE/*.MBR')`

- You can save a specific member in a file:

  ```
  OBJ('/QSYS.LIB/library-name.LIB/
      file-name.FILE/member-name.MBR')
  ```

- You can specify only the object types that the SAVOBJ command allows. For example, you cannot use the SAV command to save user profiles, because the SAVOBJ command does not allow OBJTYPE(*USRPRF).
- You cannot save some libraries in the QSYS.LIB file system with the SAVLIB command because of the type of information that they contain. Following are examples:
  - The QDOC library, because it contains documents
  - The QSYS library, because it contains system objects.

  You cannot use the SAV command to save these entire libraries:

Libraries that you cannot save using the SAV command

QDOC	QRPLOBJ	QSYS
QDOCxxxx[1]	QRPLxxxxx[2]	QSYSxxxxx[2]
QRECOVERY	QSRV	QTEMP
QRCYxxxxx[2]	QSPL	QSPLxxxx[3]

[1] Where xxxx is a value from 0002 to 0032, corresponding to an ASP.

[2] Where xxxxx is a value from 00033 to 00255, corresponding to an independent ASP.

[3] Where xxxxx is a value from 0002 to 0255, corresponding to an ASP.

- Other parameters must have these values:

  SUBTREE
  *ALL

  SYSTEM
  *LCL

  OUTPUT
  *NONE

  CHGPERIOD
  - Start date cannot be *LASTSAVE
  - End date must be *ALL
  - End time must be *ALL
  - Default, if you specify a file member

When saving objects from the QDLS file system:

This information lists restrictions that apply when you use the SAV command to save objects from the QDLS (document library services) file system.

- The OBJ and SUBTREE parameters must be one of the following:
  - `OBJ('/QDLS/path/folder-name') SUBTREE(*ALL)`
  - `OBJ('/QDLS/path/document-name') SUBTREE(*OBJ)`
- Other parameters must have these values:

  SYSTEM
  *LCL

  OUTPUT
  *NONE

  CHGPERIOD

- Start date cannot be *LASTSAVE
- End date must be *ALL
- End time must be *ALL
- Default, if `OBJ('/QDLS/path-name/document-name') SUBTREE(*ALL)` specified

PRECHK
*NO

UPDHST
*YES

SAVACT
Cannot be *SYNC

SAVACTMSGQ
*NONE

Backing up the integrated file system:

Learn how to improve your integrated file system backups.

Using concurrent backup operations

Reduce your backup windows by using multiple concurrent backups. To implement this approach you will need to determine some way to group your integrated file system data. Then you need separate SAV commands to concurrently save each of the subsets. You will need to consider the potential resource contention that can occur on the hardware resources being used. For example, performing concurrent backups on groups of data that are stored on the same set of disk units might cause contention on those disk units. You might decide to use multiple tape drives or a tape library system with multiple drives to run multiple concurrent SAV commands.

For more information about concurrent backups, see Saving to multiple devices to reduce your save window.

Related information

Save Object (SAV) command

Save Save Data File (SAVSAVFDTA) command

Auditing security on System i

Hierarchical Storage Management PDF

Using online backups: These topics are things that you can consider to use for online backups.

Using BRMS online backup of Lotus® servers

Backup, Recovery, and Media Services (BRMS) supports online backups of Lotus server databases (such as Domino for i5/OS and Quickplace). An online backup is a backup that you do while your Lotus server databases are in use; there are no save-while-active synchronization points. You can direct your online backups to a tape device, media library, save file, or a Tivoli Storage Manager server. BRMS can also create control groups that make it easy to use concurrent backups. Performing an online backup does not improve the performance of your backup. However, since your applications remain active, the duration of the backup is less important.

For more information about BRMS online backups, see Backup, Recovery, and Media Services.

If you decide to use the BRMS online backup support, you can tune the performance of the backup to your data. For more information, see Performance tuning on the BRMS web page.

Using save-while-active

The SAV command provides the SAVACT, SAVACTMSGQ, and SAVACTOPT parameters to support saving objects while active.

For more information, see Saving your system while it is active.

Backing up less data: These topics are things that you can consider to use for backing up less data.

Using the CHGPERIOD parameter to only save changed objects

The SAV command provides a CHGPERIOD parameter that can be used to find and save only objects that have changed. In some cases, this can be an effective way to reduce the amount of data you need to back up. However, the system still needs to look at each object to determine which objects have changed. If you have many files it might still take a long time to determine which objects have changed.

Structuring your directories to easily back up new files, omit data, or group your data

It might be beneficial to consider your backup strategy when you structure and name your directories. You might be able to group and name your files in some way that will make it easier to include or omit groups of directories or objects from your backups. You might want to group the directories such that you can back up all of the directories and files for an application, a user, or specified time period.

For example, if you are creating many files each day or each week, it might be useful to create a directory to contain the new files. Consider using a naming convention for the directories such that you can back up only the directory that contains the new objects or omit older directories.

Example: Create a directory structure that uses the year, month, and week to store new objects.

```
| /2008
| /2008/01
| /2008/01/01
| /2008/01/02
| /2008/01/03
| /2008/01/04
| /2008/02
```

Omitting objects from the backup

The SAV command provides the OBJ parameter that specifies the objects to be included and omitted from the backup. The OBJ parameter lets you specify a list of 300 values to be included or omitted from the SAV command. The values can be either specific directories or objects or generic values that provide wildcard support for the objects to be included or omitted.

Here are some examples of reasons why you might want to omit a directory or object from your backup:

- The directory or object is temporary and is not required if you need to recover your system.
- The directory or object is already backed up and has not changed since the last full backup.
- You are trying to group your integrated file system data so you can run multiple concurrent SAV commands.

Pattern Parameter

The SAV command provides the PATTERN parameter which lets you specify a list of 300 values which are used to group the save by either including or omitting objects which qualify for the save based on the OBJ parameter. The values can be either specific object names or generic values that provide wildcard support for the objects to be included or omitted.

Here are some examples of reasons why you might want to include or omit objects which qualify for the save based on the OBJ parameter:

- You want to save an entire directory tree, but omit objects of a specific type or name.
- You want to save all objects of a specific type without specifying which directories they might reside in.

Note: While less data might be saved, the amount of time to save the data might be increased. If patterns are specified on the PATTERN parameter, any object which qualifies for the save is compared to the list of objects on the PATTERN parameter.

Journal changes and save journal receivers

You can journal changes to directories, stream files, and symbolic links. If you set up journaling on your integrated file system data, you might need to change your save strategy. Your new strategy should be to back up the objects less frequently and instead back up the journal receivers that contain the changes you've made to the objects. This could reduce the amount of data you need to back up. However, you will need to understand and consider the implications to your recovery procedures.

Implement Hierarchical Storage Management (HSM)

If you have historical integrated file system data that is infrequently needed you might benefit from using Hierarchical Storage Management. Hierarchical Storage Management (HSM) automatically and transparently manages customer data across a storage hierarchy. The storage hierarchy can consist of high performance disk, compressed disk, and tape libraries.

When and how often data is accessed on your system depends on the type of data. A set of data that is currently being used might be accessed many times a day (hot data), or it might have become historical data which is accessed less frequently (cold data).

Through the Backup, Recovery, and Media Services (BRMS) user-defined policies, HSM can migrate or archive and dynamically retrieve infrequently used data or historical data up or down a hierarchy of storage devices.

Saving to save files (SAVF) then saving the SAVFs to tape with SAVSAVFDTA

Some customers have found that they can reduce their backup window by first backing up their data to a save file (SAVF) rather than saving directly to tape. Significant performance improvements were made to backups to save files. Of course if you back up to a save file, you need to have adequate disk space available for the save file. Chapter 15 of the System i Performance Capabilities Reference  can help you evaluate this approach on your system. You also will need to back up your save files to tape by using the Save Save File Data (SAVSAVFDTA) command. However, the SAVSAVFDTA command does not need to be completed during your backup window.

Reducing or eliminating auditing during backup or recovery operations

Measurements show that performing security auditing during save or restore operations (*SAVRST auditing) can decrease performance. Auditing provides valuable information about the actions being performed on your system and who is performing those actions. However, you need to balance the value of that information against the time you have available to perform a backup or recovery. This is especially true if you need to recover all or many objects in the integrated file system.

Reducing the number of objects scanned during the backup

Specifying the SCAN parameter during the backup might have significant performance impacts if scanning is enabled on the system. Scanning objects might be a valuable part of your system security, but

you need to consider the amount of time scanning will add to your backup window.

Saving user-defined file systems

A *user-defined file system (UDFS)* is a file system that you can create and manage yourself. You can create multiple UDFSs, with unique names.

When you use the Create User-Defined File System (CRTUDFS) command to create a UDFS, you can set the following attributes for it:

- Auditing value for objects
- Auxiliary storage pool (ASP) number where you store objects in the UDFS
- Case sensitivity for UDFS names
- Default file format
- Public authority for data and object
- Restricted rename and unlink
- Scanning option for objects
- Text description

The ASP number is determined by the directory that contains the UDFS. For example, '/dev/QASP01/MyUdfs1.udfs' indicates that MyUdfs1 is in the system ASP. The file name '/dev/MyASP/MyUdfs2.udfs' indicates MyUdfs2 is in the independent disk pool MyASP.

Note: If the UDFS is on an independent disk pool, ensure that the independent disk pool is varied on and that the UDFS is unmounted before you start the save operation.

Related information

Types of disk pools

Create User-Defined FS (CRTUDFS) command

How the system stores user-defined file systems:

As in the root (/) and QOpenSys file systems, in a user-defined file system (UDFS), users can create directories, stream files, symbolic links, and local sockets.

A single block special file object (*BLKSF) represents a UDFS. When you create a UDFS, the system also creates an associated block special file. You can only access the block special file through integrated file system commands, application programming interface (API), and the QFileSvr.400 interface. Block special file names must be of the form:

```
/dev/QASPxx/udfs_name.udfs
```

Where `xx` is the system or basic ASP number (1–32) where the user stores the UDFS and `udfs_name` is the unique name of the UDFS. Note that the UDFS name must end in the `.udfs` extension. If the UDFS is stored in an independent ASP, the block special file name will be of the form:

```
/dev/device-description/udfs_name.udfs
```

A UDFS exists only in two states: mounted and unmounted. When you mount a UDFS, you can access the objects within it. When you unmount a UDFS, you cannot access the objects within it.

To access the objects within a UDFS, you must mount the UDFS on a directory (for example, `/home/JON`). When you mount a UDFS on a directory, you cannot access the original contents of that directory. Also, you can access the contents of the UDFS through that directory. For example, the `/home/JON` directory contains a file `/home/JON/payroll`. A UDFS contains three directories mail, action, and outgoing. After mounting the UDFS on `/home/JON`, the `/home/JON/payroll` file is inaccessible, and the three directories become accessible as `/home/JON/mail`, `/home/JON/action`, and `/home/JON/outgoing`. After you unmount the UDFS, the `/home/JON/payroll` file is accessible again, and the three directories in the UDFS become inaccessible.

Related information

OS/400 Network File System Support

Saving an unmounted UDFS:

If you omit the RBDMFS parameter, you should unmount any user-defined file systems (UDFS) before you perform a save or restore operation. Use the DSPUDFS command to determine if you mounted a UDFS or if you unmounted a UDFS.

The system saves objects from an unmounted UDFS if you specify the *BLKSF for the UDFS which is contained in an ASP or independent ASP (/dev/qasp*xx*). The system saves information about the UDFS (for example, the ASP number, authority, and case sensitivity).

To save an unmounted UDFS, specify:

```
SAV  OBJ(('/dev/QASP02/udfs_name.udfs'))
```

You can use two methods omit objects from an unmounted UDFS during a save operation. You can either use the *OMIT option on the OBJ parameter or the PATTERN parameter on the SAV command. By omitting certain objects, such as Domino data or temporary objects, for example, you can reduce your backup window while saving an unmounted UDFS.

This example uses the *OMIT option on the OBJ parameter to omit UDFS objects that begin with 'b' from the top-level directory of the UDFS from the save operation:

```
SAV  DEV(jssavf) OBJ(('/dev/qasp01/js.udfs') ('/dev/qasp01/js.udfs/b*' *OMIT))
```

This example uses the PATTERN parameter to omit UDFS objects that begin with 'b' from any directory in the UDFS from the save operation:

```
SAV  DEV(jssavf) OBJ(('/dev/qasp01/js.udfs')) PATTERN(('b*' *OMIT))
```

Restrictions when you save an unmounted UDFS

The following are some restrictions that you must take into consideration when saving an unmounted UDFS.

1. You cannot specify individual objects from UDFSs for the object (OBJ) parameter on a SAV command.
2. You cannot view or work with objects in an unmounted UDFS. Therefore, you cannot determine the amount of storage or time that the system requires for the save operation after you unmount the UDFS.
3. SUBTREE(*ALL) is required.

Related information

Restoring an unmounted UDFS

Saving a mounted UDFS:

When you save a mounted UDFS, both the UDFS information and objects within the UDFS are saved. You can restore either just the objects within the mounted UDFS, or restore both the UDFS information (/dev/*asp/udfs_name.udfs*) and the objects within the UDFS.

Ordinarily, you should unmount user-defined file systems (UDFS) before save and restore operations. Menu options 21, 22, and 23 of the GO SAVE command provide an option to unmount UDFSs before the save operation.

To save a mounted UDFS, specify the following command:

```
SAV  OBJ(('/appl/dir1')
```

The system mounted the UDFS over directory /appl/dir1.

If a save operation includes objects from mounted UDFSs, file system information is saved. You can restore a mounted UDFS by specifying the RBDMFS(*UDFS) parameter on the RST command. The RBDMFS parameter rebuilds the mounted file system during the restore operation.

However, if you omit the RBDMFS parameter or specify RBDMFS(*NONE), only the objects contained in the directory are restored and none of the file system information is restored.

Related information

Restoring a mounted UDFS

Restore actions for mounted user-defined file systems

Saving document library objects

The system provides the capability to store documents and folders in a hierarchy (documents within a folder within another folder). Document library objects (DLOs) are documents and folders.

The following topics tell you:

Related reference

"Methods for saving distribution objects" on page 96
This information describes common save methods for distribution objects.

"Methods for saving IBM-supplied document library objects and folders" on page 99
This information describes common save methods for saving IBM-supplied document library objects.

How the system stores and uses document library objects:

The system provides the capability to store documents and folders in a hierarchy (documents within a folder within another folder). Document library objects (DLOs) are documents and folders.

To simplify storage management, the system stores all DLOs in one or more libraries. The name of the library in the system ASP is QDOC. Each user ASP that contains DLOs has a document library called QDOCnnnn, where `nnnn` is the number that is assigned to the ASP. From a user perspective, DLOs are not in libraries. The system files them in folders. You manipulate DLOs by using DLO commands and menus.

Some licensed programs use DLO support.

Within the integrated file system, the QDLS (Document Library Services) file system provides DLO support.

The system uses a set of search index files in the QUSRSYS library to keep track of all the DLOs on the system. The names of these database files begin with the characters `QAOSS`. The system uses other QAO* files in the QUSRSYS library to track distributions and support text search capabilities. You should periodically save these files in QUSRSYS. Menu options 21 and 23 of the GO SAVE command save both library QUSRSYS and all the DLOs on the system.

You can use the Save Document Library Object (SAVDLO) command to manually save one or more documents. This does not affect documents unless you specify the settings to free or delete storage. You can save a single document or more than one document.

Methods for saving multiple documents:

You can save all of your documents, save all documents in a list of folders, or save all documents in an auxiliary storage pool (ASP).

- Save all of your documents by typing: `SAVDLO DLO(*ALL) FLR(*ANY)`.
- Save all documents in a list of folders by typing: `SAVDLO DLO(*ALL) FLR(`*`folder`*`)`. You can specify up to 300 generic or specific folder names on the Folder (FLR) parameter.

- You can run multiple SAVDLO commands concurrently for documents within a single ASP or in multiple ASPs. You can run one or more SAVDLO commands concurrently with one or more Restore Document Library Object (RSTDLO) commands that use the same ASP. Here is an example of running concurrent SAVDLO operations with generic values:

  ```
  SAVDLO  DLO(*ANY) DEV(first-device) FLR(A* B* C* ...L*)
  SAVDLO  DLO(*ANY) DEV(second-device) FLR(M* N* O* ...Z*)
  ```

- Save all documents in an ASP by typing: `SAVDLO DLO(*ALL) FLR(*ANY) ASP(n)`.

  You might want to move the folders that contain user documents to user ASPs. You can save the document library objects (DLOs) in those ASPs regularly and not save the system ASP. This eliminates the extra time and media for saving the system folders for IBM System i Access Family, which change infrequently.

  Note: When you save System i Access Family, you must also run the SAV command. The following shows all the parameters that are needed to save everything in the integrated file system, including System i Access Family.

  ```
  SAV  DEV('/QSYS.LIB/media-device-name.DEVD')
       OBJ(('/*') +
          ('/QSYS.LIB' *OMIT)
          ('/QDLS' *OMIT))
       UPDHST(*YES)
  ```

- Save a list of documents, by user-defined name or by system object name.
- Save all documents that meet certain search values. The following table shows the parameters you can use if you specify DLO(*SEARCH).

*Table 26. Parameters for DLO(*SEARCH)*

Parameter	Definition
FLR	Folder
SRCHTYPE	*ALL, for all folders that meet the search criteria
CHKFORMRK	Marked for offline storage
CHKEXP	Document expiration date
CRTDATE	Creation date
DOCCLS	Document class
OWNER	Owner
REFCHGDATE	Document last changed date
REFCHGTIME	Document last changed time

- Save all distribution objects (mail) by typing: `SAVDLO DLO(*MAIL)`.
- Save all distribution objects, new folders, new documents, and changed documents by typing: `SAVDLO DLO(*CHG)`. This is another method for reducing the effect of online information about the amount of time and media that it takes to save DLOs. Save document library objects (DLOs) provides more information about specifying `DLO(*CHG)`.

You can use the OMITFLR parameter to exclude folders from the save operation. The OMITFLR parameter will allow up to 300 generic or specific folder names.

The OMITFLR parameter is useful if you want to omit folders that never change or only change infrequently. You can also use it to remove a group of folders from one save operation while you concurrently save that group to a different media device.

When you save DLOs from more than one ASP with the same operation, the system creates a separate file on the media for each ASP. When you restore DLOs from the media, you must specify the sequence numbers to restore the DLOs from more than one ASP.

Authority that is required for the SAVDLO command: The following parameter combinations for the SAVDLO command require either *ALLOBJ special authority, *SAVSYS special authority, or *ALL authority to the documents. You also need enrollment in the system directory:

- `DLO(*ALL) FLR(*ANY)`
- `DLO(*CHG)`
- `DLO(*MAIL)`
- `DLO(*SEARCH) OWNER(*ALL)`
- `DLO(*SEARCH) OWNER(`*`user-profile-name`*`)`

Note: You can always save your own DLOs. You must have the authorities that are specified to specify another user profile for the owner parameter.

Methods for reducing disk space that is used by documents:

Documents tend to accumulate and require more and more storage. This information describes different methods that you can use to reduce disk space that is used for documents.

- Saving documents and delete them (STG(*DELETE)). These documents no longer appear when you search for documents.
- Saving documents and free storage (STG(*FREE)). These documents appear when you search and the system marks them as offline.
- Moving documents to a user ASP. You can establish different backup strategies and different recovery strategies for these user ASPs.
- Using the Reorganize Document Library Object (RGZDLO) command.

When you save documents, specify search values such as the storage mark on the document or the document expiration date to identify which documents should have their storage freed.

Related concepts

"Freeing storage when saving" on page 5
Freeing storage when saving explains how to use the STG parameter to remove an object from your system after you save it. This only works with a limited number of commands.

Output from the SAVDLO command:

You can use the OUTPUT parameter on the SAVDLO command to show information about the saved documents, folders, and mail. You can either print the output (OUTPUT(*PRINT)) or save it to a database file (OUTPUT(*OUTFILE)).

If you print the output, you should be aware of device dependencies:

- The heading information in the output is device-dependent. All information does not appear for all devices.
- The printer file for the SAVDLO command uses a character identifier (CHRID) of 697 500. If your printer does not support this character identifier, the system displays message CPA3388. To print the SAVDLO output and not receive message CPA3388, specify the following before specifying *PRINT on the SAVDLO command:

```
CHGPRTF FILE(QSYSOPR/QPSAVDLO) CHRID(*DEV)
```

If you use an output file, the system uses the file format QSYS/QAOJSAVO.OJSDLO.

Related information

Printing

Saving and restoring spooled files

For i5/OS V5R4 or later, you can use any of the methods described here to save and restore spooled files. This information contains a table that lists the commands and APIs in order of preference.

For releases prior to V5R4, you must use indirect methods for saving and restoring spooled files. These indirect methods might not preserve all of the attributes.

Table 27. Save and restore spooled files

Save Methods	Restore Methods	Spooled file attributes preserved	When used
SAVLIB, SAVOBJ commands SAVRSTLIB, SAVRSTOBJ commands, QSRSAVO API, Save menu options 21-23	RSTLIB, RSTOBJ commands SAVRSTLIB, SAVRSTOBJ commands, QSRRSTO API, Restore menu options 21-23	Data and all attributes	i5/OS V5R4 and later
QSPOPNSP, QSPGETSP, QUSRSPLA APIs	QSPCRTSP, QSPPUTSP, QSPCLOSP APIs	Data, but not all attributes	Any release
CPYSPLF, SAVOBJ commands	CPYF command	Textual data only	Any release

When you save an output queue with the save commands, menu, or QSRSAVO API, you can choose to save all its spooled files. You can do this by specifying *ALL for the Spooled file data (SPLFDTA) command parameter, menu prompt or API key. When you restore output queues with the restore commands, menu, or QSRRSTO API, you can choose to restore any saved spooled files that do not already exist on the system. You can do this by specifying *NEW for the SPLFDTA parameter, prompt, or key. With the QSRSAVO and QSRRSTO APIs, you can also choose to save or restore spooled files by using a set of selection criteria. If you save spooled files with the QSRSAVO API using selection criteria and the *SPLF special library value, then you must use the QSRRSTO API with the *SPLF special library value to restore the spooled files.

This example describes how to save spooled files:

1. Create an output queue to store the spooled files.
   ```
   CRTOUTQ  OUTQ(lib-name/que-name)
   ```
2. Use the Work with Spooled File (WRKSPLF) command to list the spooled files.
3. Use option 2, Change Spooled File Attributes (CHGSPLFA) command to move the spooled files you want to save to the output queue you created.
4. Use the Save Object (SAVOBJ) command to save the spooled file data.
   ```
   SAVOBJ  OBJ(que-name) LIB(lib-name) DEV(dev-name) OBJTYPE(*OUTQ) SPLFDTA(*ALL)
   ```

This example describes how to restore spooled files:

1. Restore spooled files that are not already on the system.
   ```
   RSTOBJ  OBJ(que-name) SAVLIB(lib-name) DEV(dev-name)
           OBJTYPE(*OUTQ) SPLFDTA(*NEW)
   ```

Using spooled file APIs

If your source or target system is earlier than V5R4, you can use spooled file APIs as an indirect method to save and restore spooled files. This method preserves the spooled file data stream but not all of the attributes.

To save spooled files:

- The spooled files are opened using the Open Spooled File (QSPOPNSP) API.

- The spooled file data is retrieved using the Get Spooled File Data (QSPGETSP) API.
- The spooled file attributes are retrieved using the User Spooled File Attributes (QUSRSPLA) API.

To restore spooled files:

1. The spooled files are created using the Create Spooled File (QSPCRTSP) API .
2. The spooled file data is written to a new spooled file using the Put Spooled File Data (QSPPUTSP) API.
3. The spooled file is closed using the Close Spooled File (QSPCLOSP) API.

You can find an example and a tool for using these APIs in the QUSRTOOL library in the TSRINFO member of the QATTINFO file.

Copying spooled files to database files

If your source or target system is earlier than V5R4, you can copy data between spooled files and database files as an indirect method to save and restore spooled files. This method copies textual data only and not advanced function attributes such as graphics and variable fonts. This method does not provide a complete solution for saving your spooled files.

The Copy Spooled File (CPYSPLF) command saves spooled file data to a database file. The Copy File (CPYF) command can copy data from a database file to a spooled file.

Related tasks

"GO SAVE: Option 21 (saving the entire system)" on page 31
Option 21 saves everything on your system and allows you to perform the save while you are not there.

"GO SAVE: Option 23 (saving user data)" on page 32
Option 23 saves all user data. This information includes files, records, and other data that your users supply into your system.

"Performing a complete save using the GO SAVE checklist" on page 34
Use this checklist to perform a complete save operation.

Related information

Save and restore spooled files

Restoring previous release user data to a new system

API finder

Copy Spooled File (CPYSPLF) command

Methods for saving user data

You can use these link references to learn how you can save user data in your system.

An easy way to save all of your user data is with GO SAVE command, menu option 23.

The following commands enable you to manually save user data:

- Save Security Data (SAVSECDTA)
- Save Configuration (SAVCFG)
- Save Library (SAVLIB *ALLUSR)
- Save Document Library Object (SAVDLO)
- Save Object (SAV)

Related tasks

"GO SAVE: Option 23 (saving user data)" on page 32
Option 23 saves all user data. This information includes files, records, and other data that your users supply into your system.

Related information

Save Configuration (SAVCFG) command

Save Changed Objects (SAVCHGOBJ) command

Save Document Library Object (SAVDLO) command

Save Library (SAVLIB) command

Save Object (SAVOBJ) command

Save (SAV) command

Methods for saving user libraries:

This information describes common save methods for saving user libraries.

Table 28. User libraries information

Item description	When changes occur	Contains user data or changes?	IBM-supplied data?
User libraries	User libraries change regularly.	Yes	No

Common save method for user libraries	Requires restricted state?
SAVLIB *NONSYS	Yes
SAVLIB *ALLUSR	No
SAVLIB *library-name*	No[1]
SAVCHGOBJ	No[1]
GO SAVE command, menu option 21	Yes
GO SAVE command, menu option 23	No[1, 2]

[1] **Important:** For procedures where the system does not require a restricted state, you must ensure that the system can get the locks necessary to save the information. You should put your system in a restricted state whenever you save multiple libraries, documents, or directories, unless you use the save-while-active function.

[2] When you use option 23 from the GO SAVE command menu, the default is to place your system in a restricted state. If you choose the prompting option, you can cancel the display that puts your system in a restricted state.

These library objects change when you update licensed programs.

"Saving libraries with the SAVLIB command" on page 51 explains how to save one or more libraries. This information also includes special SAVLIB parameters and how to select libraries on your system.

Related concepts

"Save-while-active function" on page 113
The save-while-active function allows you to use your system during all or part of the save process, that is, save your system while it is active.

Related tasks

"GO SAVE: Option 21 (saving the entire system)" on page 31
Option 21 saves everything on your system and allows you to perform the save while you are not there.

"GO SAVE: Option 23 (saving user data)" on page 32
Option 23 saves all user data. This information includes files, records, and other data that your users supply into your system.

Related information

Save Library (SAVLIB) command

Save Changed Objects (SAVCHGOBJ) command

Methods for saving Q libraries that contain user data:

This information describes common save methods for Q libraries that contain data.

Table 29. Q libraries that contain user data information

Item description	When changes occur	Contains user data or changes?	IBM-supplied data?
Q libraries that contain user data include QGPL, QUSRSYS, QDSNX, and others. "Special values for the SAVLIB command" on page 52 includes a complete list of Q libraries that contain user data.	These libraries change regularly.	Yes	Yes

| To save the system directory files, you must end the QSNADS subsystem before saving the QUSRSYS
| library. You might also need to end the QSYSWRK, QSERVER, and ENDTCPSVR(*MGTC *DIRSRV)
| subsystems to save the QAO* files.

Common save method for Q libraries that contain user data	Requires restricted state?
SAVLIB *NONSYS	Yes
SAVLIB *ALLUSR	No[1]
SAVLIB *library-name*	No[1]
SAVCHGOBJ	No[1]
GO SAVE command, menu option 21	Yes
GO SAVE command, menu option 23	No[1, 2]

[1] **Important:** For procedures where the system does not require a restricted state, you must ensure that the system can get the locks necessary to save the information. You should put your system in a restricted state whenever you save multiple libraries, documents, or directories, unless you use the save-while-active function.

[2] When you use option 23 from the GO SAVE command menu, the default is to place your system in a restricted state. If you choose the prompting option, you can cancel the display that puts your system in a restricted state.

"Saving libraries with the SAVLIB command" on page 51 explains how to save one or more libraries. This information also includes special SAVLIB parameters and how to select libraries on your system.

Related concepts

"Save-while-active function" on page 113
The save-while-active function allows you to use your system during all or part of the save process, that is, save your system while it is active.

Related tasks

"GO SAVE: Option 21 (saving the entire system)" on page 31
Option 21 saves everything on your system and allows you to perform the save while you are not there.

"GO SAVE: Option 23 (saving user data)" on page 32
Option 23 saves all user data. This information includes files, records, and other data that your users supply into your system.

Related information

Save Library (SAVLIB) command

Save Changed Objects (SAVCHGOBJ) command

Methods for saving distribution objects:

This information describes common save methods for distribution objects.

Table 30. Distribution objects information

Item description	When changes occur	Contains user data or changes?	IBM-supplied data?
Distribution objects	Distribution objects in QUSRSYS change regularly.	Yes	No

Common save method for distribution objects	Requires restricted state?
SAVDLO	No[1]
GO SAVE command, menu option 21	Yes
GO SAVE command, menu option 23	No[1, 2]
GO SAVE command, menu option 30	Yes
GO SAVE command, menu option 32	Yes

[1] **Important:** For procedures where the system does not require a restricted state, you must ensure that the system can get the locks necessary to save the information. You should put your system in a restricted state whenever you save multiple libraries, documents, or directories, unless you use the save-while-active function.

[2] When you use option 23 from the GO SAVE command menu, the default is to place your system in a restricted state. If you choose the prompting option, you can cancel the display that puts your system in a restricted state.

Related concepts

"Save-while-active function" on page 113
The save-while-active function allows you to use your system during all or part of the save process, that is, save your system while it is active.

Related tasks

"GO SAVE: Option 21 (saving the entire system)" on page 31
Option 21 saves everything on your system and allows you to perform the save while you are not there.

"GO SAVE: Option 23 (saving user data)" on page 32
Option 23 saves all user data. This information includes files, records, and other data that your users supply into your system.

Related reference

"Saving document library objects" on page 89
The system provides the capability to store documents and folders in a hierarchy (documents within a folder within another folder). Document library objects (DLOs) are documents and folders.

Related information

Save Document Library Object (SAVDLO) command

Methods for saving network server storage spaces:

This information describes common methods for saving network server storage spaces.

You can save and restore network server storage spaces, also known as virtual disks, that are associated with an integrated Windows server or integrated Linux server.

Table 31. Network server storage spaces information

Item description	When changes occur	Contains user data or changes?	IBM-supplied data?
Network server storage spaces	Network server storage spaces for integrated server licensed programs (QFPNWSSTG directory) change regularly.	Yes	Yes

Common save method for network server storage spaces	Requires restricted state?
SAV[4]	No
GO SAVE command, menu option 21[1]	Yes
GO SAVE command, menu option 23[1]	No[2, 3]

[1] You must vary off the network servers. You can perform this option from the GO SAVE command menu if you select option 21, 22, or 23. Select the integrated server you want to vary off from the Specify Command Defaults screen.

[2] When you use option 23 from the GO SAVE command menu, the default is to place your system in a restricted state. If you choose the prompting option, you can cancel the display that puts your system in a restricted state.

[3] **Important:** For procedures where the system does not require a restricted state, you must ensure that the system can get the locks necessary to save the information. You should put your system in a restricted state whenever you save multiple libraries, documents, or directories, unless you use the save-while-active function.

[4] Network server storage spaces can be linked to the following network server descriptions: *IXSVR, *ISCSI, and *GUEST NWSD. You can use the save-while-active function when the disk is linked to *ISCSI and *GUEST NWSD, but not when the disk is linked to *IXSVR.

Related tasks

"Using save-while-active with network server storage spaces" on page 120
You can use the save-while-active function to save the network server storage spaces while keeping the integrated server online. The save-while-active function reduces or eliminates your outage for save operations.

"GO SAVE: Option 21 (saving the entire system)" on page 31
Option 21 saves everything on your system and allows you to perform the save while you are not there.

"GO SAVE: Option 23 (saving user data)" on page 32
Option 23 saves all user data. This information includes files, records, and other data that your users supply into your system.

"Saving data for IXS and IXA-attached integrated Windows servers" on page 106
You can do a full-system backup of the integrated Windows server to i5/OS, save individual Windows files and directories, or save the network server description, configuration objects, and the associated disk drives.

"Saving data for iSCSI-attached integrated servers" on page 106
You can do a full-system backup of an iSCSI-attached integrated server to i5/OS, or save the network

server description, configuration objects, and the associated disk drives. You also can back up individual files and directories for the integrated Windows server and the integrated Linux server.

Related information

Save Object (SAV) command

Methods for saving user-defined file systems:

This information describes common save methods for user-defined file systems information.

Table 32. User-defined file systems information

Item description	When changes occur	Contains user data or changes?	IBM-supplied data?
User-defined file systems	User-defined file systems change regularly.	Yes	Some

You might want to unmount all user-defined file systems (UDFSs) before you perform the save operation. You can perform this option from the GO SAVE command menu if you select option 21, 22, or 23. Then select **Y** at the *Unmount file systems* prompt on the Specify Command Defaults display.

When you save a mounted UDFS, all file system information is saved. You can restore a mounted UDFS by specifying the RBDMFS(*UDFS) parameter on the RST command.

Common save method for user-defined file systems (UDFS)	Requires restricted state?
SAV	No[1]
GO SAVE command, menu option 21	Yes

[1] **Important:** For procedures where the system does not require a restricted state, you must ensure that the system can get the locks necessary to save the information. You should put your system in a restricted state whenever you save multiple libraries, documents, or directories, unless you use the save-while-active function.

Related tasks

"GO SAVE: Option 21 (saving the entire system)" on page 31
Option 21 saves everything on your system and allows you to perform the save while you are not there.

Related information

Save Object (SAV) command

Methods for saving directories in the Root and the QOpenSys file systems:

This information describes common save methods for directories in the Root and the QOpenSys file systems information.

Table 33. Directories in the Root and the QOpenSys file systems information

Item description	When changes occur	Contains user data or changes?	IBM-supplied data?
Directories in the Root and the QOpenSys file systems	Directories in the Root and QOpenSys file systems change regularly.	Yes	Some

Common save method for directories in the Root and the QOpenSys file systems	Requires restricted state?
SAV	No
GO SAVE command, menu option 21	Yes
GO SAVE command, menu option 23	No[1, 2]

[1] When you select menu option 23 of the GO SAVE command, the command menu option places your system in a restricted state by default. If you choose the prompting option, you can cancel the display that puts your system in a restricted state.

[2] **Important:** For procedures where the system does not require a restricted state, you must ensure that the system can get the locks necessary to save the information. You should put your system in a restricted state whenever you save multiple libraries, documents, or directories, unless you use the save-while-active function.

Related concepts

"Save-while-active function" on page 113
The save-while-active function allows you to use your system during all or part of the save process, that is, save your system while it is active.

Related tasks

"GO SAVE: Option 21 (saving the entire system)" on page 31
Option 21 saves everything on your system and allows you to perform the save while you are not there.

"GO SAVE: Option 23 (saving user data)" on page 32
Option 23 saves all user data. This information includes files, records, and other data that your users supply into your system.

Related information

Save Object (SAV) command

Methods for saving IBM-supplied document library objects and folders:

This information describes common save methods for saving IBM-supplied document library objects.

Table 34. IBM-supplied document library objects and folders information

Item description	When changes occur	Contains user data or changes?	IBM-supplied data?
IBM-supplied document library objects and folders (usually start with Q, used by IBM System i Access Family)	These library objects change when you update licensed programs.	No[1]	Yes

[1] You should avoid changing objects or storing user data in these IBM-supplied libraries or folders. You could lose or destroy these changes when you install a new release of the operating system. If you make changes to objects in these libraries, note them carefully in a log for future reference.

Common save method for IBM-supplied document library objects and folders	Requires restricted state?
SAVDLO[2]	No[3]
GO SAVE command, menu option 21	Yes
GO SAVE command, menu option 23	No[3, 4]
GO SAVE command, menu option 30	Yes

Common save method for IBM-supplied document library objects and folders	Requires restricted state?
GO SAVE command, menu option 32	Yes

[2] To ensure that the system saves all System i Access Family data, end subsystem QSERVER.

[3] **Important:** For procedures where the system does not require a restricted state, you must ensure that the system can get the locks necessary to save the information. You should put your system in a restricted state whenever you save multiple libraries, documents, or directories, unless you use the save-while-active function.

[4] When you use option 23 from the GO SAVE command menu, the default is to place your system in a restricted state. If you choose the prompting option, you can cancel the display that puts your system in a restricted state.

Related concepts

"Save-while-active function" on page 113
The save-while-active function allows you to use your system during all or part of the save process, that is, save your system while it is active.

"Saving changed document library objects" on page 71
You can use the Save Document Library Object (SAVDLO) command to save DLOs that have changed since a particular time.

Related tasks

"GO SAVE: Option 21 (saving the entire system)" on page 31
Option 21 saves everything on your system and allows you to perform the save while you are not there.

"GO SAVE: Option 23 (saving user data)" on page 32
Option 23 saves all user data. This information includes files, records, and other data that your users supply into your system.

Related reference

"Saving document library objects" on page 89
The system provides the capability to store documents and folders in a hierarchy (documents within a folder within another folder). Document library objects (DLOs) are documents and folders.

Related information

Save Document Library Object (SAVDLO) command

Methods for saving user document library objects and folders:

This information describes common save methods for saving user document library objects.

Table 35. User document library objects and folders information

Item description	When changes occur	Contains user data or changes?	IBM-supplied data?
User document library objects and folders	User document library objects and folders change regularly.	Yes	Some

Common save method for user document library objects and folders	Requires restricted state?
SAVDLO	No
GO SAVE command, menu option 21	Yes
GO SAVE command, menu option 23	No[1, 2]

Common save method for user document library objects and folders	Requires restricted state?
GO SAVE command, menu option 30	Yes
GO SAVE command, menu option 32	Yes

[1] When you use option 23 from the GO SAVE command menu, the default is to place your system in a restricted state. If you choose the prompting option, you can cancel the display that puts your system in a restricted state.

[2] **Important:** For procedures where the system does not require a restricted state, you must ensure that the system can get the locks necessary to save the information. You should put your system in a restricted state whenever you save multiple libraries, documents, or directories, unless you use the save-while-active function.

Related tasks

"GO SAVE: Option 21 (saving the entire system)" on page 31
Option 21 saves everything on your system and allows you to perform the save while you are not there.

"GO SAVE: Option 23 (saving user data)" on page 32
Option 23 saves all user data. This information includes files, records, and other data that your users supply into your system.

Related information

Save Document Library Object (SAVDLO) command

Methods for saving IBM-supplied directories without user data:

This information describes common save methods for IBM supplied directories without user data information.

Table 36. IBM-supplied directories without user data information

Item description	When changes occur	Contains user data or changes?	IBM-supplied data?
IBM-supplied directories without user data	IBM-supplied directories without user data change when you apply Program Temporary Fixes (PTFs). They also change when you install a new release of the operating system, or when you update licensed programs.	No	Yes

Common save method for IBM-supplied directories without user data	Requires restricted state?
SAV	Yes
GO SAVE command, menu option 21	Yes
GO SAVE command, menu option 22	Yes

Related tasks

"GO SAVE: Option 21 (saving the entire system)" on page 31
Option 21 saves everything on your system and allows you to perform the save while you are not there.

"GO SAVE: Option 22 (saving system data)" on page 32
Option 22 saves only your system data. It does not save any user data. Option 22 puts your system into a restricted state. This means that no users can access your system, and the backup is the only thing that is running on your system.

Related information

Save Object (SAV) command

Saving logical partitions and system applications

With logical partitions, you can distribute resources within a single system to make it function as if it were two or more independent systems. You can back up each logical partition separately, or as a set of connected systems.

System i models support three types of logical partitions:

- Primary and secondary partitions on System i 270 and 8*xx* models. Each logically-partitioned system has one primary partition and one or more secondary partitions.
- Integrated servers running on an i5/OS partition. An integrated server is a combination of integrated server hardware, network components, virtual disks, shared devices, and i5/OS integrated server configuration objects.
- System partitions on systems that use POWER5™ or POWER6™ processors. You can install AIX, Linux, and i5/OS operating systems on these systems.

  Attention: If you are using the Hardware Management Console (HMC), you must back up the HMC in addition to saving the individual logical partitions.

The diagram shows the save commands that can be used for different file systems:

- The root (/) file system is saved with SAV.
- QSYS.LIB can be saved with SAVSYS, SAVCFG, SAVSECDTA, SAVLIB, SAVOBJ, SAVCHGOBJ, or SAV.
- QDLS (Document library services) can be saved with SAVDLO, or SAV.
- QOpenSys (Open systems) is saved with SAV.
- Domino server data directory is saved with SAV.
- User-defined file systems (/dev/QASP*xx*/) or (/dev/*asp-name*/) are saved with SAV.
- Other file systems, such as QNTC for Linux, are saved with SAV as well.

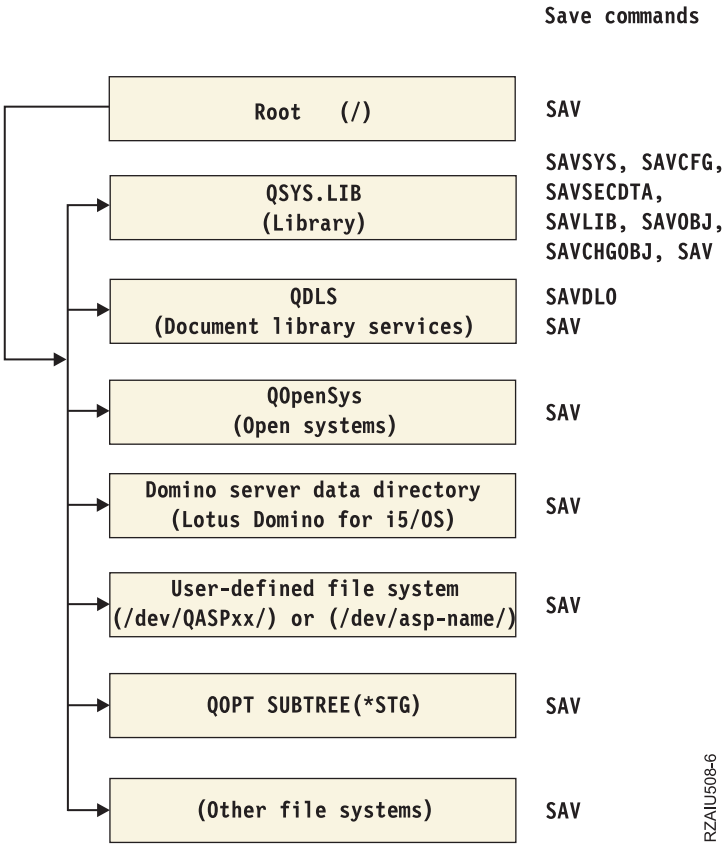


Figure 5. File systems–Save commands

Note: The following file systems are not saveable:

- Network file system (NFS)
- QFileSvr.400

Related tasks

"Performing a complete save using the GO SAVE checklist" on page 34
Use this checklist to perform a complete save operation.

Related information

Lotus Domino reference library

How logical partitioning works

Partitioning the server

Backing up and restoring the HMC

Backup considerations with logical partitions

The process of backing up a logical partition is fundamentally the same as backing up a system without logical partitions. Each logical partition requires its own save strategy.

Here are a few items that should affect how you plan your backup strategy:

- It is important to remember that each logical partition functions independently of any others. Therefore you cannot perform a single, entire system backup. Instead, you need to back up each logical partition separately.
- As part of your backup strategy, remember that a processor failure, main storage failure, failure in the primary partition, or disaster shuts down the entire system. This might require you to recover all or

some of your logical partitions. Therefore, plan carefully how you use your logical partitions and how often you need to perform a backup of each logical partition.

- You can generally perform these backups at the same time since each logical partition functions like an independent system. This can reduce the time that is required for performing backups.
- If any secondary partitions switch a removable media device between themselves, you must back up each of these logical partitions sequentially. You must manually remove and add the removable media device between the logical partitions after each save operation. Use System i Navigatorto change resources for logical partitions.
- The system automatically maintains the configuration data for your logical partitions. This data is not saved to or restored from removable media.
- You should print your system configuration when you make changes to your logical partition configuration.
- Any function that requires you to power off or restart the system (like applying program temporary fixes [PTFs]) requires special care. If you need to power off or restart only a secondary partition, then you might safely do it. However, if you need to power off or restart the primary partition, then you need to power off all the secondary partitions **before** you perform that function.

Related concepts

"Backing up a logical partition"
Each logical partition functions like an independent system, and needs to be backed up individually.

Related information

Backing up and restoring the HMC

Partitioning the server

System i Navigator

Backing up a logical partition

Each logical partition functions like an independent system, and needs to be backed up individually.

You cannot include multiple logical partitions in the same save operation. You must back up each logical partition individually. However, you can perform a backup for each logical partition at the same time (provided all logical partitions have a dedicated removable media device).

The system automatically maintains the configuration data for your logical partitions. You cannot save it to removable media.

You need to make two copies of each backup you perform because you should always store one copy off site in case of a disaster.

It is essential that you have a backup and recovery strategy for each logical partition so that you do not lose any of your important data.

If you have any advanced program-to-program communications (APPC) controls configured that use OptiConnect on the logical partition, vary off these controllers before performing the save operation. If you do not vary off these controllers, they go into a failed status, are marked as damaged, and are not saved.

| You must perform each backup from the console or a workstation that is attached to that logical partition.
| Follow the steps in GO SAVE: Option 21 as you back up each logical partition.

Related concepts

"Backup considerations with logical partitions" on page 103
The process of backing up a logical partition is fundamentally the same as backing up a system without logical partitions. Each logical partition requires its own save strategy.

"Saving configuration data on a logical partition"
Logical partition configuration data is automatically maintained for the life of the physical system. Each logical partition load source contains the configuration data.

Related tasks

"Recommended recovery procedures after eliminating save-outage time" on page 143
If you perform save-while-active operations to eliminate save outage time and you specified *NOCMTBDY for the SAVACTWAIT pending record changes value, you can be left with objects that are saved with partial transactions.

"GO SAVE: Option 21 (saving the entire system)" on page 31
Option 21 saves everything on your system and allows you to perform the save while you are not there.

Related information

Backing up critical HMC data

Backing up and recovering AIX logical partitions that use i5/OS virtual I/O resources

OptiConnect

Planning a backup and recovery strategy

Saving configuration data on a logical partition

Logical partition configuration data is automatically maintained for the life of the physical system. Each logical partition load source contains the configuration data.

Only disaster recovery to a different physical system requires that you rebuild the configuration from the beginning. You should print your system configuration when you make changes to your logical partition configuration. This printout will help you as you rebuild the configuration.

During a save operation, the configuration data for the logical partition is not saved to the media volume. This enables data to be restored to a system even if it has logical partitions. However, you can work with the configuration data for logical partitions as needed for recovery purposes.

Attention: Logical partitions that you keep powered off for extended periods should be restarted at least once after any change to the logical partition configuration. This enables the system to update the changes on that logical partition's load source.

Attention: If you are using the Hardware Management Console (HMC), you must back up the HMC in addition to saving the individual logical partitions.

Related concepts

"Backing up a logical partition" on page 104
Each logical partition functions like an independent system, and needs to be backed up individually.

Related information

Backing up and restoring the HMC

Backing up critical HMC data

Saving data for integrated servers

You can back up and recover integrated server data from i5/OS, the integrated Windows server, the integrated Linux server, and VMWare.

An integrated server is a combination of integrated server hardware, network components, virtual disks, shared devices, and i5/OS integrated server configuration objects.

iSCSI-attached System x™ and blade systems
You can integrate System x or blade systems using System i storage, x86-based hardware, and the Linux, Windows, or VMWare operating systems.

IXS or IXA-attached integrated Windows servers
You can configure i5/OS and the Windows operating system to work with a System i integration with BladeCenter® and System x solution.

If you want to save everything on an AIX, i5/OS, Linux, VMWare, or Windows logical partition, you must use GO SAVE Option 21. This option puts your system into a restricted state and saves the network storage spaces, network server descriptions, objects, and other configuration information for disaster recovery purposes.

Related tasks

"Performing a complete save using the GO SAVE checklist" on page 34
Use this checklist to perform a complete save operation.

Saving data for IXS and IXA-attached integrated Windows servers

You can do a full-system backup of the integrated Windows server to i5/OS, save individual Windows files and directories, or save the network server description, configuration objects, and the associated disk drives.

You can perform any of the following tasks:

- Save your integrated server files to System i tape devices or disks.
- Use the SAV command to back up individual integrated Windows server files or directories.
- Back up the disk drives and the network server description. When you install an integrated server, i5/OS creates a network server description and predefined disk drives for your server that you need to back up. Because Windows server considers them a unified system, you need to save all the disk drives and the network server description to restore correctly.

Related reference

"Methods for saving network server storage spaces" on page 97
This information describes common methods for saving network server storage spaces.

Related information

Backing up and recovering IXS or IXA-attached integrated Windows servers

Backing up the NWSD and other objects associated with integrated Windows servers

Backing up individual integrated Windows server files and directories

Saving data for iSCSI-attached integrated servers

You can do a full-system backup of an iSCSI-attached integrated server to i5/OS, or save the network server description, configuration objects, and the associated disk drives. You also can back up individual files and directories for the integrated Windows server and the integrated Linux server.

You can perform any of the following tasks:

- Save your integrated server files to System i tape devices or disks.
- Back up individual integrated server files or directories using the SAV command.
- When you save the storage space objects that are associated with an integrated server, you also need to save the network server description (NWSD).
- Back up the Network Server Host Adapter (NWSH) object for an iSCSI-attached integrated server.
- Back up Network Server Configuration objects and validation lists for an iSCSI-attached integrated server.
- Back up predefined and user-defined disk drives for an integrated server.
- Save user enrollment information for an integrated server.

Related reference

"Methods for saving network server storage spaces" on page 97
This information describes common methods for saving network server storage spaces.

Related information

Backing up and recovering integrated servers

Backing up the NWSD and other objects associated with integrated servers

Backing up predefined disks for integrated servers

Backing up user-defined disks for integrated servers

Saving individual files on integrated servers

You can use the SAV command to save individual Windows or Linux files and directories on integrated servers. This function is called *file-level backup*.

You must configure the integrated server before you can save individual files and directories to i5/OS.

The Integrated Server Support option enables you to save integrated server data (files, directories, shares, and the Windows registry) to tape, optical or disk (*SAVF) along with other i5/OS™ data and restore the data on an individual basis.

You can do file-level backups for the following configurations:

- iSCSI-attached integrated Windows server.
- IXS-attached integrated Windows server.
- iSCSI-attached integrated Linux server.
- Linux running on a logical partition.

Note: File-level backup is not supported on AIX servers. You need to use third-party solutions to do a file-level backup from AIX servers.

This command saves the MYFILE file in the MYSHARE share from an integrated server named '/QNTC/MYSERVER'.

```
SAV  DEV('/QSYS.LIB/MYLIB.LIB/MYSAVF.FILE')
     OBJ(('/QNTC/MYSERVER/MYSHARE/MYFILE'))
```

Related information

Saving your integrated Windows server files

Backing up individual integrated Windows server files and directories

Backing up and recovering individual integrated Linux server files and directories

Saving Linux data on a logical partition

You can back up and recover a Linux server that is running in a System i logical partition.

The IBM Extended Integrated Server Support licensed product provides support for *file-level backup* on Linux servers. You can use the save and restore commands to save files to System i tape, disk, or optical devices. However, if you want to save selected files on an AIX logical partition, you must use third-party software to do the backup.

You also can back up virtual and directly-attached disks for Linux and i5/OS utilities, and back up network server storage spaces on Linux servers in a System i logical partition.

Related information

Backup options for virtual and directly attached disks

Backing up and recovering individual files and directories for Linux servers running in logical partitions

Saving storage (Licensed Internal Code data and disk unit data)

The save storage process copies the Licensed Internal Code and all of the disk unit data to tape. The media volume that the system produces is a sector-by-sector copy of all permanent data on configured disk units. You cannot restore individual objects from the save tape.

Note: You should use the save and restore storage processes for disaster backup and recovery along with the standard commands for saving and restoring. This procedure is not intended to be used for copying or distributing data to other systems. IBM does not support using the processes for saving and restoring storage as a means to distribute the Licensed Internal Code and the operating system to other system.

Note: The save storage process does not save any of the data from an independent auxiliary storage pool (ASP) that is configured on the system. It only saves the configuration record that indicates that there is an ASP. You must save your independent ASP data separately using the standard commands.

Purpose of saving storage

This information explains several purposes for saving storage.

- The processes for saving and restoring storage provide a one-step method for backing up and recovering the data on an entire system. The restore storage process is an easy and fast method for restoring the data for an entire system.
- The save storage media is for a complete system recovery, and you cannot use it to restore individual objects. You must complement a save storage approach with the SAVSYS, SAVLIB, SAVDLO, and SAV commands.
- To correctly carry out a save storage approach, you should have multiple levels of your backup media.
- The save storage operation does not save disk sectors that are not used or that contain temporary data.

Hardware considerations for saving storage

Learn the limitations of hardware during a save storage procedure.

- If the tape unit supports hardware data compression, then tape unit uses hardware data compression. If the tape unit does not support device data compression, then you might use programming data compression. Generally if the tape unit device operates faster than possible for data compression, the tape unit writes data without compression to the device.
- The system only uses one tape unit.
- The save storage process does not start unless all of the configured disk units are operating.
- The system cannot use some tape units as an alternate IPL device. In these cases, you cannot use these tape units to restore the Licensed Internal Code and the Licensed Internal Code PTFs from the save storage tape.
- The disk configuration of the restoring system must be the same as the disk configuration of the saving system. The disk types and models must be the same or equivalent with some additional devices. Serial numbers and physical addresses do not need to be the same. All disk units that were saved are required for the restore operation.
- A virtual tape device cannot be used.

Operational considerations for saving storage

Before you save storage, take the information in this topic into consideration.

- You can only run the save storage process when the system is in a restricted state.
- The user must have save system (*SAVSYS) special authority to use the Save Storage (SAVSTG) command.

- The SAVSTG command causes the system to power down and starts the system again as though you specified PWRDWNSYS RESTART(*YES). An initial program load (IPL) of the system occurs after the command completes. The save storage function implicitly occurs during the IPL of the system from the dedicated service tools (DST) function.

  Attention logical partitioning users:

  - If you are going to use this command on the primary partition, you must power off all secondary partitions before running the command.
  - To save your entire system configuration, you must save each logical partition individually.
- You can save the first tape without an operator being present. After you save the first tape, DST messages appear that ask for the next tape so the save operation can continue.
- As the amount of storage on the system increases, the chance of an unrecoverable media error increases. Clean the tape unit frequently.
- You must specify a device name on the command. Expiration date (EXPDATE) and clear (CLEAR) parameters are optional. You cannot specify a volume ID.
- The save storage process does not start unless the console is available. If the console is not available, a system reference code appears on the control panel.
- When the save storage operation completes successfully, a normal IPL occurs.

Recovering from save storage errors

If a tape error occurs, the system attempts to recover from the error by automatically trying the operation again.

If the system cannot recover, you must restart the save storage operation on a new tape volume. The operation continues from the last completed tape volume that was saved.

Saving storage for mirrored protection

If the system is using mirrored protection, only one copy of the data from each mirrored pair is saved. When you restore your system by using the SAVSTG tapes, mirrored protection will not be active.

Starting the save storage procedure:

After you complete the prerequisites listed here, you can begin the save storage procedure.

Do these things before you begin:

- Initialize at least three more tapes than you think that you will need to complete the save operation. Initialize them as standard-labeled tapes and specify the maximum density for the tape unit you are using. The number of tapes that you need depends on the size of the system, the number of objects, and the capacity of the tape.

  Each tape should have a volume ID of SAVEDS and an external label that allows you to easily identify the tape. Ensure that each of the tapes support the same density.
- Clean the read/write heads of the tape unit.
- Apply any program temporary fixes (PTFs).
- Print a list of all the PTFs currently on the system. Type the following and press the Enter key:

  ```
  DSPPTF  LICPGM(*ALL) OUTPUT(*PRINT)
  ```
- Ensure that you saved the hardware configuration information from the system. Use the Save Configuration (SAVCFG) command or the Save System (SAVSYS) command to save the configuration objects. The restore storage procedure uses the SAVSYS media volume or the SAVCFG media volume to restore the hardware configuration information.
- Print a list of the current network attributes. Type the following and press the Enter key:

```
DSPNETA  OUTPUT(*PRINT)
```

Keep this Network Attributes list with the tapes that are written during the save storage operation.

Attention logical partitioning users:

- Using the Save Storage (SAVSTG) command causes your system to perform an IPL. If you are running this command on the primary partition, you must *power off* the secondary partitions before continuing.
- In order to save your entire system configuration, you must save each logical partition individually.

1. Sign on at the console with a user profile that has *SAVSYS special authority.
2. Notify users that the system will be unavailable.
3. Change the QSYSOPR message queue to break mode:
   ```
   CHGMSGQ  MSGQ(QSYSOPR) DLVRY(*BREAK) SEV(60)
   ```
4. Type the following to bring the system to a restricted state:
   ```
   ENDSBS  SBS(*ALL) OPTION(*CNTRLD) DELAY(600)
   ```

   Note: For the delay parameter, specify a number of seconds that allows your system time to bring most jobs to a normal end. On a large, busy system, you might need a longer delay.

   The system sends messages to the QSYSOPR message queue. These messages indicate that the subsystems ended, and the system is in a restricted state. When the subsystems have ended, continue with the next step.
5. Load the first media volume of the SAVSTG media, and make the media device ready.
6. Check the control panel on your processor to ensure that the system is in normal mode.

   Note: You can access the control panel information through the control panel on the system, through the Hardware Management Console (HMC) or through system service tools (SST) on the primary partition.
7. If you are not using logical partitioning, continue with the next step. Otherwise, if you are performing this operation from the primary partition, be sure to power down all secondary partitions.
8. Enter the save storage command, such as:
   ```
   SAVSTG  DEV(TAP01) CLEAR(*ALL)
   ```
   You can also enter an expiration date (EXPDATE(*mmddyy*)).
9. Press the Enter key. The system will power down with a restart IPL. This is similar to PWRDWNSYS OPTION(*IMMED) RESTART(*YES). This means that when you enter the command, the system will power down and perform an automatic IPL.

   When the IPL occurs, a dedicated service tools (DST) function starts saving storage. If the operator correctly loads the media volume and the expiration date check passes, the operator does not need to be present for the first media volume.

   If you load the media volume correctly, the following save status display continually displays the progress of the save operation.

```
                          Function Status

                    You selected to save storage.



                            1 % Complete
```

The *Percent saved* field on the display estimates the progress of the total amount of saved sectors. However, this estimate does not accurately predict the time it takes to save or the number of tapes that you need to complete the save operation. The reason is that the system does not save unused sectors.

Responding to messages:

While the SAVSTG procedure is running, you might see either the Handle Tape or Diskette Intervention display or the Device Intervention Required display.

```
                    Handle Tape or Diskette Intervention

Device:

I/O manager code . . . . . . . . . . . . . :      ________

Type choice, press Enter.

  Action  . . . . . . . . . . . . . . . . . . . . 1=Cancel
                                                  ____________________________
                                                  3=Continue
                                                  ____________________________

F3=Exit            F12=Cancel
End of tape encountered.  Load next volume.
```

```
                        Device Intervention Required

Device type. . . . . . . . . . . . . . . . . . . . :      _____

I/O manager code . . . . . . . . . . . . . . . . . :      ________

Type choice, press enter

  Action  . . . . . . . . . . . . . . . . . . . . . . 1=Cancel
                                                      2=Ignore
                                                      3=Continue
                                                      4=Format
```

When one of these displays appears, look for messages at the bottom of the display or for an I/O manager code on the display. Respond to the display by using the following information:

Table 37. Handling SAVSTG Messages

Message or Code	Your Action
End of tape encountered. Load next volume.	Load the next tape volume. Select option 3 (Continue), and press the Enter key.
Active files exist on media.	To continue the save operation to tape, select option 2 (Ignore) to ignore the active files. Press the Enter key.
Tape unit not ready.	Make the tape unit ready, select option 3 (Continue), and press the Enter key.
Media is write protected.	Replace the tape with a tape that is not write-protected and select option 3 (Retry). Press the Enter key.
Device is not able to process the media format.	Select option 4 (Format), and press the Enter key.
Tape or diskette loaded is blank.	Select option 4 (Format), and press the Enter key.
I/O manager code 8000 0001C.	Replace the tape with a tape that can be formatted to the requested density and select option 3 (Retry). Press the Enter key.

If an unrecoverable tape media error occurs, do the following:

1. Remove the tape that failed from the tape device. Do not put the tape that failed with the other tapes that you already used during the save storage operation. You cannot use the failed tape during the restore storage operation.
2. Load a different tape in the media device.
3. Press the F3 key to return to the Use Dedicated Service Tools menu.
4. Go to "Resuming a save storage operation."

Completing the SAVSTG process:

When the last tape is complete and no errors have occurred, the tape automatically rewinds and a normal IPL occurs. You will then need to follow some specific steps to complete the process.

Do the following:

1. The system updates the data area QSAVSTG in library QSYS to show the date and time of the save operation. Use the Display Object Description (DSPOBJD) command to display the date and time of the save storage operation.
2. Ensure that the save operation completed successfully. Use the Display Log (DSPLOG) command to display the history (QHST) log:

   ```
   DSPLOG QHST
   ```

   Or use the Display Message (DSPMSG) command to display the QSYSOPR messages:

   ```
   DSPMSG QSYSOPR
   ```

   Look for a save storage completion message or diagnostic messages that indicate that the system could not read some sectors. If the system found any damaged sectors that it could not read, this means that your tapes might not be complete. If you use them to restore storage, the operation might fail. Contact your service representative for assistance. Then repeat the save storage operation.

This completes the save storage procedure. If you do not want the system to perform an automatic IPL, you can use an autostart job, which powers down the system.

Canceling a save storage operation

To cancel the save storage operation, press the F19 key. This action cancels an active save storage operation.

Resuming a save storage operation

Your system must meet these prerequisites before you can resume a save storage operation.

You can use this procedure only if the following conditions are true:

- The save storage operation finished saving the Licensed Internal Code.
- The save storage operation completed writing to at least one tape during the save storage operation.
- You attached all disk units, and the disk units are operating.

If an error occurs that stops a save storage operation (for example, system power loss, operator error, or tape drive error), you can start the save storage operation again.

Do the following to resume the save storage operation:

1. Select manual mode on the control panel of your processor.
2. Power on the system by using the Power switch or the Power button. The IPL or Install the System menu is shown.
3. Select option 3 (Use Dedicated Service Tools (DST)) and press the Enter key.
4. Sign on DST by using the password that is assigned to your system for full DST authority. The Use Dedicated Service Tools (DST) menu that appears on the console.

5. From the Use Dedicated Service Tools (DST) menu, select option 9 (Work with save storage and restore storage) and press the Enter key.
6. Select option 4 (Resume save storage) and press the Enter key.

   If the system does not allow you to resume the save storage operation, a display with an explanation appears on the console.
7. If you see the Resume Save Storage display on the console, load the tape that the system last wrote to when the save storage operation stopped. Press the Enter key.
8. If the volume identifier of the tape that is loaded is different from the volume identifier of the first save storage tape, the Device Intervention Required display appears. The message at the bottom says that the **Wrong volume loaded**.

   To continue the save operation, type SAVEDS on the "New volume" line and select option 4 to format the tape.

Save-while-active function

The save-while-active function allows you to use your system during all or part of the save process, that is, save your system while it is active.

You can use the save-while-active function, along with your other backup and recovery procedures, to reduce or eliminate your outage for particular save operations. The amount of time during the backup process that you cannot use the system is the **save-outage time**. The save-while-active function allows you to use your system during all or part of the save process, that is, save your system while it is active. This allows you to reduce or eliminate your save-outage time. In contrast, other save functions allow no access, or only allow read access, to the objects as you are saving them.

Related reference

"Methods for saving security data" on page 63
Use any of these methods for saving security data.

"Methods for saving configuration objects in QSYS" on page 64
Use any of these methods for saving configuration objects in QSYS.

"Methods for saving i5/OS optional libraries (QHLPSYS, QUSRTOOL)" on page 65
Use any of these methods for saving i5/OS optional libraries.

"Saving system information" on page 59
Use the Save system information (SAVSYSINF) command to perform a partial save of the data saved by the Save system (SAVSYS) command.

Related information

SAVLICPGM

Save-while-active concepts

The save-while-active function is an option on several i5/OS save commands. It allows you to save parts of your system without putting your system in a restricted state.

You can use the save-while-active function to reduce your save outage or to eliminate your save outage.

How it works

i5/OS objects consist of units of storage, which are called **pages**. The system maintains two copies of the pages of the object that change while you are performing the save operation:

- The first image contains the updates to the object with which normal system activity works.
- The second image is an image of the object at a single point in time. The save-while-active job uses this image to save the object to the media.

In other words, when an application makes changes to an object during a save-while-active job, the system uses one image of the object's pages to make the changes. At the same time, the system uses the other image to save the object to the media. The image that the system saves does not have the changes you made during the save-while-active job. The image on the media is as it existed when the system reached a checkpoint.

Checkpoints

The **checkpoint** for an object is the instant in time that the system creates an image of that object. The image that the system creates at that instant in time is the **checkpoint image** of the object.

Creating a checkpoint image is similar to taking a photograph of a moving automobile. The point in time that you took the photograph equates to the checkpoint. The photograph of the moving automobile equates to the checkpoint image. When the system has finished making the checkpoint image of the object, the object has reached a checkpoint.

Despite the name save-while-active, you cannot change objects while the system obtains their checkpoint images. The system allocates (or locks) objects as it obtains checkpoint images. After the system obtains the checkpoint images, you can change the objects.

Synchronization

When you save more than one object, you must choose when the objects will reach a checkpoint in relationship to each other. This is *synchronization*. The following are the three kinds of synchronization:

Full synchronization
With full synchronization, the checkpoints for all of the objects occur at the same time. The checkpoints occur during a time period in which no changes can occur to the objects. IBM strongly recommends that you use full synchronization, even when you are saving objects in only one library. You also have the option of synchronizing data that is saved from multiple save operations.

Library synchronization
With library synchronization, the checkpoints for all of the objects in a library occur at the same time.

System-defined synchronization
With system-defined synchronization, the system decides when the checkpoints for the objects occur. The checkpoints for the objects might occur at different times resulting in complex restore procedures.

Save-outage time

The amount of time during the backup process that you cannot use the system is the **save-outage time**. You can use the save-while-active function to **reduce** or **eliminate** your save outage.

The easiest and recommended way to use the save-while-active function is to **reduce** your save-outage time. You can reduce your save-outage time by ending your applications that change objects. You can restart the applications after the system has reached a checkpoint for those objects. You can choose to have the save-while-active function send a notification when it completes the checkpoint processing. After the save-while-active function completes checkpoint processing, it is safe to start your applications again. When you use the save-while-active function in this way, the save-outage time can be much less than with normal save operations.

You can also use the save-while-active function to **eliminate** your save-outage time. When you use the save-while-active function to eliminate your save-outage time, you do not end the applications that make changes to the objects you save. However, the save operation affects the performance and response time

of your applications. You should also use journaling or commitment control for all of the objects you are saving when using save-while-active in this way. Using the save-while-active function to eliminate your save-outage time might also greatly increase the complexity of your recovery procedures.

Save-while-active commands

The save-while-active function is an option on the i5/OS save commands listed below:

Command	Function
SAVLIB	Save Library
SAVOBJ	Save Object
SAVCHGOBJ	Save Changed Objects
SAVDLO	Save Document Library Objects
SAV	Save
SAVRSTLIB	Save/Restore Library
SAVRSTOBJ	Save/Restore Object
SAVRSTCHG	Save/Restore Changed Objects
SAVRSTDLO	Save/Restore Document Library Objects
SAVRST	Save/Restore

Related concepts

"Save-while-active restrictions" on page 123
The following restrictions apply to all of the commands which provide the save-while-active function.

Checkpoint processing with save-while-active

Checkpoint processing occurs after the system determines exactly which objects it will save for a particular library. If the save-while-active request is for multiple libraries, then the system performs checkpoint processing for all libraries in the save request.

Checkpoint processing does not require that the system maintain two complete copies of the objects you are saving. The system only maintains two copies of the pages of the object that the applications are changing while you are performing the save operation. The more pages that applications change for an object during the save-while-active request, the greater the storage requirement for the object. After the system completes checkpoint processing to create the checkpoint image of the page, performance decreases slightly for the first update to a page. The performance impact varies depending on the disk type, available disk storage, and processor model. Further updates to the same changed page do not require any additional processing with respect to the checkpoint version of the page.

The following figure shows how the system maintains a checkpoint image of an object during a save-while-active operation. The shaded parts of the diagram represent the checkpoint version of the object. An explanation of the steps follows the figure.

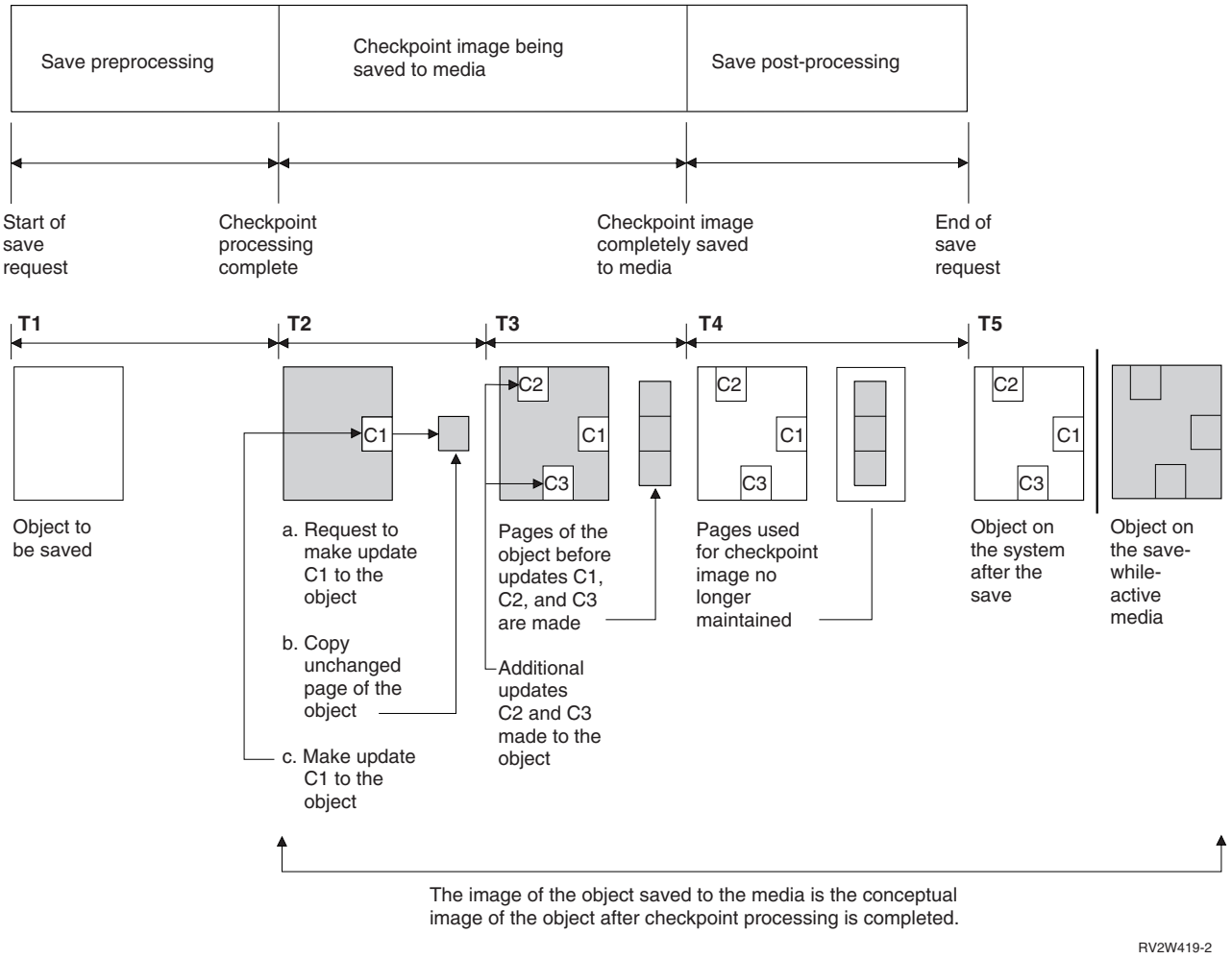


Figure 6. System management of updates to objects after checkpoint processing is complete

The figure above shows a timeline with T1 — T5:

1. Time T1 is the save preprocessing phase of the save-while-active operation. The object reaches a checkpoint at the end of time T1.
2. Time T2 shows an update to the object, referred to as C1. The update occurs while the save-while-active request saves the object to the media.
   a. An application makes a request to update C1.
   b. The system first makes a copy of the original page.
   c. The applications make the change to the object.

   The original page copied is then part of the checkpoint image for the object.
3. Time T3 shows that the object received two additional changes, C2 and C3. Any additional change requests that are made to the pages of the object already changed for C1, C2, or C3 do not require any additional processing. At the end of time T3, the save-while-active request has completely saved the object to the media.
4. Time T4 shows that the system no longer maintains copied pages for the checkpoint image of the object because the system no longer needs them.
5. Time T5 shows the object on the system has the C1, C2, and C3 changes. But the copy, or image, of the object saved to the media does not contain those changes.

Related concepts

"Commitment control with save-while-active" on page 118
This information applies if you are using commitment control and save-while-active to eliminate your save-outage time. This information applies only if you are not specifiying *NOCMTBDY for handling pending record changes on the SAVACTWAIT parameter.

Timestamp processing with save-while-active

The save-active-time for an object can be useful when you determine which recovery procedures to use after you restore objects from the media.

All of the changes made to the object before the save active timestamp will be present for the object on the save-while-active media. The changes made to the object after the save active timestamp will not be present for the object on the save-while-active media.

If you specify UPDHST(*YES) on the save command, the system records the date and time that it performs a save operation for an object. The system takes the timestamp early during the save preprocessing phase. The timestamp identifies when the save operation started for the object. This timestamp is the **save-time** for the object. Multiple objects that you save with one save request will have the same save time if they all reside in the same library. This timestamp displays in the **save date/time** field when you use the Display Object Description (DSPOBJD) command displays.

The save-while-active function introduces an additional timestamp that relates to save processing. This additional timestamp is the save-active-time for an object. The **save-active-time** identifies the time an object that you saved with the save-while-active function object reached the checkpoint. The save-active-time is the same for all of the objects that reach a checkpoint together.

When you use the Display Object Description (DSPOBJD) command, the save-active-time displays in the **save active date/time** field. The system only updates the save-active-time for an object if you specify UPDHST(*YES) on the save command when you request the save-while-active operation.

Some objects do not require special save-while-active checkpoint processing. Therefore the save-while-active timestamp is the same time that the object's description is saved. Examples of this are object types *JOBQ and *OUTQ that have only their descriptions saved, not their contents. This is also true for files that do not have any members.

For physical file members, the **last save date/time** information that the DSPFD command identifies is either the last save-time or the last save-active-time. The information that displays depends on which type of save operation you last performed for each of the members.

The recovery considerations do not apply if you are using the save-while-active function to reduce your save-outage time.

Related tasks

"Recommended recovery procedures after eliminating save-outage time" on page 143
If you perform save-while-active operations to eliminate save outage time and you specified *NOCMTBDY for the SAVACTWAIT pending record changes value, you can be left with objects that are saved with partial transactions.

Related information

Journal management

Recovery procedure considerations: This consideration applies to journaled objects and libraries that are saved with the save-while-active function. The start of the save journal entry contains both the save-time and save-active-time. The object saved journal entry in the journal also contains both the save-time and save-active-time. Look for the journal entry that identifies when the journaled file member reached the checkpoint. All journal entries after this journal entry for a journaled object are reflected in the data that

is saved during a save-while-active operation. This information might be useful when you determine what recovery procedures are necessary after restoring journaled objects from the save-while-active media.

Commitment control with save-while-active

This information applies if you are using commitment control and save-while-active to eliminate your save-outage time. This information applies only if you are not specifiying *NOCMTBDY for handling pending record changes on the SAVACTWAIT parameter.

If an object receives updates under commitment control during the checkpoint processing phase of a save-while-active operation, the system saves the object at a commitment boundary. The system saves all objects that reach a checkpoint together at the same common commitment boundary.

During the save preprocessing phase of a save-while-active request, the system ensures that it saves the objects commitment boundary as follows:

- If the job performing the save-while-active request is not currently at a commitment boundary, the save request ends without saving any objects. This processing is the same for any save request.
- If updates are in progress for any objects in a group that are reaching a checkpoint together, the system delays the checkpoint. The checkpoint resumes when all of the transactions reach a commitment boundary. The system waits the amount of time specified on the second element of SAVACTWAIT parameter for these transactions to reach a commitment boundary. If uncommitted transactions still exist when the specified time expires, the save request ends.
- The system identifies which jobs have commitment definitions that are not currently at a commitment boundary and are delaying the checkpoint processing. The system waits until uncommitted transactions delay checkpoint processing for a group of objects for approximately 30 seconds. The system then sends a CPI8365 message to the QSYSOPR message queue for each job that is delaying the save-while-active request. After you receive these messages, you can then take the appropriate actions to bring all commitment definitions for those jobs to a commitment boundary.
- When no more commitment definitions are delaying the save-while-active job, the save-while-active job completes the checkpoint processing for the objects. After the checkpoint processing ends, the system enables changes for those objects under commitment control.
- If a commitment definition has uncommitted changes, it could possibly delay a save-while-active request. The uncommitted changes could delay the save-while-active request even though the changes are not for any database files. This situation can occur if you are journaling any of the database files to the same journal as the commitment definition is using for unrelated, uncommitted changes and if you specify a value greater than 0 for the second element of the SAVACTWAIT parameter.
- If an application is performing a read-for-update operation but no changes have been made, the application is considered to have started a commit cycle. The system enables a checkpoint to be established in the middle of a commit cycle as long as no changes have been made. Checkpoint processing does not stop if the application is performing only a read-for-update operation.
- The system temporarily delays a job that has all commitment definitions at a commitment boundary when both of the following are true:
  - When it is likely that an application will change an object that is under commitment control
  - When that object is reaching a checkpoint

  The system holds that job until the objects reach a checkpoint, or the checkpoint processing for the object exceeds the time specified on the SAVACTWAIT parameter. During the time the system delays a job at a commitment boundary, the Work Active Job (WRKACTJOB) command displays **CMTW** as the job status.

Related concepts

"Performance considerations for save-while-active" on page 121
While you can run save-while-active operations any time, save-while-active operations will affect the performance of other applications you are running.

Related tasks

"Checkpoint processing with save-while-active" on page 115
Checkpoint processing occurs after the system determines exactly which objects it will save for a particular library. If the save-while-active request is for multiple libraries, then the system performs checkpoint processing for all libraries in the save request.

Commitment control with save-while-active and *NOCMTBDY: This information applies if you are using commitment control and save-while-active to eliminate your save-outage time. This information applies only if you specified *NOCMTBDY for handling pending record changes on the SAVACTWAIT parameter.

- If the job performing the save-while-active request is not currently at a commitment boundary, the save continues and objects are saved with partial transactions.
- If updates other than pending record changes are in progress for any objects in a group that are reaching a checkpoint together, the system delays the checkpoint. The checkpoint resumes when all of the transactions reach a commitment boundary. The system waits the amount of time specified on the third element of SAVACTWAIT parameter for these transactions to reach a commitment boundary. If uncommitted transactions still exist when the specified time expires, the save request ends.

Using save-while-active to synchronize the saved data

To fully synchronize the checkpointed data for multiple save operations, use the Start Save Synchronization (STRSAVSYNC) command to specify the number of save operations that you want to synchronize. Then start each save operation, specifying full synchronization for each operation. Multiple save operations are run concurrently in different jobs.

You can synchronize any combination of Save Object (SAV), Save Library (SAVLIB), Save Object (SAVOBJ), or Save Changed Object (SAVCHGOBJ) commands. For example, you might specify the following commands: STRSAVSYNC, SAV, and SAVLIB. To synchronize multiple save operations for directories and libraries, complete the following steps:

1. Issue the STRSAVSYNC command to start the save-while-active action.
2. Issue the Save Object (SAV) command for each session to save the directories. The system responds by displaying message CPI373F, `Waiting for all SYNCID &1 operations to start`. The system is waiting for the other save operations to start.
3. Issue the Save Library (SAVLIB) commands for each session to save the libraries. The system responds by displaying message CPI373F, `Waiting for all SYNCID &1 operations to start`.
4. The system processes the checkpoints for each of the save operations.

   The system responds with a `Checkpoint reached` message to the save-while-active message queue (SAVACTMSGQ). You might also receive checkpoint progress messages before reaching the `Checkpoint reached` message.

   The system saves the data for each of the specified directories and libraries.

   The system issues a completion message for each of the specified save operations.

You can also synchronize multiple save operations in batch mode, or synchronize save operations for two different libraries that are journaled to the same journal.

Example: Save-while-active cross-file-system synchronization in batch mode

This example shows a way to synchronize the checkpointed data when saving a library and directory. The STRSAVSYNC command starts a synchronized checkpoint named SYNCMYDATA for two save-while-active operations. The two participating save operations are submitted for batch processing. The first save-while-active operation saves the MYLIB library to the TAP01 device, and the second one saves the MYDIR directory to the TAP02 device. Each participating save operation specifies the SAVACT value for full synchronization of the data that it saves, and the synchronization ID, SYNCMYDATA.

```
STRSAVSYNC  SYNCID(SYNCMYDATA)  NUMSYNC(2)
SBMJOB      CMD(SAVLIB  LIB(MYLIB)  DEV(TAP01)
            SAVACT(*SYNCLIB)  SYNCID(SYNCMYDATA))
SBMJOB      CMD(SAV  DEV('/QSYS.LIB/TAP02.DEVD')  OBJ(('/MYDIR'))
            SAVACT(*SYNC)  SYNCID(SYNCMYDATA))
```

The following is a list of the errors that you might encounter during the save-while-active processing.

Error message	Cause	Action
CPF37BC, Synchronization ID &1 ended. Wait time exceeded.	One or more save jobs fail to start within the wait time (STRSAVWAIT) specified on the STRSAVSYNC command.	Reissue the STRSAVSYNC command using a larger wait time value (STRSAVWAIT) and the save commands.
CPF37BB, Synchronization ID &1 already started	STRSAVSYNC specifies an existing synchronization ID.	Reissue the STRSAVSYNC command using a unique synchronization ID (SYNCID).
CPF37BE, Synchronization ID &1 not started	The system attempts to synchronize the save jobs but the user did not issue the STRSAVSYNC command.	Issue the STRSAVSYNC command, and then issue the SAV or SAVLIB commands.
CPF37B9, Synchronization ID &1 in use	The system attempted to start more save job operations with the same SYNCID value.	Reissue the STRSAVSYNC command using the correct number of save jobs to synchronize (NUMSYNC). For example, if you need five save jobs, but specified only 4 in the STRSAVSYNC command, you need to do the following: 1. End the other four save jobs that already started. 2. Prepare the media. 3. Reissue the STRSAVSYNC command. 4. Reissue the save commands.

Related concepts

"Full synchronization" on page 133
All objects you are saving reach a checkpoint at the same time. The system then saves them to the media. IBM strongly recommends that you use full synchronization, even when you are saving objects in only one library.

Related information

Start Save Synchronization (STRSAVSYNC)

Save Library (SAVLIB)

Save Object (SAV)

Using save-while-active with network server storage spaces

You can use the save-while-active function to save the network server storage spaces while keeping the integrated server online. The save-while-active function reduces or eliminates your outage for save operations.

Network server storage spaces can be linked to the following network server descriptions: *IXSVR, *ISCSI, and *GUEST NWSD. You can use the save-while-active function when the disk is linked to *ISCSI and *GUEST NWSD, but not when the disk is linked to *IXSVR.

Saving a storage space while it is active

| This command saves the objects that are associated with storage space '/QFPNWSSTG/MYDISK'. The
| associated network server description (*NWSD) is varied on.

```
| SAV   DEV('/QSYS.LIB/MYLIB.LIB/MYSAVF.FILE')
|       OBJ(('/QFPNWSSTG/MYDISK'))
|       SAVACT(*YES) SAVACTOPT(*NWSSTG)
```

| If you do not specify the SAVACT (save active) and SAVACTOPT (save active option) parameters in the
| SAV command, you must shut down the integrated server to save the storage space.

| **Related concepts**

| "Additional save-while-active option (SAVACTOPT) parameter" on page 135
| The SAV command provides additional save-while-active options which you specify on the
| SAVACTOPT parameter. The default is *NONE, which means that no additional options are used
| during a save-while-active operation.

| **Related reference**

| "Methods for saving network server storage spaces" on page 97
| This information describes common methods for saving network server storage spaces.

| **Related information**

| Backing up storage spaces for an active Linux server in a logical partition

| Backing up and recovering integrated Linux servers

Considerations and restrictions for the save-while-active function

The save-while-active function affects important aspects of your system such as performance, auxiliary storage, and commitment control. The pages that follow contain considerations and restrictions in regard to these aspects of your system.

The pages that apply to you depend on whether you are reducing or eliminating your save-outage time.

Use dynamic device allocation to allocate tape drives more efficiently.

Performance considerations for save-while-active

While you can run save-while-active operations any time, save-while-active operations will affect the performance of other applications you are running.

You should run save-while active operations during times of low system activity. A few interactive jobs or batch jobs that are primarily read-only, are examples of activities that allow better system performance during the save-while-active operation.

In general, the system performs checkpoint processing faster for a small number of larger objects than for a large number of smaller objects.

You should not use the save-while-active function when the system is very busy or when there is very little disk storage available. Before you save large amounts of data (such as all user libraries), you should initially use the save-while-active function on a limited amount of data. Using the save-while-active feature on a limited amount of data will help you determine its impact on your system's performance and storage.

Related concepts

"Save-while-active restrictions" on page 123
The following restrictions apply to all of the commands which provide the save-while-active function.

"Commitment control with save-while-active" on page 118
This information applies if you are using commitment control and save-while-active to eliminate your save-outage time. This information applies only if you are not specifiying *NOCMTBDY for handling pending record changes on the SAVACTWAIT parameter.

Central processing unit (CPU) and save-while-active:

The relationship between the system's CPU and a save-while-active operation depends on the available CPU capacity and the characteristics of other jobs on the system

Available CPU capacity: The amount of CPU capacity that is available for the save process can have a large influence on the time required for the save operation to complete. Therefore, be prepared for the save-while-active operation to take longer than a save operation on a restricted system. The change in the time required for the save operation to complete might be as little as 10 percent longer to four to five times longer or more. This depends on the system resources that are available for the save operation. As a guideline, allow only about 30% of the CPU for workloads that are running in the background.

Characteristics of other jobs on the system: The active jobs during a save-while-active operation can affect both the response time and the duration of the save operation. Try to use the save-while-active function when CPU utilization is low and the amount of update activity on the system is low.

Auxiliary storage activity and save-while-active:

When choosing the time period for a save-while-active operation, evaluate the activity in auxiliary storage without save-while-active processing.

Ideally, disks should be less than 30 percent busy before adding the activity for the save operation. This is due to the heavy auxiliary storage activity that is added with the save-while-active operation.

Main storage (memory) and save-while active:

How a save-while-active operation affects main storage depends on three items,

- Pageable size of the machine pool
- Job priority and pool usage
- Number and size of objects

Pageable size of the machine pool: Additional pages are required in the machine pool for the system to use during the save-while-active operation. Additionally, saving many small objects or file members places additional requirements on the pageable portion of the machine pool. You should consider the addition of 1200KB to the machine pool a minimum. Additional memory might improve the response time and the save-time.

Additional megabytes of storage for the machine pool might help performance if saving thousands of small objects or file members (less than 50KB object sizes). You should monitor the machine pool for paging activity.

Job priority and pool usage: You must decide which jobs have priority: the save operation or the other activity on the system. You should give the save operation a lower priority than the interactive jobs, but a higher priority than other batch jobs. This priority will maintain the best response time for interactive jobs, but still allow the save to complete as quickly as possible. In addition, separate the save operation from other work on your system by using a separate memory pool. The size of this separate pool should be a minimum of 10MB (16MB if you are using a high speed tape device). The full synchronization and library synchronization options generally require a few additional megabytes of memory. If there are thousands of objects or file members in the save-while-active operation, you should add more memory to the memory pool. This is especially true if the objects are small. To determine the correct pool size for your system, monitor the paging activity in the pool during a save and adjust the memory as necessary. However, if the pool is a shared memory pool, then the settings in the system value, QPFRADJ, will adjust its performance.

Number and size of objects: If you are saving many small objects or file members, the paging in the machine pool might increase. You should monitor paging in the machine pool. You should take steps to minimize paging to maintain better overall system performance. These recommendations are also apply for normal save and restore operations.

DLO activity and save-while-active:

If the save-while-active operation is run at a time when users are updating document library objects (DLO), the save-while-active process might affect these users.

When users are changing document library objects, they might notice a delay if the save-while-active operation is performing checkpoint processing for the document library objects.

For example, an application might be editing a document while a save-while-active operation is running. It is possible that the application could attempt to update the document when the save-while-active operation is performing checkpoint processing on that document. If that happens, the application will probably wait until checkpoint processing completes before it can make the update. If the save-while-active job is running at low priority, or on a busy system, the application might wait for an extended time.

If the save-while-active operation does not complete checkpoint processing for the document library objects within 30 minutes, the user function ends abnormally. The abnormal end of the user function indicates there is a problem. The system administrator should determine why the save-while-active process is taking an excessive amount of time for the document library objects to reach a checkpoint. Then, the system administrator should take the appropriate action to correct the problem. This might require contacting your service representative.

Storage considerations for save-while-active

The save-while-active function uses more disk storage than normal save operations.

As applications change the objects in a save-while-active operation, the system makes copies of the data that reach a checkpoint. The system could run out of available storage if the following happens:

- The data on your system uses a high percentage of the disk capacity.
- A large amount of the data changes during a save-while-active operation.

If the system sends messages that it is running out of storage, you should be prepared to stop the save operation or some applications.

The full synchronization option uses the most additional storage. The system-defined synchronization option uses the least additional storage.

Related concepts

"Save-while-active restrictions"
The following restrictions apply to all of the commands which provide the save-while-active function.

Save-while-active restrictions

The following restrictions apply to all of the commands which provide the save-while-active function.

- The save-while-active function is only available on the commands listed in the Save-while-active function.
- You cannot use the save-while-active function in the following situations:
  - When all subsystems have ended. If you have ended all subsystems, the save operation is the only user job that is active. It must finish before you can restart your subsystems and applications. The following save operations require that you end all subsystems. Therefore, you cannot use the save-while-active function with these operations:
    - Saving the system library

- Saving all libraries
- Saving the entire system

– When freeing or deleting storage during a save operation. If specifying STG(*FREE) or STG(*DELETE) on a save command, or CHKFORMRK(*YES) on the SAVDLO command, you cannot use the save-while-active function.

- You should not use the save-while-active function when the system is very busy or when there is very little disk storage available. Before you save large amounts of data (such as all user libraries), you should initially use the save-while-active function on a limited amount of data. Using the save-while-active feature on a limited amount of data will help you determine its impact on your system's performance and storage.
- You should not load, apply, or remove program temporary fixes (PTF)s when running a save-while-active operation.
- You must issue separate save commands to use the save-while-active function for objects in libraries, document library objects, and objects in directories. If you need to synchronize objects you are saving with different commands, first end your applications until all of the objects have reached a checkpoint.
  – If you have only one media device, each command must finish before the next can start. If you use the save-while-active function to reduce your save-outage time, save folders and directories first. Save libraries last. Saving the objects in this order will probably provide the greatest reduction in the save-outage time.
  – If you have multiple media devices, and you use the save-while-active function to reduce your save-outage time, save libraries, folders, and directories concurrently. This will probably provide the greatest reduction in you save-outage time.
- You cannot save objects that you create after the save operation begins.
- You cannot save objects that other jobs are using during checkpoint processing.
- Do not use System Service Tools (SST) functions for objects you are currently saving by a save-while-active operation.

Related concepts

"Save-while-active concepts" on page 113
The save-while-active function is an option on several i5/OS save commands. It allows you to save parts of your system without putting your system in a restricted state.

"Performance considerations for save-while-active" on page 121
While you can run save-while-active operations any time, save-while-active operations will affect the performance of other applications you are running.

"Storage considerations for save-while-active" on page 123
The save-while-active function uses more disk storage than normal save operations.

Related reference

"Save-while-active object locking rules" on page 125
The object locking rules that the system uses for save-while-active requests are less restrictive than the rules it uses for other save operations.

Library restrictions:

- Full synchronization is not available when you use save all IBM libraries using SAVLIB LIB(*IBM).
- If you have specified *NOCMTBDY for the SAVACTWAIT parameter, you cannot save any *IBM library or any library that begins with Q (except for QGPL).

Integrated file system restrictions: Consider the following when using the save-while-active function with the SAV or SAVRST commands with integrated file systems:

- The wait time option is not available.
- When you are saving objects in libraries or document library objects, the considerations stated for those objects also apply.

Document library restrictions: Consider the following considerations when you use the save-while-active function to save document library objects.

- Full synchronization is not available. Only system-defined synchronization is available.
- Checkpoint notification is not available. This means that you cannot determine when it might be safe to restart your applications that use document library objects. When saving document library objects, the benefit of the save-while-active function is that objects are allocated for a shorter time than with normal save operations.
- You might cannot save documents during save-while-active processing if a reclaim operation (RCLDLO command) is running.
- Folders might not be saved during save-while-active processing if a reorganize operation (RGZDLO command) or a reclaim operation (RCLDLO command) is running.
- Some applications use application programming interfaces (APIs) or shared folders to work with a document like a personal computer. When they update document data, they save the updates to a temporary file. The application does not permanently write changes to the document until the application session ends. Therefore these applications can update a document while a save-while-active operation is running.

  Other applications update documents directly as the application receives data. For example, some spreadsheet applications and image applications work this way. If this type of application updates a document while a save-while-active operation is running, the application does not save document. The job log receives Diagnostic messages CPF8A80:**Document in use** and CPF90AC:**Document not saved** to indicate that the application did not save the object because the object was in use.

Save-while-active object locking rules

The object locking rules that the system uses for save-while-active requests are less restrictive than the rules it uses for other save operations.

These object locking rules allow users to perform update operations and use most object-level commands after the system performs checkpoint processing. Generally, the system keeps a shared, no update (*SHRNUP) lock on the objects through the checkpoint processing. After the checkpoint completes, the system unlocks most of the objects. Other objects remain allocated with a shared for read (*SHRRD) lock.

The following table shows the locks a normal save operation holds, by a save-while-active operation during checkpoint processing, and by a save-while-active operation after checkpoint processing is complete.

Table 38. Lock type needed for save operation

		Save-While-Active	
Object Type	**SAVACT(*NO)**	**Establish Checkpoint**	**After Checkpoint**
Most object types	*SHRNUP	*SHRNUP	None
Configuration object	None	[1]	[1]
Data area	*SHRNUP	*SHRRD	None
Database members	*SHRNUP	*SHRRD	None
Document	*SHRNUP	*SHRRD	None
Folder	*SHRRD	*SHRRD	None
Job queue	*SHRRD	*SHRRD	None
Journal	*SHRRD	*SHRRD	None
Journal receiver	*SHRRD	*SHRRD	*SHRRD
Library, when the library or an object in it is being saved	*SHRUPD	*SHRUPD	*SHRRD
Output queue	*SHRRD	*SHRRD	None
Product load	*SHRNUP	*SHRNUP	*SHRRD
Spooled file	*EXCL	*EXCL	[5]
System resource management object	*SHRNUP	[1]	[1]

Table 38. Lock type needed for save operation (continued)

		Save-While-Active	
Object Type	**SAVACT(*NO)**	**Establish Checkpoint**	**After Checkpoint**
User profiles, authorization lists, and authority holders	*SHRRD	[1]	[1]
Object, if STG(*FREE) is specified	*EXCL[2]	[1]	[1]
Objects in directories	Share with readers	Share with readers[3, 4]	Share with readers and writers[3]

[1] The save-while-active function is not available when saving these objects.

[2] Applies to document, file, journal receiver, module, program, SQL package, and service program. Other types remain as listed previously.

[3] Objects in QNTC are not synchronized with SAVACT(*SYNC). Furthermore, all locks for these file systems will be released before the checkpoint message is sent.

[4] Objects that are saved with SAVACTOPT(*ALWCKPWRT) and have the QP0L_ATTR_ALWCKPWRT system attribute set, have an implied *share with readers and writers* lock.

[5] A lock is held that prevents another save action against the spooled file. All other spooled file actions, such as displaying, copying, deleting, and printing, are allowed.

These locking rules pertain to object-level locks and not database record-level locks. The locking rules allow the opening and closing of database file members and any record-level I/O operations to database file members during any phase of the save-while-active operation.

Related concepts

"Save-while-active restrictions" on page 123
The following restrictions apply to all of the commands which provide the save-while-active function.

Object locking: During save-while-active checkpoint processing:

During checkpoint processing, these locking rules can conflict with object-level lock types of exclusive allow read (*EXCLRD); exclusive, no read (*EXCL); and share update (*SHRUPD).

Some object-level system commands and user applications can acquire these lock types. User applications that acquire these object-level locks generally conflict with save-while-active operations until the checkpoint processing is complete for the objects. User applications that use system commands that require these object-level locks also conflict with save-while-active operations until the checkpoint processing is complete for the objects. Lock conflicts can prevent the save operation from saving the object. Lock conflicts can also can prevent applications from using the object. To eliminate lock conflicts during checkpoint processing, you should end your applications until checkpoint processing is complete.

If you are saving spooled files with SPLFDTA(*ALL) specified, quiesce your spooling writers until checkpoint processing is complete. To quiesce the spooling writers, hold the output queues of each spooling writer or end the spooling writer.

In general, checkpoint processing operations prevent the following list of operations from occurring for objects you are saving.

- Changing an object
- Deleting an object
- Renaming an object
- Moving an object to a different library or folder
- Changing the ownership of an object
- Compressing or decompressing an object

Object locking: After save-while-active checkpoint processing:

After completing checkpoint processing, an attempt to perform one of the operations that are listed in this topic will result in a message stating that the library is in use.

- Performing additional save or restore operations on objects or libraries being saved
- Deleting, renaming, or reclaiming a library from which objects are being saving.
- Loading, applying, removing, or installing PTFs that affect a library from which objects are saved
- Saving, restoring, installing, or deleting licensed programs that contain a library from objects you are saving

In addition, the following object types have operations that are restricted after checkpoint processing is complete. An attempt to perform one of the operations that are listed below the following objects below will result in a message stating that the object is in use:

**FILE-PF (physical file):*

- Using the Change Physical File (CHGPF) command with the parameter specifications of SRCFILE, ACCPTHSIZ, NODGRP, or PTNKEY to change a physical file.
- Using an SQL Alter Table statement to change a physical file.

**JRN (journal):*

- Deleting a journal with an associated journal receiver.
- Using the Work with Journal (WRKJRN) interface to recover a journal that has an associated journal receiver you are saving.

**JRNRCV (journal receiver):*

- Deleting or moving the journal receiver.
- Deleting the journal with which the receiver is associated.
- Using the Work with Journal (WRKJRN) interface to recover a damaged journal receiver.

**PRDLOD (product load):*

- Deleting, moving, or renaming the product load.

Restrictions for commitment control with save-while-active

Restrictions for commitment control with save-while-active consist of object-level resource restrictions and application programming interface (API) resource restrictions.

Related information

Commitment Control

Object-level resource restrictions: You cannot make object-level resource changes for objects under commitment control that are in the object-level resource library while the system performs checkpoint processing for those objects. You cannot make object-level resource changes if either of the following are true:

- The commitment definition is at a commitment boundary.
- Only record-level changes have been made in the uncommitted transaction.

For this situation, the change does not occur until the save-while-active request completes checkpoint processing for the library. After a delay of approximately 60 seconds, you receive inquiry message CPA8351. The inquiry message allows you to continue to wait for the checkpoint processing to complete or to cancel the request for the object-level resource. If the job is a batch job, the QSYSOPR message queue receives inquiry message CPA8351.

Application programming interface (API) resource restrictions: You can register an API resource within a commitment control transaction with the QTNADDCR API. If you set the **Allow save while active** field to `Y` when you use this API, the considerations in this topic do not apply.

You cannot place resources under commitment control if the system is performing checkpoint processing for any save-while-active request and either of the following are true:

- With the Add Commitment Resource API (QTNADDCR program), the commitment definition is at a commitment boundary.
- Only record-level changes have been made in the uncommitted transaction.

For this situation, the add is delayed until checkpoint processing is complete for the save-while-active request. After a delay of approximately 60 seconds, you receive inquiry message CPA8351. The inquiry message allows you to continue to wait for the checkpoint processing to complete or to cancel the request for the API resource. If the job is a batch job, the QSYSOPR message queue receives the inquiry message CPA8351.

If a commitment definition has an API commitment resource associated with it, and checkpoint processing is being performed for any save-while-active request, then the job performing a commit or rollback operation for the commitment definition is delayed immediately after the commit or rollback has been performed. The system delays the job until the completion of checkpoint processing for the save-while-active request. After the checkpoint processing is complete, control is returned back to the job issuing the commit or rollback. This delay is necessary because a commitment definition with an API commitment resource is only considered to be at a commitment boundary immediately after a commit or rollback operation but before control is returned to the user program. Once the commit or rollback operation returns control back to the user program, the commitment definition is no longer considered to be at a commitment boundary.

Saving to multiple devices to reduce your save window

Use these save methods to decrease your save window by saving to multiple devices.

Setting up saves to multiple devices

You can reduce your save window by using multiple devices. When you save to multiple devices you can use one of two techniques. You can issue a single save operation as one job, or you can issue multiple save operations as several jobs.

Single save operation

Save (or restore) operations identify a media file by the device (DEV), sequence number (SEQNBR), volume identifiers (VOL), and file label (LABEL) parameters. These parameters only allow one media file to be identified. However, a parallel save (or restore) operation uses more than one media file. You can solve this problem by using a media definition.

A media definition (*MEDDFN) allows you to identify more than one media file. A media definition defines the devices, sequence numbers, and volume identifiers that the parallel save operation will use.

A media definition also allows you to specify whether to save the data in parallel or serial format and whether to use dynamic device allocation.

You create a media definition by using the Create Media Definition (QsrCreateMediaDefinition (ILE) or QSRCRTMD (OPM)) API.

Multiple save operation

When you issue multiple save operations to save different sets of data to different media devices, you perform *concurrent* save operations. The following scenarios provide some examples of situations when you might want to perform concurrent saves within the integrated file system.

- Save the complete integrated file system structure and all user libraries concurrently:

```
SAV  DEV('/QSYS.LIB/TAP01.DEVD') OBJ(('/*') ('/QSYS.LIB' *OMIT) ('/QDLS' *OMIT))
SAVLIB  LIB(*ALLUSR) DEV(TAP02)
```

- Save separate unmounted user-defined file systems concurrently:

```
SAV  DEV('/QSYS.LIB/TAP01.DEVD') OBJ(('/dev/udfs-directory/udfs-01.udfs')
SAV  DEV('/QSYS.LIB/TAP02.DEVD') OBJ(('/dev/udfs-directory/udfs-02.udfs')
```

Saving libraries to multiple devices for a single save operation:

You can perform a save operation while using more than one media device simultaneously.

A traditional save to a single device produces one or more tape files on the tape media. A media file is produced for each saved library. When data is saved to multiple devices in a single operation, the data can be saved in parallel format. The data in each media file is spread across each device. Each device might contain pieces of each saved object. When saving multiple libraries to multiple devices in a single operation, the data can also be saved in serial format. The data for each media file is entirely written to one device. Each device contains entire libraries.

You can perform a save operation while using more than one media device simultaneously. If you save a single library, the data that is produced on the save media will have a *parallel* save format. The data is spread across the media devices. If you use Backup, Recovery, and Media Services (BRMS), the save format is also parallel.

If you save multiple libraries to more than one media device, the system saves each library to a single device in *serial* format. If you use BRMS to save multiple libraries to more than one media device, the format could be a mixture of parallel and serial formats.

Notes: The following shows when the system will use a parallel or serial save operation. You can specify the save format in a media definition.

1. This table shows the default format.
2. For BRMS, you can specify the format in the Parallel Type field in a control group.
3. For the save commands, you must use a media definition (*MEDDFN) that you create with the Create Media Definition (QSRCRTMD) API. You can specify the format when you create the media definition.
4. You cannot save *ALLUSR, *IBM, OR *NONSYS libraries in parallel format.

Table 39. Library parallel and serial saves

Save scenario	Using SAVLIB, SAVOBJ command	Using BRMS
Save one library to multiple devices	Parallel	Parallel
Save multiple libraries to multiple devices	Serial	Could be a mixture of parallel and serial

This table shows the correlation between the libraries being saved and some possible results of the media files that are produced.

Table 40. Libraries saved

Data saved	Number of devices	Format	Tape media files produced
Library A	1	Serial	A

Table 40. Libraries saved (continued)

Data saved	Number of devices	Format	Tape media files produced
Library A	2	Parallel	Device 1: A Device 2: A
Libraries A, B, C, D	1	Serial	A, B, C, D
Libraries A, B, C, D	2	Parallel	Device 1: A, B, C, D Device 2: A, B, C, D
Libraries A, B, C, D	2	Serial	Device 1: A, C Device 2: B, D

Once you create a media definition, a convenient way to save all of your user libraries to multiple devices is to specify SAVLIB LIB(*ALLUSR) DEV(*MEDDFN). If you happen to have a particularly large library that you do not want to save in serial format, you could omit that library and save it individually in parallel format.

BRMS provides an easy to use interface that allows you to perform parallel save operations without creating a media definition. You specify which tape devices to use in parallel, and BRMS builds and manages the media definition for you.

Related information

Create Media Definition API

Saving the integrated file system using multiple devices for a single save operation:

A traditional save to a single device produces one tape file on the tape media. You can perform a save operation while using more than one media devices simultaneously.

Integrated file system data saved by a single SAV command using multiple devices will be in a *parallel* save format. The data is spread across the media devices. If you use Backup, Recovery, and Media Services (BRMS), the save format is also parallel.

Note: Using a media definition to save your integrated file system data to a single device specified in a *MEDDFN is the same as specifying that device on the SAV command. It is not beneficial to use a *MEDDFN when saving to a single device. The data is saved in serial format.

Table 41. Integrated file system parallel saves

Save scenario	Using SAV command	Using BRMS
Save integrated file systems to multiple devices	Parallel	Parallel

This table shows the correlation between the integrated file system being saved and the name of the media files being produced.

Table 42. Integrated file system saved

Data saved	Number of devices	Format	Tape media files produced
Integrated file system data	1	Serial	SAVdatetime
Integrated file system data	2	Parallel	Device 1: SAVdatetime Device 2: SAVdatetime

Once you create a media definition, a convenient way to save the entire integrated file system to multiple devices is to specify SAV DEV ('/QSYS.LIB/Y.LIB/X.meddfn') OBJ (('/*') ('/QSYS.LIB' *OMIT) ('/QDLS' *OMIT).

BRMS provides an easy-to-use interface that allows you to perform parallel save operations without creating a media definition. You specify which tape devices to use in parallel, and BRMS builds and manages the media definition for you.

1.

   Note: Performing a parallel save with large objects might improve performance. However, if saving small objects, the performance might decrease.

2.

   Note: Restoring individual objects from a parallel save might take a substantial amount of time.

Dynamic device allocation:

Dynamic device allocation enables you to allocate tape devices as they are needed.

You can allocate your tape devices in any of the following ways:

- All tape devices needed for the save operation are allocated in the beginning.
- Only one tape device is allocated at the beginning of a save operation. The maximum number of devices are allocated when data is ready to be written.
- The number of devices specified for the minimum parallel device resources field is allocated at the beginning of a save operation. Additional devices are allocated when data is ready to be written.

Note: Use the Create Media Definition API to specify your preferred value.

Dynamic tape allocation restrictions

- Initially all of the save operations will continue to allocate at least one device. Any operation that does not use a media definition will allocate its device at the beginning of the operation.
- Devices will not be dynamically deallocated.
- The dynamically allocated devices will be limited to these points in time
  - After a save-while-active checkpoint.
  - When the initial library data is ready to be written to an available device.

Restrictions for saving to multiple devices:

The devices that you specify in a media definition must be compatible standalone tape devices or tape media library devices.

The tape volumes that you specify must have compatible media formats.

Note: Your results might depend on the device type that you use. This is because different device types might identify different formats for the same media. For example, one 8 mm device might identify a tape as having an FMT7GB format, while a different 8 mm device might identify the same tape as having an FMT5GB format.

You might use a media definition on the following commands and APIs:

Name	API[1]	Command[2]
Save Library		SAVLIB
Save Object	QSRSAVO	SAVOBJ
Save	QsrSave	SAV
Save Changed Object		SAVCHGOBJ
Restore Library		RSTLIB
Restore Object (Library)		RSTOBJ

Name	API[1]	Command[2]
Restore Object Integrated File System	QsrRestore	RST
Create Media Definition	QsrCreateMediaDefinition QSRCRTMD	
Delete Media Definition	QsrDeleteMediaDefinition QSRDLTMD	DLTMEDDFN
Retrieve Media Definition	QsrRetrieveMediaDefinition QSRRTVMD	

[1] For more information regarding these APIs, refer to the API finder.

[2] For more information regarding these CL commands, refer to the CL command finder.

You must have *USE authority to the media definition, *EXECUTE authority to the media definition library, and normal save or restore authority for each device you specify in the media definition.

You cannot use a media definition if the save or restore command or API specifies any of the following:

- Volume identifiers
- A sequence number
- A save file
- An optical file

You cannot use a media definition if your system has been enabled for CD-ROM premastering by using the Generate CD-ROM Premastering Information (QLPCDINF, QlpGenCdPremasteringInfo) API.

Parallel format and media definition usage restrictions

- SAVLIB LIB(*ALLUSR, *IBM, *NONSYS) cannot save data in parallel format. You will receive this error message if you specify a media definition in parallel format.
- A media definition cannot be used to restore a list of libraries or generic libraries.

Related information

API finder

System CL Command reference

BRMS

Create Media Definition (QsrCreateMediaDefinition (ILE) or QSRCRTMD (OPM))

Parameters for the save-while-active function

Use these options to specify how you will use the save-while-active function.

Related tasks

"Recommended procedure for reducing your save-outage time" on page 138
Use this general procedure to reduce your outage for particular save operations.

Synchronization-level values for Save Active (SAVACT) parameter

This table shows which synchronization levels are available for each command and the value to specify for each level.

Table 43. SAVACT parameter values

Command	Full Synchronization	Library Synchronization	System-Defined Synchronization
SAVLIB SAVOBJ SAVCHGOBJ	*SYNCLIB	*LIB	*SYSDFN
SAVRSTLIB SAVRSTOBJ SAVRSTCHG	not available	*LIB	*SYSDFN
SAVDLO SAVRSTDLO	not available	not available	*YES
SAV SAVRST	*SYNC	not available	*YES

Full synchronization:

All objects you are saving reach a checkpoint at the same time. The system then saves them to the media. IBM strongly recommends that you use full synchronization, even when you are saving objects in only one library.

Full synchronization usually completes checkpoint processing in the least amount of time, and it has the least impact to your recovery procedures. Because it allocates all objects you are saving before obtaining a checkpoint image of them, it usually keep objects locked longer than other options. This option also uses the most additional storage.

To request full synchronization of the data saved within an operation, specify *SYNCLIB for the SAVACT parameter when saving library data, or specify *SYNC when saving directory data.

To request full synchronization of the data saved within multiple save operations, use the STRSAVSYNC command before starting the save operations. Because the backup data is synchronized, all of the data is saved at a single point in time and represents a consistent view of all the data. You can synchronize data from libraries and directories, a set of save library operations, or a set of save directory operations. Because there is only one synchronization checkpoint, it improves system performance for the save-while-active function.

Related tasks

"Using save-while-active to synchronize the saved data" on page 119
To fully synchronize the checkpointed data for multiple save operations, use the Start Save Synchronization (STRSAVSYNC) command to specify the number of save operations that you want to synchronize. Then start each save operation, specifying full synchronization for each operation. Multiple save operations are run concurrently in different jobs.

Related information

Start Save Synchronization (STRSAVSYNC)

Library synchronization:

All objects in a library reach a checkpoint at the same time. But different libraries reach checkpoints at different times. This option might be useful if all of the following are true.

- You are saving more than one library.
- Each of your applications is dependent on only one library.

- Full synchronization uses more storage than you have available or it will keep objects locked longer than your business needs will allow.

  Related concepts

  "Using the precheck option" on page 4
  Use the precheck option explains how to have the system check certain criteria on each object that you save on a library-by-library basis. This option is not required.

System-defined synchronization:

Using this option could cause lengthy recovery procedures. You should only use this option for objects that you are protecting with journaling or commitment control to avoid extremely complex recovery procedures.

Objects you are saving might reach checkpoints at different times. This option will usually keep objects locked for the shortest period of time and use the least amount of additional storage. But it will usually take the longest to complete checkpoint processing. It will also result in the most complex recovery procedures if you do not end your applications during the checkpoint processing. When you save objects in libraries, *SYSDFN operates the same as *LIB.

The wait time (SAVACTWAIT) parameter

The SAVACTWAIT parameter specifies the amount of time to wait for an object that is in use, or for transactions with pending changes to reach a commit boundary, before continuing the save operation.

You can specify three wait time elements in the SAVACTWAIT parameter.

Related tasks

"Monitoring your save-while-active operation" on page 141
Do the following procedures as they apply if you are using the save-while-active function to eliminate your save-outage time.

Object locks: The default value is 120 seconds. You can specify the amount of time to wait for the object to become available. You can specify any number of seconds from 0 to 99999 for object locks, or *NOMAX to have the save-while-active operation wait indefinitely. If you end your applications before starting the save operation, specify 0 seconds. If you do not end your applications, specify a value large enough for your applications to make the objects available.

If an object is not available during checkpoint processing, the save-while-active operation will wait up to the specified number of seconds for the object to become available. While waiting for an object, the save operation does nothing else. The save operation might need to wait for several objects. The total time that the save-while-active operation waits might be much longer than the value specified. If an object does not become available within the specified time, the object is not saved, but the save operation continues.

Pending record changes: The default value is *LOCKWAIT. You can specify any number of seconds from 0 to 99999 for transactions with pending record changes. You use *NOCMTBDY to save objects without waiting for commit boundaries. If you use *NOMAX, the save-while-active operation will wait indefinitely. If 0 is specified, all objects being saved must be at commit boundaries.

After the save-while-active operation allocates a group of objects that it is synchronizing, it might wait a specified number of seconds for all jobs that are using the same journals as these objects to reach commitment boundaries. If these jobs do not reach commitment boundaries within the specified time, the save operation ends. If you specify a value greater than 30, the system, after waiting 30 seconds, sends a CPI8365 message to the QSYSOPR message queue for each job for which the save-while-active operation is waiting.

Other pending changes: The default value is *LOCKWAIT. You can specify the amount of time to wait for transactions with Data Definition Language (DDL) object changes or any API commitment resource

that is added without the option to allow normal save processing. If you use *NOMAX there is no maximum wait time. You can specify any number of seconds from 0 to 99999. If 0 is specified, and only one name is specified for the Objects (OBJ) parameter, and *FILE is the only value specified for the Object types (OBJTYPE) parameter, the system will save the object without requiring the types of transactions that are listed above to reach a commit boundary.

The checkpoint notification (SAVACTMSGQ) parameter

This information contains a table that shows the messages that are sent for each ommand when the check point processing is complete.

You can specify the checkpoint notification on the SAVACTMSGQ parameter. The specified message queue receives a message after checkpoint processing is complete. An operator or a job can monitor this message queue and restart applications when checkpoint processing is complete.

Table 44. SAVACTMSGQ checkpoint completion messages

Command	Full Synchronization	Library Synchronization	System-Defined Synchronization	Save Operation Abnormal Termination
SAVLIB SAVOBJ SAVCHGOBJ	CPI3712[1]	CPI3710 for each library	CPI3710 for each library	CPI3711
SAVRSTLIB SAVRSTOBJ SAVRSTCHG	not available	CPI3710 for each library	CPI3710 for each library	CPI3711
SAV objects in libraries	CPI3712[1]	not available	CPI3710 for each library	CPI3711
SAVDLO SAVRSTDLO SAV objects in folders	not available	not available	not available	not available
SAV objects in directories SAVRST	CPI3712	not available	CPI3712	CPI3722
Note: [1] Prior to the CPI3712 checkpoint completion message, messages CPI3724 and CPI3725 are sent to the message queue and to the workstation to indicate the progress of the checkpoint processing. CPI3724 is sent for each library as the operation begins to allocate the objects in that library. CPI3725 is sent when all objects have been allocated as the operation begins to obtain the checkpoint images of the objects.				

Related tasks

"Recommended procedure for reducing your save-outage time" on page 138
Use this general procedure to reduce your outage for particular save operations.

Additional save-while-active option (SAVACTOPT) parameter

The SAV command provides additional save-while-active options which you specify on the SAVACTOPT parameter. The default is *NONE, which means that no additional options are used during a save-while-active operation.

Applications should only use the allow checkpoint write (*ALWCKPWRT) option to save objects which are associated with the application. Also, the applications should have additional backup and recovery considerations such as Lotus Domino databases.

Objects with the QP0L_ATTR_ALWCKPWRT server attribute set are locked with O_SHARE_RDWR by the save operation. You can update data before the save-while-active operation reaches a checkpoint.

You will need to verify these objects after you restore them. You might also need to perform additional recovery procedures before they are usable.

You also can use the SAVACTOPT(*NWSSTG) command to specify additional options for saving network server storage spaces.

Related tasks

"Using save-while-active with network server storage spaces" on page 120
You can use the save-while-active function to save the network server storage spaces while keeping the integrated server online. The save-while-active function reduces or eliminates your outage for save operations.

Save-while-active and your backup and recovery strategy

How your save-while-active function fits into your backup and recovery strategy depends on whether you will reduce or eliminate your save-outage time. These pages contain information to help you decide how you will use the save-while-active function. It also contains pages with technical descriptions of the save-while-active function.

How the save-while-active function fits into your backup and recovery strategy depends on if you plan to reduce or eliminate your save-outage time.

Reducing your save-outage time

Reducing your save-outage time is the easiest way to use the save-while-active function. When you use this option, the restore procedure is the same as when you perform a standard save operation. In addition, you can use the save-while-active function to reduce your save-outage time without using journaling or commitment control. Unless you have no tolerance for a save-outage time, you should use the save-while-active function to reduce your save outage.

Eliminating your save-outage time

You can use the save-while-active function to eliminate your save outage. Use this option only if you have no tolerance for a save-outage time. You should use the save-while-active function to eliminate your save-outage time only for objects that you protect with journaling or commitment control. In addition you will have considerably more complex recovery procedures. You should consider these more complex recovery procedures in your disaster recovery plan.

Making your decision

This topic might help you decide how the save-while-active function fits into your backup and recovery plan. Review your applications. Other procedures that you use in your backup and recovery strategy still apply. You should still consider them when you review your backup and recovery procedures. You might conclude one of the following:

- Your current save strategy is adequate for your scheduled save-outage time.
- Critical application libraries are candidates for save-while-active processing.
- Your critical application libraries are candidates, but might require modification to minimize recovery procedures.
- Critical documents or folders are candidates.
- All application libraries are candidates because of a compressed save-outage time.
- You will use save-while-active to reduce your save-outage time because you can tolerate a small save outage time.
- You will use save-while-active to eliminate your save-outage time for the following reasons:
  - You have no tolerance for a save-outage time.
  - You are already using journaling and commitment control.

– You plan to use journaling and commitment control.

The following pages might help you make an informed decision on how to use the save-while-active function.

Reducing save-outage time: Overview

This information tells you what happens when you use the save-while-active function to reduce your save-outage time.

Reducing your save-outage time is the recommended way to use the save-while-active function. To reduce your save-outage time, you can end the applications that make changes to the objects you are saving. You can restart the applications when the system has established a checkpoint for application-dependent objects.

An application-dependent object is any object that applications use and update. By using the save-while-active to reduce your save-outage time, you will need to perform no additional recovery procedures when you restore the objects.

You can specify to have the system send you a message when it has completed checkpoint processing of the following:

- For all objects within a particular library
- For all libraries in the save request

You can start the applications again when all application-dependent objects have reached a checkpoint. The checkpoint images of the objects that you save then appear as if you performed a dedicated save during the time the applications were ended.

If you are saving objects from multiple libraries and a common application-dependency that spans the libraries exists, do not restart the applications right away. You should wait until checkpoint processing has completed for all the libraries in the save request. When the checkpoint processing has completed for all the libraries, you can then restart the applications.

This method can substantially reduce your save-outage time, even though it does not eliminate it.

Related concepts

"Reducing your save-outage time" on page 138
Use the save-while-active function to reduce your save-outage time. This is the easiest way to use the save-while-active function.

Eliminating save-outage time: Overview

This information tells you what happens when you use the save-while-active function to eliminate your save-outage time.

The save-while-active function can eliminate your outage for particular save operations by not waiting for applications to end before starting the save procedure. However, you will have more complex and longer recovery procedures after restoring objects from the media.

You will have more complex recovery procedures because eliminating your save-outage time saves objects at different application boundaries. For save-while-active purposes, an **application boundary** is a point in time:

- When all of the objects that a particular application is dependent on are at a consistent state in relationship to each other.
- When the objects are also in a state where you can start or restart the application.

When you choose to eliminate your save-outage time, applications can update the objects you are saving before the objects reach a checkpoint. When this happens the system cannot determine if the images of

those objects reached application boundaries when you restore those objects. Therefore at restore time, you need to define recovery procedures to bring those objects to a common application boundary. You will need these recovery procedures to bring the objects to a consistent state in relationship to each other. For this reason you should protect the objects you are saving with journaling or commitment control.

Furthermore, if you do not use commitment control, partial transactions can be saved without your knowledge. When you use commitment control, you can choose to have the save operation save all objects at transaction boundaries. However, if applications do not reach commitment boundaries within the specified time, the save operation will fail.

You should consider each of the following when you determine these recovery procedures:

- If the objects that the applications are dependent on consist entirely of database files or if they depend on other object types such as integrated file system objects.
- If the objects that the applications are dependent on are in a single library or span multiple libraries.
- If the objects that the applications are dependent on are journaled objects.
- If the changes the applications made to the objects are under commitment control.

Related concepts

"Considerations for recovery procedures after eliminating save-outage time" on page 142
This topic discusses some of the considerations for save-while-active recovery procedures. In general, the system cannot preserve application boundaries because they are defined by the application. It is left up to you to provide for any of the appropriate recovery procedures when you use the save-while-active function to eliminate your save-outage time.

"Example: Restoring libraries after reducing save-outage time" on page 140
This example shows a typical restore procedure after you reduce save-outage time in a library. Your exact use of the function might differ, based on your specific application requirements.

Related tasks

"Recommended recovery procedures after eliminating save-outage time" on page 143
If you perform save-while-active operations to eliminate save outage time and you specified *NOCMTBDY for the SAVACTWAIT pending record changes value, you can be left with objects that are saved with partial transactions.

Related reference

"Eliminating your save-outage time" on page 141
Use the save-while-active function to eliminate your save-outage time.

Reducing your save-outage time

Use the save-while-active function to reduce your save-outage time. This is the easiest way to use the save-while-active function.

Use the following general procedures to reduce your save-outage time for particular save operations. You need to end the applications for the objects you are saving before you perform these procedures. However, these procedures require **no additional recovery** procedures.

Related concepts

"Reducing save-outage time: Overview" on page 137
This information tells you what happens when you use the save-while-active function to reduce your save-outage time.

Recommended procedure for reducing your save-outage time

Use this general procedure to reduce your outage for particular save operations.

This procedure is the recommended way to use the save-while-active function on a daily basis. This save-while-active operations saves the objects as if they were saved in a dedicated fashion. This procedure does not require any special recovery procedures.

1. End all application jobs that are making updates to the application-dependent objects.
2. Start the save-while-active operation for the objects that reside in the application libraries. Specify a message queue on which to receive the checkpoint completion message.
3. Wait for the checkpoint completion or termination message identified in SAVACTMSGQ checkpoint completion messages at the message queue you specified on the SAVACTMSGQ parameter.
4. Start the application jobs again.
5. For journaled objects in the save request, if you did not save their receivers in the request, save those receivers after the save request finishes.

Related reference

"Parameters for the save-while-active function" on page 132
Use these options to specify how you will use the save-while-active function.

"The checkpoint notification (SAVACTMSGQ) parameter" on page 135
This information contains a table that shows the messages that are sent for each ommand when the check point processing is complete.

Example: Reducing save-outage time for two libraries

This example makes use of two libraries, LIB1 and LIB2. Both libraries contain objects that you will save on a daily basis.

Your current save strategy ends jobs that make changes to the objects in the two libraries for the entire time that the you are saving the libraries.

For this example, objects of any type can exist in the two libraries. The objects that exist in the two libraries might or might not be journaled.

The several hour save-outage time can be greatly reduced by the following steps:

1. End all application jobs that are making updates to the objects in libraries LIB1 and LIB2.
2. Submit the following command as an individual batch job:

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

   Note: You could also use the SAVOBJ or SAVCHGOBJ commands depending on your specific needs.

   The objects in library LIB1 and LIB2 reach a checkpoint together, as specified by SAVACT(*SYNCLIB), and the system saves the libraries to TAP01. The system sends the message indicating that checkpoint processing is complete to QSYSOPR.

   You are also saving access paths for the logical files, as specified by ACCPTH(*YES). If you specify this, the access paths, in most cases, will not need to be built after restoring the files from this save media.

   A single save command saves the libraries to provide a consistent checkpoint. This is also faster than saving both libraries to the same storage device with separate commands. Using two save commands to two separate media devices allows the system to perform the checkpoint processing for the libraries concurrently. It might also allow the system to perform checkpoint processing faster than saving both libraries with a single save command.
3. After checkpoint processing is complete, the message queue QSYSOPR receives the message CPI3712. If checkpoint processing does not complete for the objects, message queue receives the message CPI3711 and the save operation ends.
4. After receiving CPI3712 message, start the application jobs that make updates to the objects in the two libraries.

The objects exist on the media as they were at the time the application jobs were ended, prior to the save command being run. However, the save-while-active function greatly reduces the amount of time that the applications are not available.

Example: Reducing save-outage time for a directory

This example uses a directory, MyDirectory. The directory contains objects that you will save on a daily basis. Your current save strategy ends jobs that make changes to the objects in the directory for the entire time that the you are saving the directory.

The objects that exist in the directory might or might not be journaled.

The several hour save-outage time can be greatly reduced by the following steps:

1. End all application jobs that are making updates to the objects in MyDirectory.
2. Submit the following command as an individual batch job:

```
SAV  DEV('/QSYS.LIB/TAP01.DEVD')
     OBJ('/MyDirectory') SAVACT(*SYNC)
     SAVACTMSGQ('QSYS.LIB/LIB1.LIB/MSGQ1.MSGQ')
```

   The objects in directory MyDirectory reach a checkpoint together, as specified by SAVACT(*SYNC). The system saves the objects TAP01. The system sends the message indicating that checkpoint processing is complete to MSGQ1
3. After checkpoint processing is complete, the message queue receives the message CPI3712. If checkpoint processing does not complete for the objects, message queue receives the message CPI3722 and the save operation ends.
4. After receiving CPI3712 message, start the application jobs that make updates to the objects in the directory.

The objects exist on the media as they were at the time the application jobs were ended, prior to the save command being run. The save-while-active function greatly reduces the amount of time that the applications are not available.

Example: Restoring libraries after reducing save-outage time

This example shows a typical restore procedure after you reduce save-outage time in a library. Your exact use of the function might differ, based on your specific application requirements.

You can restore the objects from the media just as if you did not use the save-while-active function. The restore requires no additional recovery procedures. You can restore the two libraries with the following commands:

```
RSTLIB SAVLIB(LIB1) DEV(TAP01)

RSTLIB SAVLIB(LIB2) DEV(TAP01)
```

Related concepts

"Eliminating save-outage time: Overview" on page 137
This information tells you what happens when you use the save-while-active function to eliminate your save-outage time.

"Considerations for recovery procedures after eliminating save-outage time" on page 142
This topic discusses some of the considerations for save-while-active recovery procedures. In general, the system cannot preserve application boundaries because they are defined by the application. It is left up to you to provide for any of the appropriate recovery procedures when you use the save-while-active function to eliminate your save-outage time.

Related tasks

"Recommended recovery procedures after eliminating save-outage time" on page 143
If you perform save-while-active operations to eliminate save outage time and you specified *NOCMTBDY for the SAVACTWAIT pending record changes value, you can be left with objects that are saved with partial transactions.

Example: Restoring a directory after reducing save-outage time

This example shows a typical restore procedure after you reduce save-outage time in a directory. Your exact use of the function might differ, based on your specific application requirements.

You can restore the objects from the media just as if you did not use the save-while-active function. The restore requires no additional recovery procedures. You can restore the directory with the following command:

```
RST  DEV('/QSYS.LIB/TAP01.DEVD') +
  OBJ('/MyDirectory')
```

Eliminating your save-outage time

Use the save-while-active function to eliminate your save-outage time.

Use the following general procedures to eliminate your save-outage time for particular save operations. These save-while-active procedures do not require any applications to be ended to perform the save operation. However, these save-while-active methods do require **additional recovery** procedures.

IBM highly recommends that you use these procedures only for objects you are protecting with journaling or commitment control.

Related concepts

"Eliminating save-outage time: Overview" on page 137
This information tells you what happens when you use the save-while-active function to eliminate your save-outage time.

Recommended procedure for eliminating save-outage time

This procedure outlines how you can use the save-while-active function to eliminate save-outage time. You will not end the application jobs.

1. Start the save-while-active operation for the objects. You can do this specifying (SAVACT(*SYNCLIB)) for libraries or (SAVACT(*SYNC)) for directories on the save command.
2. When you receive the message CPI3712 (for SAVACT(*SYNCLIB)) or CPI3710 (for SAVACT (*SYNC)), no additional lock conflicts for objects or jobs with uncommitted transactions occur.
3. If checkpoint processing does not complete for the objects you are saving, the message queue specified for the SAVACTMSGQ parameter receives the message CPI3712 or message CPI3712 and the save operation ends.
4. Objects with a lock conflict still allow checkpoint processing to complete, and the save operation continues. However, the system does not save objects with a lock conflict.
5. The save-while-active operation ends.
6. For every journaled object in the save-while-active request, save each attached journal receiver that the save-while-active operation did not save.

Monitoring your save-while-active operation

Do the following procedures as they apply if you are using the save-while-active function to eliminate your save-outage time.

Related concepts

"The wait time (SAVACTWAIT) parameter" on page 134
The SAVACTWAIT parameter specifies the amount of time to wait for an object that is in use, or for transactions with pending changes to reach a commit boundary, before continuing the save operation.

Checking for lock conflicts:

1. During checkpoint processing, look for possible lock conflicts by monitoring the save-while-active job.

   A status of LCKW on the Work Active Jobs (WRKACTJOB) display identifies a lock conflict.
2. If a lock conflict exists for a particular object, identify the job that holds the conflicting lock with the Work with Object Locks (WRKOBJLCK) command.
3. Take appropriate steps to have the job release the lock so that the save-while-active job can continue and perform the save for that particular object.

4. If a save-while-active request does not save a particular objects due to lock conflicts, resolve all lock conflicts.
5. Issue the **entire** save-while-active request again. You should not just re-save the objects that had a lock conflict. Otherwise objects you saved in the two save-while-active requests will not be in a consistent state each other. This situation can lead to a complex recovery procedure.

Monitoring save-while-active operations for objects under commitment control:

1. During checkpoint processing, if changes to the objects you are saving are made under commitment control and *NOCMTBDY is not used for the SAVACTWAIT pending record changes value, monitor the QSYSOPR message queue for CPI8365 messages.

   CPI8365 messages indicate that the jobs have commitment definitions that are preventing the save-while-active job from proceeding. The QSYSOPR message queue only receives CPI8365 informational messages if you specify the SAVACTWAIT time to be at least 30 seconds.

   Note: See or information about controlling the amount of time that elapses while waiting for commitment definitions to reach a commitment boundary.
2. Take the appropriate steps, as outlined in the recovery portion of the CPI8365 message, to bring all commitment definitions for a job to a commitment boundary.
3. The save-while-active request ends if you cannot reach a commitment boundary for a particular commitment definition.
4. Depending on the type of uncommitted changes one of the following happens:
   - The job log receives CPF836C messages.
   - The QSYSOPR message queue receives CPI8367 messages.

   In either case, the messages contain the job names that had commitment definitions that prevented the save-while-active request for the library.

Considerations for recovery procedures after eliminating save-outage time

This topic discusses some of the considerations for save-while-active recovery procedures. In general, the system cannot preserve application boundaries because they are defined by the application. It is left up to you to provide for any of the appropriate recovery procedures when you use the save-while-active function to eliminate your save-outage time.

Additional recovery procedures are needed to bring the objects to a consistent state in relationship to each other after the restore operation is completed. You must determine the exact steps that are required for these recovery procedures at the time the objects are being saved. The recovery procedures must be performed after the objects from the save-while-active media are restored, but before the objects are used by any application.

You need to consider these recovery procedures if you are using the save-while-active function to eliminate your save-outage time:

Related concepts

"Eliminating save-outage time: Overview" on page 137
This information tells you what happens when you use the save-while-active function to eliminate your save-outage time.

"Example: Restoring libraries after reducing save-outage time" on page 140
This example shows a typical restore procedure after you reduce save-outage time in a library. Your exact use of the function might differ, based on your specific application requirements.

Related tasks

"Recommended recovery procedures after eliminating save-outage time" on page 143
If you perform save-while-active operations to eliminate save outage time and you specified *NOCMTBDY for the SAVACTWAIT pending record changes value, you can be left with objects that are saved with partial transactions.

If you use commitment control within your application, force one checkpoint during the save operation, and wait for transaction boundaries: If you specify SAVACT(*SYNCLIB) for the save operation, then all the data is saved with one common checkpoint. If you use commitment control to define all of the application boundaries and wait for transaction boundaries during the save operation, the recovery procedure is a basic restore of your objects.

If you use commitment control within your application, allow multiple checkpoints during the save operation, and wait for transaction boundaries: If you specify SAVACT(*SYSDFN) or SAVACT(*LIB) for the save operation, then the data is saved with multiple checkpoints. If you use commitment control to define all of the application boundaries and wait for transaction boundaries during the save operation, the recovery procedure requires you to apply or remove journaled changes to reach a common application boundary.

If you use commitment control within your application, force one checkpoint during the save operation, and do not wait for transaction boundaries: If you specify SAVACT(*SYNCLIB) for the save operation, then the data is saved with one common checkpoint. If you use commitment control and specify *NOCMTBDY on the SAVACTWAIT parameter for the save operation, the recovery procedure requires you to apply or remove journaled changes to complete or rollback your partial transactions and reach commit boundaries.

If you use commitment control within your application, allow multiple checkpoints: If you specify SAVACT(*SYSDFN) or SAVACT(*LIB) for the save operation, then the data is saved with multiple checkpoints. If you use commitment control and specify *NOCMTBDY on the SAVACTWAIT parameter for the save operation, the recovery procedure requires you to apply or remove journaled changes to complete partial transactions and bring them to a common application boundary.

If you do not use commitment control but all objects are journaled: If all application-dependent objects are journaled but commitment control is not used, then you can apply or remove journaled changes. These commands can bring all of the objects to an application boundary after restoring them from the save-while-active media. However, application boundaries are not recorded in the journal so you will need to determine where the boundaries are on an object by object basis. When the journaled object reaches a checkpoint, the journal receiver gets an additional journal entry in conjunction with the object saved journal entry. The journal entry notes that you used the save-while-active function to save the object and is used by the APYJRNCHG and RMVJRNCHG commands as the location to start the operation when the FROMENT(*LASTSAVE) parameter is used. It is critical that the currently attached journal receiver be saved along with the objects being journaled. If more than one journal is being used to journal the objects, then all attached receivers must be saved. Include the request to save the receiver in the same save request as that for the journaled objects. Or save the receiver in a separate save request after the save of the journaled objects. This save is necessary because the attached journal receiver will contain the entries that might be required by any apply or remove journaled changes operation that is part of the recovery when using the save-while-active media.

If commitment control is not used and objects are not journaled: If you do not define your application boundaries you will need to do a restore and do a recovery from an abnormal end. If you do not know what procedures are required for recovering an abnormal end, then use the method to Example: Restore libraries after reducing save-outage time.

Recommended recovery procedures after eliminating save-outage time

If you perform save-while-active operations to eliminate save outage time and you specified *NOCMTBDY for the SAVACTWAIT pending record changes value, you can be left with objects that are saved with partial transactions.

It is recommended that you use Backup, Recovery, and Media Services (BRMS) to automate your backup and recovery operations. BRMS automatically applies changes to objects with partial transactions and restores them to a usable state.

The following provides some recommended recovery procedures after restoring from the save-while-active media. The following procedure is a recommendation only. Your recovery procedures might need to be somewhat different depending on your applications and your particular application dependencies.

The recovery for journaled objects might include Apply Journaled Changes (APYJRNCHG) and Remove Journaled Changes (RMVJRNCHG) operations. The following recommendation uses the APYJRNCHG command exclusively. The APYJRNCHG command is the most common recovery operation that brings journaled objects to application boundaries. However, you can use the RMVJRNCHG command instead of the APYJRNCHG to bring the journaled objects to an application boundary. Use the RMVJRNCHG command if you are removing changes from the journaled object. You can use the RMVJRNCHG command if you are journaling before images for the journaled object.

If you need to use the APYJRNCHG command for the recovery, you must specify a known application boundary for either the ending sequence number (TOENT) parameter or the ending large sequence number (TOENTLRG) parameter but not both. Specify the FROMENTLRG parameter regardless of whether all objects reached a checkpoint together. You must run multiple APYJRNCHG commands if the objects are journaled to different journals.

The following steps give a general recommendation to follow for recovery procedures:

1. If some of the objects you are restoring are journaled objects, make sure that the necessary journals are on the system.
2. If all necessary journals are not on the system, restore the journals first. The system automatically restores the journals first if both items below are true:
   - The journals are in the same library as the objects you are restoring.
   - You used the same save request to save the journals and the objects.
3. Restore objects from the save-while-active media.
4. If some of the objects restored are journaled objects, restore any required journal receivers that do not already exist on the system.
   a. Start by restoring receivers that contain the start of save journal entries for the journaled objects.
   b. Continue restoring receivers until you restore the receiver that contains the journal entry that is the desired application boundary. These receivers need to be online for each of the journals used to journal the restored objects.
5. If all of the application-dependent objects are journaled, skip to step 9. If only some or none of the application-dependent objects are journaled, go to step 6.
6. If some application-dependent objects are not journaled objects, and one of the following scenarios is true, go to step 7. Otherwise, go to step, 8.
   a. All of the objects are in the same library and are saved using SAVACT(*LIB).
   b. All objects in all of the libraries are saved using SAVACT(*SYNCLIB).
7. You can perform the recovery procedures in Example: Restore libraries after reducing save-outage time. All of the objects reached a checkpoint together and the restored objects are in a consistent state in relationship to each other. However, if you need to bring the objects forward to some defined application boundary, you can only use the APYJRNCHG command for the journaled objects. For objects that are not journaled, you must perform user-defined recovery procedures.
8. If neither of the scenarios in 6 are true, then the objects are not saved in a consistent state in relationship to each other. Use the APYJRNCHG command to bring the journaled objects forward to some common application boundary. For objects that are not journaled, you must perform user-defined recovery procedures.
9. If all of the application-dependent objects are journaled, and all of the application-dependent objects are under commitment control, skip to step 11. Otherwise, go to step 10.

10. If all application-dependent objects are journaled objects but all of the changes made to the objects are not made under commitment control, then you must use APYJRNCHG command to bring all of the objects to an application boundary.
11. If all of the application-dependent objects are under commitment control and the objects exist in different libraries go to step 12. Otherwise, go to step 13.
12. If the objects exist in different libraries, then the objects restored are at commitment boundaries. However, not all of the objects will be at the same common commitment boundary. Bring the objects to the same common commitment boundary with the APYJRNCHG command. Specify the CMTBDY(*YES) parameter to bring the objects forward to some common application boundary.

    By specifying CMTBDY(*YES), you ensure that the apply operation starts on a commitment boundary. You also ensure that the system applies complete transactions up through the sequence number that you specified to correspond with your application boundary.
13. If all application-dependent objects are journaled objects that exist in the same library, and the files are only updated under commitment control, the system restores the files as they existed at some common commitment boundary when you saved the data.

    Use the APYJRNCHG command specifying the CMTBDY(*YES) parameter to bring the files forward to some defined application boundary if one of the following is true:

    - The common commitment transaction boundary is not an application boundary.
    - Additional transactions exist in the journal that you want to apply to the objects.

    By specifying CMTBDY(*YES), you can ensure that the apply operation starts on a commitment boundary. You also ensure that the system applies complete transactions up through the specified sequence number that corresponds to your application boundary.

    If the commitment boundary is an application boundary, then no additional recovery procedures are necessary.

Related concepts

"Example: Restoring libraries after reducing save-outage time" on page 140
This example shows a typical restore procedure after you reduce save-outage time in a library. Your exact use of the function might differ, based on your specific application requirements.

"Backing up a logical partition" on page 104
Each logical partition functions like an independent system, and needs to be backed up individually.

"Eliminating save-outage time: Overview" on page 137
This information tells you what happens when you use the save-while-active function to eliminate your save-outage time.

"Considerations for recovery procedures after eliminating save-outage time" on page 142
This topic discusses some of the considerations for save-while-active recovery procedures. In general, the system cannot preserve application boundaries because they are defined by the application. It is left up to you to provide for any of the appropriate recovery procedures when you use the save-while-active function to eliminate your save-outage time.

"Timestamp processing with save-while-active" on page 117
The save-active-time for an object can be useful when you determine which recovery procedures to use after you restore objects from the media.

Related information

BRMS

Example: Restoring objects with partial transactions

Journal management

Example: Eliminating save-outage time for libraries

This example shows a typical use of the save-while-active function to eliminate a save-outage time. Your exact use of the function might differ, based on your specific application requirements.

This example uses two libraries, LIB1, and LIB2. Both libraries contain only journaled objects and the journals for those objects. The changes made to the journaled objects might or might not be made under commitment control.

This example demonstrates a save-while-active operation that does not end the applications that are making changes to the objects in these libraries. Not ending the applications introduces additional restore considerations for the recovery operation after you restore the objects from the save-while-active media.

Eliminate the save-outage time with the following steps:

1. Submit the following command as an individual batch job:

   ```
   SAVLIB LIB(LIB1 LIB2) DEV(TAP01) SAVACT(*SYNCLIB) +
          SAVACTWAIT(600) +
          SAVACTMSGQ(QSYSOPR) +
          ACCPTH(*YES)
   ```

   Note: You can also use the SAVOBJ or SAVCHGOBJ commands, depending on your specific needs.

   The system waits 10 minutes, as specified by the SAVACTWAIT parameter, to resolve each lock conflict and for any active commitment definitions to reach a commitment boundary during checkpoint processing.

   By specifying ACCPTH(*YES), you are also saving access paths for the logical files. Access paths, in most cases, will not be built after restoring the files from this save media.

   The recovery procedures needed when restoring objects from this media are dependent on each of the database members in LIB1 and LIB2 being updated with the timestamp of this save operation.
2. When checkpoint processing is complete, QSYSOPR receives message CPI3712 as specified by the SAVACTMSGQ parameter. Until the QSYSOPR message queue receives the CPI3712 message, monitor lock conflicts that the save-while-active job might encounter.
3. Wait for the save-while-active job to complete.
4. After the batch job has completed, verify that all of the required objects were saved. If lock conflicts prevented some of the objects from being saved, you should issue the original save command again after resolving any and all lock conflicts.
5. Save the receiver containing the earliest start of save entry from each journal being used to journal the objects in libraries LIB1 and LIB2. You can get the earliest receiver from the OUTFILE on the save command. If the attached journal receivers do not reside in library LIB1 or LIB2, then you must issue separate save requests to save each of the attached receivers.

   Save all of the attached receivers with the following command. Multiple save commands might be necessary for this step. It is not necessary to use the save-while-active function when saving journal receivers. The following command defaults to SAVACT(*NO).

   ```
   SAVOBJ OBJ(attached-receiver) +
          LIB(attached-receiver-library) +
          OBJTYPE(*JRNRCV) +
          DEV(TAP01)
   ```

Example: Saving objects with partial transactions

This example shows a typical use of the save-while-active function to eliminate save-outage time by not waiting for commitment boundaries. Your exact use of the function might differ, based on your specific application requirements.

This example uses a checking and savings account. Both libraries contain journaled objects and the journals for those objects. The changes might or might not be made under commitment control.

This example demonstrates a save without waiting for commitment boundaries and does not end the applications that are making changes to the objects that are in these libraries. Not ending the applications introduces additional restore considerations for the recovery operation after you restore the objects from the media.

Use the following steps to eliminate save-outage time without waiting for commitment boundaries:

1. Submit the following command before the transaction is ended:

```
SAVLIB LIB(CHK SAV)  DEV(TAP01) SAVACT(*SYNCLIB) +
 SAVACTWAIT(30 *NOCMTBDY 30) +
 SAVACTMSGQ(QSYSOPR) +
 ACCPTH(*YES)
```

   Note: You can also use the SAVOBJ or SAVCHGOBJ commands, depending on your specific needs.

   The system waits 30 seconds, as specified by the SAVACTWAIT parameter to resolve each lock conflict during checkpoint processing. The objects will not be saved if lock conflicts are not resolved within the specified time.

   By specifying ACCPTH(*YES), you are also saving access paths for the logical files. Access paths, in most cases, will not be built after restoring the files from this save media.

   The recovery procedures needed when restoring objects from this media are dependent on each of the database members in the CHK and SAV being updated with the time stamp of this save operation.

2. When checkpoint processing is complete, QSYSOPR receives message CPI3712 as specified by SAVACTMSGQ parameter. Until the QSYSOPR message queue receives the CPI3712 message, monitor lock conflicts that the save-while-active job might encounter.
3. Wait for the save job to complete.
4. After the batch job has completed, verify that all of the required objects were saved. If any objects were saved in a partial state, the files must be either rolled forward or backward to a consistent state before they can be used.
5. Save the appropriate receivers of each journal being used to journal the objects in libraries CHK and SAV. You must include the receivers to be saved starting with the receiver containing the start of commit entry for any transactions which were open when the save checkpoint processing took place through the attached receiver. The save OUTFILE will indicate the name of the earliest receiver for each object which will need to be available to use the APYJRNCHG command during the recovery process. You must issue a separate save request to save these receivers if these receivers do not exist in library CHK or SAV

   Note: It is highly recommended that you save all of the attached receivers with the following command.

   Multiple save commands might be necessary for this step. Note that it is not necessary to use the save-while-active function when saving journal receivers. The following command defaults to SAVACT(*NO).

```
SAVOBJ OBJ (attached-receiver)+
LIB (attached-receiver-library)+
OBJTYPE(*JRNRCV)+
DEV(TAP01)
```

Example: Eliminate save-outage time for a directory

This example shows a typical use of the save-while-active function to eliminate save-outage time in a directory. Your exact use of the function might differ, based on your specific application requirements.

This example uses the directory, MyDirectory. MyDirectory contains only journaled objects.

This example demonstrates a save-while-active operation that does not end the applications that are making changes to the objects in this directory. Not ending the applications introduces additional restore considerations for the recovery operation after you restore the objects from the save-while-active media.

Eliminate the save-outage time with the following steps:

1. Submit the following command as an individual batch job:

```
SAV DEV('/QSYS.LIB/TAP01.DEVD')  +
    OBJ('/MyDirectory') UPDHST (*YES) SAVACT(*SYNC) +
  SAVACTMSGQ(QSYS.LIB/LIB1.LIB/MSGQ1.MSGQ) +
```

2. When checkpoint processing is complete for the directory, the message queue receives the message CPI3712, as specified by the SAVACTMSGQ parameter. Until the message queue, MSQ1, receives the CPI3712 message, monitor lock conflicts that the save-while-active job might encounter.
3. Wait for the save-while-active job to complete.
4. After the batch job has completed, verify that all of the required objects were saved. If lock conflicts prevented some of the objects from being saved, you should issue the original save command again after resolving any and all lock conflicts.
5. Save the attached receiver of each journal being used to journal the objects in directory MyDirectory.

   Save all of the attached receivers with a command such as the one below. Multiple save commands might be necessary for this step. It is not necessary to use the save-while-active function when saving journal receivers. The following command defaults to SAVACT(*NO).

   ```
   SAV DEV('/QSYS.LIB/TAP01.DEVD')  +
       OBJ('/QSYS.LIB/MYLIB.LIB/JRNR*.JRNRCV')
   ```

Example: Restoring libraries after eliminating save-outage time

This example shows a typical restore procedure after you eliminate save-outage time in a library. Your exact use of the function might differ, based on your specific application requirements.

Perform the following steps when restoring libraries LIB1 and LIB2:

1. Restore the two libraries with the following commands:

   ```
   RSTLIB SAVLIB(LIB1) DEV(TAP01)

   RSTLIB SAVLIB(LIB2) DEV(TAP01)
   ```

   If the journals still exist on the system, they are not restored. That is not a problem.

   If they did not exist, the system will restore the journal objects before the other objects.

   At the completion of these restore commands, the objects exist on the system, but they will not be in a consistent state in relationship to each other.
2. Restore the necessary journal receivers that were attached at the time the libraries were saved. If the journal receivers are in libraries other than LIB1 or LIB2 at the time of the save and they do not currently exist on the system, use the following restore command to restore the receivers:

   ```
   RSTOBJ OBJ(attached-receiver-at-save-time) +
          SAVLIB(receiver-library) +
          DEV(TAP01)
   ```

   If the attached receivers were in LIB1 or LIB2 when you saved the data and they did not exist before the RSTLIB operation, they were restored as part of that RSTLIB operation.
3. Determine a point in time, or application boundary, in which to bring the objects in LIB1 and LIB2. This way all of the objects are in a consistent state in relationship to each other. After determining the desired application boundary, you might need to restore additional journal receivers. If you need to restore additional journal receivers, but the receivers are not online, restore them with the following restore command. Multiple restore commands might be necessary for this step:

   ```
   RSTOBJ OBJ(other-needed-receivers) +
          SAVLIB(receiver-library) +
          DEV(TAP01)
   ```

   The Work with Journal Attributes (WRKJRNA) and Display Journal (DSPJRN) commands can be helpful in finding the application boundary.

   You can use the WRKJRNA command to determine the appropriate range of receivers you need for the ensuing Apply Journaled Changes (APYJRNCHG) operations. You can use the DSPJRN command to locate the exact sequence number that identifies the desired application boundary. If multiple journals are involved, you must locate the same application boundary (most likely identified by the timestamp) in each journal. You must also note the appropriate journal sequence number.

4. Bring the objects forward to a specific application boundary with one of the following Apply Journaled Changes (APYJRNCHG) commands. Different variations of the APYJRNCHG command might be appropriate based on the given criteria.

   If any objects received changes during the save operation, and they were under commitment control, the commitment boundaries will be preserved on the following APYJRNCHG commands. If you do not want the commitment control boundaries preserved, then you specify CMTBDY(*NO) on the following APYJRNCHG commands:

   a. Use the commands below to apply the journaled changes to the objects if the following is true:
      - The journaled objects for which changes are to be applied were saved.
      - You did not restore the journal (which is not a problem) because the objects were being restored to the system from where they were saved.
      - The media used represent the most recent save of the objects.
      - You saved the objects specifying UPDHST(*YES) on the save command.

      ```
      APYJRNCHG JRN(jrnlib/jrnname) +
                OBJ((LIB1/*ALL)) +
                TOENT(seq#-for-application-boundary)

      APYJRNCHG JRN(jrnlib/jrnname) +
                OBJ((LIB2/*ALL)) +
                TOENT(seq#-for-application-boundary)
      ```

      If multiple journals are involved, then repeat these commands for each journal specifying the correct sequence number (TOENT parameter) that identifies the desired application boundary. Note that the TOENT sequence number is very likely different for each journal in LIB1 and LIB2, but they all identify a common application boundary.

   b. Use the commands below to apply the journaled changes to the objects if the following is true:
      - The objects were saved prior to V5R3.
      - You restored the journal.
      - The media used represent the most recent save of the objects.
      - You saved the objects specifying UPDHST(*YES) on the save command.

      ```
      APYJRNCHG JRN(jrnlib/jrnname) +
           OBJ((LIB1/*ALL)) +
                RCVRNG(rcv-attached-at-save-time +
                       ending-rcv) +
                TOENT(seq#-for-application-boundary)

      APYJRNCHG JRN(jrnlib/jrnname) +
                OBJ((LIB2/*ALL)) +
                RCVRNG(rcv-attached-at-save-time +
                       ending-rcv) +
                TOENT(seq#-for-application-boundary)
      ```

      If multiple journals are involved, then repeat these commands for each journal specifying the correct sequence number (TOENT parameter) that identifies the desired application boundary. Note that the TOENT sequence number is very likely different for each journal in LIB1 and LIB2, but they all identify a common application boundary. If the journaled objects for which changes are going to be applied were saved in V5R3 or later, then the system can determine the correct receiver range when the default of RCVRNG(*LASTSAVE) is used. In this situation, the apply command from step a works.

   c. If your objects were saved before V5R3 and the save-while-active media used does not represent the most recent save of the objects specifying UPDHST(*YES), do the following commands.

      1) Use the DSPJRN command to determine the sequence number of the start-of-save journal entry for each object.
      2) Issue an individual APYJRNCHG command for each of the objects.

      The following example demonstrates such an APYJRNCHG command:

```
APYJRNCHG JRN(jrnlib/jrnname) +
          OBJ((filelib/filename filembr)) +
          RCVRNG(rcv-attached-at-save-time +
                 ending-rcv) +
          FROMENT(seq#-for-start-of-save-entry) +
          TOENT(seq#-for-application-boundary)
```

Some of the APYJRNCHG commands could specify multiple objects if there is a continuous series of start-of-save entries in the journal. The members identified by the continuous series of start-of-save journal entries could be applied to with a single APYJRNCHG command by specifying the earliest sequence number of all the start-of-save entries in the continuous series for the FROMENT parameter. Use the *LASTSAVE value in the FROMENT parameter.

Example: Restoring objects with partial transactions

If you perform save-while-active operations that can result in objects that are saved with partial transactions, it is recommended that you use Backup, Recovery, and Media Services (BRMS).

You can use BRMS to automate your backup and recovery operations. BRMS automatically applies changes to objects with partial transactions and restores them to a usable state.

If an object is saved with partial transactions, FROMENT(*LASTSAVE) will be required when applying or removing journaled changes on the restored version of the object.

When you use the character-based interface to restore objects with partial transactions, perform the following steps to restore libraries CHK and SAV:

1. Restore the two libraries with the following commands:

   ```
   RSTLIB SAVLIB(CHK) DEV(TAP01)

   RSTLIB SAVLIB(SAV) DEV(TAP01)
   ```

   If the journals still exist on the system, they are not restored. However, this is not a problem.

   If they did not exist, the system will restore the journal objects before the other objects.

2. Restore the earliest receiver as specified by the outfile. If the journal receivers are in libraries other than CHK or SAV at the time of the save and they do not currently exist on the system, use the following restore command to restore the receivers:

   ```
   RSTOBJ OBJ(attached-receiver-at-save-time) +
          SAVLIB(receiver-library) +
          DEV(TAP01) +
          OUTPUT(*OUTFILE)OUTFILE(lib/file)
   ```

   If the attached receivers were in CHK or SAV when you saved the data and they did not exist before the RSTLIB operation, they were restored as part of that RSTLIB operation.

3. Determine a point in time, or application boundary, in which to bring the objects in CHK and SAV. This way all of the objects are in a consistent state in relationship to each other. After determining the desired application boundary, you might need to restore additional journal receivers. You can use the WRKJRNA command to determine the appropriate range of receivers you need for the ensuing Apply Journaled Changes (APYJRNCHG) operations. You can use the DSPJRN command to locate the exact sequence number that identifies the desired application boundary. If multiple journals are involved, you must locate the same application boundary (most likely identified by the timestamp) in each journal. You must also note the appropriate journal sequence number. If you need to restore additional journal receivers, but the receivers are not online, restore them with the following restore command. Multiple restore commands might be necessary for this step:

   ```
   RSTOBJ OBJ(other-needed-receivers) +
          SAVLIB(receiver-library) +
          DEV(TAP01)
   ```

4. Bring the objects forward to a specific application boundary with one of the following Apply Journaled Changes (APYJRNCHG) commands. Different variations of the APYJRNCHG command might be appropriate based on the given criteria.

If any objects received changes during the save operation, and they were under commitment control, the commit boundaries will be preserved on the following APYJRNCHG commands. If you do not want to have the commitment control boundaries preserved, then you need to specify CMTBDY(*NO) on the following APYJRNCHG commands.

a. Use the commands below to apply the journaled changes to the objects (completed or partial) if the following is true:
   - You did not restore the journal because the objects were being restored to the system from where they were saved.
   - The media used represent the most recent save of the objects.
   - You saved the objects specifying UPDHST(*YES) on the save command.
   - The libraries CHK and SAV are journaled libraries.

```
APYJRNCHG JRN(jrnlib/jrnname) +
          FROMENT(*LASTSAVE) +
          OBJ((CHK/*ALL *ALL)) +
          TOENTLRG(seq#-for-application-boundary)

APYJRNCHG JRN(jrnlib/jrnname) +
          FROMENT(*LASTSAVE) +
          OBJ((SAV/*ALL *ALL)) +
          TOENTLRG(seq#-for-application-boundary)
```

   If multiple journals are involved, then repeat these commands for each journal specifying the correct sequence number (TOENTLRG parameter) that identifies the desired application boundary. Note that the TOENTLRG sequence number is very likely different for each journal in CHK and SAV, but they all identify a common application boundary.

b. Use the commands below to apply the journaled changes to the objects (completed or partial) if the following is true:
   - You restored the journal.
   - The media used represent the most recent save of the objects.
   - You saved the objects specifying UPDHST(*YES) on the save command.
   - The libraries CHK and SAV are journaled libraries.

```
APYJRNCHG JRN(jrnlib/jrnname) +
          OBJ((CHK/*ALL *ALL)) +
          RCVRNG(rcv-attached-at-save-time +
                 ending-rcv) +
          FROMENT(*LASTSAVE) +
          TOENTLRG(seq#-for-application-boundary)

APYJRNCHG JRN(jrnlib/jrnname) +
          OBJ((SAV/*ALL *ALL)) +
          RCVRNG(rcv-attached-at-save-time +
                 ending-rcv) +
          FROMENT(*LASTSAVE) +
          TOENTLRG(seq#-for-application-boundary)
```

   If multiple journals are involved, then repeat these commands for each journal specifying the correct sequence number (TOENTLRG parameter) that identifies the desired application boundary. Note that the TOENTLRG sequence number is very likely different for each journal in CHK and SAV, but they all identify a common application boundary.

c. Do the following commands if the save-while-active media used does not represent the most recent save of the objects specifying UPDHST(*YES).
   1) Use the DSPJRN command to determine the sequence number of the start-of-save journal entry for each object.
   2) Issue an individual APYJRNCHG command for each of the objects.

   The following example demonstrates such an APYJRNCHG command:

```
APYJRNCHG JRN(jrnlib/jrnname) +
          OBJ((filelib/filename filembr)) +
          RCVRNG(rcv-attached-at-save-time +
                 ending-rcv) +
          FROMENT(seq#-for-start-of-save-entry) +
          FROMENT(*LASTSAVE) +
          TOENT(seq#-for-application-boundary)
```

If you are using a release after V5R3 and the most recent save of the objects is not being used, FROMENT(*LASTSAVE) cannot be specified on the APYJRNCHG commands. An individual sequence number must be specified for each of the objects in libraries CHK and SAV.

Some of the APYJRNCHG commands could specify multiple objects if there is a continuous series of start-of-save entries in the journal. The members identified by the continuous series of start-of-save journal entries could be applied to with a single APYJRNCHG command by specifying the earliest sequence number of all the start-of-save entries in the continuous series for the FROMENT parameter. If you are using V5R3 or later, use the *LASTSAVE value in the FROMENT parameter.

Related information

Backup, Recovery, and Media Services

Example: Restoring a directory after eliminating save-outage time

This example shows a typical restore procedure after you eliminate save-outage time in a directory. Your exact use of the function might differ, based on your specific application requirements.

Perform the following steps when restoring directory MyDirectory:

1. Restore the directory with the following command:

   ```
   RST DEV('/QSYS.LIB/TAP01.DEVD') +
     OBJ('/MyDirectory')
   ```

   At the completion of these restore commands, the objects exist on the system, but they will not be in a consistent state in relationship to each other.

2. Restore the necessary journal receivers that were attached at the time the directory was. Use, a command such as the following to restore the receivers:

   ```
   RST DEV('/QSYS.LIB/TAP01.DEVD') +
     OBJ('receiver-path')
   ```

3. Determine a point in time, or application boundary, in which to bring the objects in MyDirectory. This way all of the objects are in a consistent state in relationship to each other. After determining the desired application boundary, you might need to restore additional journal receivers. If you need to restore additional journal receivers, but the receivers are not online, restore them with a restore command such as the following. Multiple restore commands might be necessary for this step:

   ```
   RST  DEV('/QSYS.LIB/TAP01.DEVD') +
     OBJ('receiver-path')
   ```

   The Work with Journal Attributes (WRKJRNA) and Display Journal (DSPJRN) commands can be helpful in finding the application boundary.

   You can use the WRKJRNA command to determine the appropriate range of receivers you need for the ensuing Apply Journaled Changes (APYJRNCHG) operations. You can use the DSPJRN command to locate the exact sequence number that identifies the desired application boundary. If multiple journals are involved, you must locate the same application boundary (most likely identified by the timestamp) in each journal. You must also note the appropriate journal sequence number.

4. Bring the objects forward to a specific application boundary with one of the following Apply Journaled Changes (APYJRNCHG) commands. Different variations of the APYJRNCHG command might be appropriate based on the given criteria.

   a. Use the commands below to apply the journaled changes to the objects if the following is true:
      - The objects were saved prior to V5R3.
      - You did not restore the journal.

- The media used represent the most recent save of the objects
- You saved the objects specifying UPDHST(*YES) on the save command.
- If the above conditions are not met but you are using V5R3.

```
APYJRNCHG JRN(jrnlib/jrnname) +
          OBJPATH(/MyDirectory) +
          SUBTREE(*ALL)+
          TOENT(seq#-for-application-boundary)
```

If multiple journals are involved, then repeat these commands for each journal specifying the correct sequence number (TOENT parameter) that identifies the desired application boundary.

b. Use the commands below to apply the journaled changes to the objects if the following is true
   - The objects were saved prior to V5R3.
   - You restored the journal.
   - The media used represent the most recent save of the objects.
   - You saved the objects specifying UPDHST(*YES) on the save command.

```
APYJRNCHG JRN(jrnlib/jrnname) +
          OBJPATH(/MyDirectory) +
          SUBTREE(*ALL)+
          RCVRNG(rcv-attached-at-save-time +
                 ending-rcv) +
          TOENT(seq#-for-application-boundary)+
```

In the situation where the journal is restored, and the journaled objects for which changes are going to be applied were saved prior to V5R3, the system cannot determine the correct receiver range. Therefore, the correct range of receivers must be specified on the RCVRNG parameter. The attached receiver at the time that the directory was saved is the specified starting journal receiver. If the journaled objects for which changes are going to be applied were saved in V5R3 or later, then the system can determine the correct receiver range when the default of RCVRNG(*LASTSAVE) is used. In this situation, the apply command from step a works correctly.

If multiple journals are involved, then repeat these commands for each journal specifying the correct sequence number (TOENT parameter) that identifies the desired application boundary.

c. If you are not using V5R3, do the following commands if the save-while-active media used does not represent the most recent save of the objects specifying UPDHST(*YES).
   1) Use the DSPJRN command to determine the sequence number of the start of save journal entry for each object.
   2) Issue an individual APYJRNCHG command for each of the objects.

   The following example demonstrates such an APYJRNCHG command:

```
APYJRNCHG JRN(jrnlib/jrnname) +
          OBJPATH(/MyDirectory) +
          RCVRNG(rcv-attached-at-save-time +
                 ending-rcv) +
          FROMENT(seq#-for-save or start-of-save-entry) +
          TOENT(seq#-for-application-boundary)
```

Because the most recent save of the objects is not being used, you cannot specify FROMENT(*LASTSAVE) on the APYJRNCHG command. You must specify an individual sequence number for directory MyDirectory

Some of the APYJRNCHG commands could specify multiple objects if there is a continuous series of save or start-of-save entries in the journal. The objects identified by the continuous series of save or start-of-save journal entries could be applied to with a single APYJRNCHG command by specifying the earliest sequence number of all the save or start-of-save entries in the continuous series for the FROMENT parameter. Use the *LASTSAVE value in the FROMENT parameter.

Encrypted backups

If you use an encrypting tape drive, you can use save commands or Backup, Recovery, and Media Services (BRMS) to perform an encrypted backup. However, if you use the software encryption method, you must use BRMS to perform the encrypted backup.

Related tasks

"Performing a complete save using the GO SAVE checklist" on page 34
Use this checklist to perform a complete save operation.

Related information

Managing master keys

Loading and setting save/restore master key

The save/restore master key is a special purpose master key used to encrypt all the other master keys when you save them in a Save System (SAVSYS) operation. The save/restore master key itself is not saved. The save/restore master key has a default value. So, for optimum security, the save/restore master key should be set to another value.

The save/restore master key has only two versions. The versions are new and current.

Note: Since the save/restore master key is not included in the Save System operation, it is recommended that you write the passphrases for the save/restore master key and store them securely.

You should set the save/restore master key before performing the SAVSYS operation. To set the save/restore master key, you must first load master key parts and then set the save/restore master key.

You can load as many master key parts as you want for the save/restore master key. Setting the save/restore master key causes the new save/restore master key version to move to the current save/restore master key version. After the save/restore master key has been set, you should perform the SAVSYS operation to save the master keys on the save media.

To load a save/restore master key from the IBM Systems Director Navigator for i5/OS interface, follow these steps:

1. Select **Security** from your IBM Systems Director Navigator for i5/OS window.
2. Select **Cryptographic Services Key Management**.
3. Select **Manage Master Keys**.
4. Select the **Save/restore master key**.
5. Select **Load Part** from the **Select Actions** menu.
6. Specify the **Passphrase** and click **OK**.

If you prefer to write your own application to load the save/restore master key, you can do so by using the Load Master Key Part (QC3LDMKP; Qc3LoadMasterKeyPart) API.

You can also use the Add Master Key Part (ADDMSTPART) CL command to load a master key part for the save/restore master key.

To set the save/restore master key, select the **Save/restore master key** and then from the **Select Actions** menu, select **Set**.

If you prefer to write your own application to set the save/restore master key, you can do so by using the Set Master Key (QC3SETMK; Qc3SetMasterKey) API.

You can also use the Set Master Key (SETMSTKEY) CL command to set the save/restore master key that has parts already added.

You should also perform a SAVSYS operation whenever you load and set any of the master keys

Saving and restoring master keys

If a master key is lost, all keys encrypted under that master key, and consequently all data encrypted under those keys, are lost. Therefore, it is important to backup your master keys.

There are two methods of backing up your master keys:

- **Save the individual passphrases**

  Master key passphrases should not be stored on the system in plaintext. Also, do not encrypt them under any master key or any key encrypted under a master key. If the master keys are lost (for example, when the Licensed Internal Code is installed) or damaged, you will be unable to recover the passphrases and therefore the master keys. Store the passphrases securely outside the system, such as in separate safes.

- **Save the master keys by performing a SAVSYS operation**

  Master keys are saved as part of a SAVSYS operation. To protect the master keys while on save media, they are encrypted with the save/restore master key. The save/restore master key is the only master key that is not saved as part of the SAVSYS operation.

  To back up the master keys, follow these steps:

  1. Set the save/restore master key.
  2. Perform a SAVSYS operation.

  To recover the master keys on the target system, the save/restore master key on the target system must match the save/restore master key on the source system at the time of the SAVSYS operation. If they match, the master keys are automatically decrypted and made ready for use. If they do not match, the restored master keys are put in pending versions. When you attempt to use a master key that has a pending version (for example, you encrypt using a key from a keystore file that is encrypted under a master key with a pending version), you get an error message indicating there is an unrecovered master key. You must either recover the pending master key version by setting the correct value for the save/restore master key on the target system, or you must clear the pending master key version.

  The save/restore master key has a default value. Therefore, if it is not changed on either the source or target systems, the master keys will restore without any intervention. However, using the default save/restore master key is not recommended as this provides little protection. You should load and set the save/restore master key for optimum security of the master keys while on SAVSYS media.

  When master keys are restored and decrypted successfully with the save/restore master key, they are moved into the current versions. If a master key already has a current version, it is moved to the old version. Therefore, it is important that there are no keys on the system encrypted under the old version, because that will be lost. After restoring the master keys, you must translate all keystore files and any other keys encrypted under a master key.

  There might be instances when you do not want your master keys, or some of your master keys, to be distributed to another system through the SAVSYS media. When you do not want any of your master keys to successfully restore and decrypt on another system, ensure you have loaded and set the save/restore master key prior to the SAVSYS operation, and do not share it with the target system. On the target system, the pending versions are needed to be cleared.

  If you want to distribute only some of your master keys, you can do the same. Then, share the passphrases for the master keys you want to share. Otherwise, you will need to temporarily clear the master keys you do not want distributed.

  Even when the master keys are backed up using the SAVSYS operation, you should write down the passphrases for the master keys and store them securely; this is in case the Licensed Internal Code install from the SAVSYS operation fails.

  Note: Any time you change a master key, you must back it up.

Backing up encrypted auxiliary storage pools

Disk encryption enables you to encrypt data stored in user auxiliary storage pools (ASPs) and independent ASPs. You back up an encrypted ASP in the same way as for an unencrypted ASP. However, if the data in the system ASP or independent ASP is lost, you need to perform additional recovery steps.

In order to use disk encryption, you must have installed Option 45 - Encrypted ASP Enablement, a feature of the operating system. The option to enable encryption is available when you create an user ASP or independent ASP using Systems Director Navigator for i5/OS or System i Navigator.

When you set up an encrypted ASP, the system generates a data key, which encrypts the data written to that storage pool and decrypts data read from that storage pool. The data keys for independent ASPs are kept with the storage pool and are protected with the ASP master key. User ASPs are protected with a data key that is stored in the Licensed Internal Code

Data is encrypted only while it resides on the ASP. When you read the data, it is decrypted. When doing a save operation, the data is decrypted as it is read for the save operation. The data is encrypted on the save media only if you are doing an encrypted backup using either an encrypting tape drive or the software solution.

You can perform an encrypted backup of data in an encrypted ASP. During the backup, the ASP data is decrypted as it is read, and gets encrypted again as it is written to the tape.

To back up the data in an encrypted ASP, use any of the following commands:

- SAVSYS command
- GO SAVE Option 21 (saves the entire system)
- GO SAVE Option 23 (saves user data)

Important: If you switch an encrypted independent ASP from one system to another in a cluster, you need to make sure that the ASP master key is set to the same value on both systems.

Related tasks

"Saving independent ASPs" on page 55
You can save independent auxiliary storage pools (ASPs) in System i Navigator) separately, or you can save them as part of a full system save (GO SAVE Option 21), or when you save all user data (GO SAVE: Option 23). Independent ASPs are also known as *independent disk pools*.

Related information

Loading and setting auxiliary storage pool (ASP) master key

Restoring encrypted auxiliary storage pools

Disk encryption

Backup programming techniques

The programming techniques include recovering jobs, displaying status messages, and redirecting output from save and restore commands to an output file.

Considerations for job recovery

Job recovery and starting again should be a basic part of application design. Applications should be designed to handle.

- Unexpected data problems, such as alphabetic data occurring where numeric data is expected
- Operator problems, such as operators taking the wrong option or canceling the job
- Equipment problems, such as workstation, disk unit, and communication line failures

Job recovery procedures should ensure the integrity of the user's data and allow for easy starting of the interrupted application. Journaling and commitment control can be used in application design to help in job recovery. Recovery procedures should be transparent to the end users.

Interactive job recovery

If you are running a data entry job or one that updates a single file, it is unlikely that you need to plan an extensive recovery strategy. The operators can inquire against the file to determine which record was last updated and then continue from that point.

To recover from inquire-only jobs, the workstation operators start where they left off. When using update transactions for many files, consider using a journal or commitment control. The system automatically recovers journaled files during the initial program load (IPL) following an abnormal end of the system, or during make available (vary on) processing of an independent ASP after an abnormal vary off. In addition, the journal can be used for user-controlled forward or backward file recovery. You can protect with journaling other object types, in addition to database physical files.

Commitment control, using the file changes recorded in the journal, provides automatic transaction and file synchronization. During job end, the system automatically rolls back file updates to the beginning of the transaction. In addition, the commitment control notify object can assist you in restarting the transaction.

When designing an interactive application, consider the possibility that you can experience equipment problems with your workstations and communications lines. For example, suppose your computer system loses power. If you have an uninterruptible power supply installed to maintain power to the processing unit and disk units, your system remains active. However, in this example, your workstations lost power. When your programs attempt to read or write to the workstations, an error indication is returned to the program. If the application is not designed to handle these errors, the system can spend all its time in workstation error recovery.

You should design your interactive applications to look at error feedback areas and handle any errors indicated. If the application handles the errors and stops, the system resource is not used to do nonproductive error recovery. Examples of using error feedback areas and error recovery routines can be found in the programming languages reference manuals.

Batch job recovery

Print-only batch jobs normally do not need special recovery to start again. Running the program again might be adequate.

Batch jobs that perform file updates (add, change, or delete actions) present additional considerations for starting again and recovery. One approach to starting again is to use an update code within the record. As a record is updated, the code for that record can also be updated to show that processing for that record is complete. If the job is started again, the batch program positions itself (as a result of the update code) to the first record that it had not processed. The program then continues processing from that point in the file.

Another way to start batch processing again is to save or copy the file before starting the job. You can use one of the following commands to save or copy the file:

- Save Object (SAVOBJ)
- Copy File (CPYF)

Then, if you need to start again, restore or copy the file to its original condition and run the job again. With this approach, you need to ensure that no other job is changing the files. One way to ensure this is to get an exclusive lock on the file while the job is running. A variation of this approach is to use the journal. For example, if starting again is required, you could issue the Remove Journal Change (RMVJRNCHG) command to remove changes to the files. Then, run the job again against the files.

If your batch job consists of a complex input stream, you probably want to design a strategy for starting again into the input stream. Then, if the batch job needs to be started again, the job determines from what point the stream continues.

Commitment control also can be used for batch job recovery. However, if you plan to use commitment control for batch jobs, consider that the maximum number of record locks allowed in a commit cycle is 4 000 000. Therefore, you might need to divide the batch job into logical transactions. For example, if your batch program updates a master file record followed by several detail records in another file, each of those sets of updates can represent a logical transaction and can be committed separately. Locks are held on all records changed within a commit cycle. Therefore, changed data is made available more quickly if your batch job is divided into small, logical transactions.

Journaling can also be used to assist in batch job recovery just as it can be for interactive jobs.

Information in output files

Most save commands create output that shows what the system saved. Depending on which command you use, you can direct this output to a printer (OUTPUT(*PRINT)), a database file (OUTPUT(*OUTFILE)), a stream file, or a user space.

The default for save commands is not to create output. You must request it each time you run the save command. You can change the default for the OUTPUT parameter for save commands by using the Change Command Default (CHGCMDDFT) command.

You can do one of two things: print the output and store it with your media, or create a program to analyze and report on the information in the output file.

You can use the OUTPUT parameter with these commands:

SAV	SAVDLO	SAVSAVFDTA	SAVSYSINF
SAVCFG	SAVLIB	SAVSECDTA	
SAVCHGOBJ	SAVOBJ	SAVSYS	

If you use an output file for the Save Document Library Object (SAVDLO) command, the system uses the file format QSYS/QAOJSAVO.OJSDLO. Use the Display File Field Description (DSPFFD) command to look for the file layout.

The SAV command does not support sending output to an output file. You can send output from the SAV command to a stream file or to a user space. “Interpreting output from save (SAV) and restore (RST)” on page 159 shows the layout for the stream file or user space.

If you use an output file for any of the other commands that are listed above, the system uses the file format QSYS/QASAVOBJ.QSRSAV.

The SAVCHGOBJ, SAVLIB, SAVOBJ, and SAV commands have an information type (INFTYPE) parameter to specify how much detail you want in the output. See “Interpreting output from save commands” on page 176 for more information.

For the names of the model database output files that the save commands use, see the online information for the save commands.

Related reference

“Save operation output file information” on page 176
This table shows the format for the save operation (QASAVOBJ) output file information. Unused fields, fields that are not set, contain a value of zero for numeric fields and blanks for character fields.

Interpreting output from save (SAV) and restore (RST)

When you use the Save (SAV) command or the Restore (RST) command, you can direct output to a stream file or to a user space.

If data already exists in the stream file or user space that you specify, the command writes over that data. It does not append the new data to any existing data.

To specify a stream file, you must have *W authority to the stream file and *R authority to the directory for the stream file.

To specify a user space, you must have *CHANGE authority to the user space and *USE authority to the library. The server needs an *EXCLRD lock on the user space.

Related concepts

"Using the Save (SAV) command" on page 78
This information explains how to use the SAV command with the OBJ parameter.

Related reference

"Determining objects that the system saved (save messages)" on page 7
This information describes how save messages work and what information is available from the output files.

Entry header information

When a Save (SAV) command or the Restore (RST) command is run, the output can be directed to a stream file or user area.

The content of the output is divided into entries. Each entry in the output has an associated header. This header contains data that specifies the length of the entry and the type of the entry. Each type of entry has its own format. This header information allows the content of the output to be divided into entries that have specific formats. This enables the data in the output to be parsed.

No count of the entries is kept, instead, the end of an entry is determined by *entry length*. An entry might contain variable length elements. This might result in the entry being padded.

The number of entries in the output is variable. Entries will appear one after the other until a trailer entry is reached. The trailer entry is the last entry in the output.

For each field in the header, an offset is specified in bytes. This offset is relative to the base address of the header, or the beginning of the first field in the header.

The table below shows the format for the header information as it is output by the SAV or RST commands.

Table 45. Entry header information output–SAV and RST commands

Offset (bytes)				
Decimal	**Hex**	**Type (in bytes)**	**Set by[1]**	**Field**
0	0	BINARY(4)	S/R	Entry type
4	4	BINARY(4)	S/R	Entry length

Note:

1. **Set by column.** The following column values indicate which operations write the content of the field into the output:

   Value **Condition**

Note:

S	Save operation set this field.
R	Restore operation set this field.
S/R	Either operation set this field.
(blank)	Not set by either operation. The associated field is set to zero for numeric fields, blank for character fields, or empty for variable-length character fields.

Command information entries

This table describes the format of the command output for the SAV and RST commands.

Command information entries are output in the format described in the table below. The *entry type* value in the header determines if the entry associated with the header is a command information entry.

The system associates a coded character set identifier (CCSID) with all data. This association is maintained across all save and restore operations.

For each field, an offset is specified in bytes. This offset is relative to the base address of the entry, or the beginning of the first field in the entry header.

Table 46. Command information entry output–SAV and RST commands

Offset (bytes)		Type (in bytes)	Set by[1]	Field
Decimal	**Hex**			
0	0	BINARY(8)	S/R	See the table in Entry header information for more format details.
8	8	BINARY(4)	S/R	Device name offset [2]
12	C	BINARY(4)	S/R	File label offset [3]
16	10	BINARY(4)	S/R	Sequence number
20	14	BINARY(4)	S/R	Save active
24	18	BINARY(4)	S/R	CCSID of data
28	1C	BINARY(4), UNSIGNED	S/R	Number of records
32	20	CHAR(10)	S/R	Command
42	2A	CHAR(10)	S/R	Expiration date
52	34	CHAR(8)	S/R	Save date/time
60	3C	CHAR(10)	S/R	Start change date
70	46	CHAR(10)	S/R	Start change time
80	50	CHAR(10)	S/R	End change date
90	5A	CHAR(10)	S/R	End change time
100	64	CHAR(6)	S/R	Save release level
106	6A	CHAR(6)	S/R	Target release level
112	70	CHAR(1)	S/R	Information type
113	71	CHAR(1)	S/R	Data compressed
114	72	CHAR(1)	S/R	Data compacted
115	73	CHAR(8)	S/R	Save system serial number
123	7B	CHAR(8)	R	Restore date/time

Table 46. Command information entry output–SAV and RST commands (continued)

Offset (bytes)		Type (in bytes)	Set by[1]	Field
Decimal	Hex			
131	83	CHAR(6)	R	Restore release level
137	89	CHAR(8)	R	Restore system serial number
145	91	CHAR(10)	S/R	Save active option
155	9B	CHAR(1)	S/R	Save format
156	9C	BINARY(4)	S/R	Media file number
160	A0	BINARY(4)	S/R	Total media files
164	A4	CHAR(1)	S/R	Private authorities requested
165	A5	CHAR(10)	S/R	Synchronization ID

Notes:

1. **Set by column.** The following column values indicate which operations write the content of the field into the output:

Value	Condition
S	Save operation set this field.
R	Restore operation set this field.
S/R	Either operation set this field.
(blank)	Not set by either operation. The associated field is set to zero for numeric fields, blank for character fields, or empty for variable-length character fields.

2. **Format of device name.** You can find the first entry using the **Device name offset** field to get to the *Number of device name* field. The **Number of device names** field is not repeated.

Type (in bytes)	Content	Field
BINARY(4)	(blank)	Number of device identifiers

   Then, moving to the first device identifier. Each device identifier consists of a length followed by its name. The device name fields are repeated for each device identifier.

Type (in bytes)	Content	Field
BINARY(4)	S/R	Device name length
CHAR(*)	S/R	Device name

3. **Format of file label.** You can find the start of the file label using the **File label offset** field. The file label fields are not repeated.

Type (in bytes)	Content	Field
BINARY(4)	S/R	File label length
CHAR(*)	S/R	File label

Directory information entries

This table describes the format of the directory entry output for the SAV and RST commands.

The *Entry type* value in the entry header determines if the entry associated with the header is a directory information entry.

For each field, an offset is specified in bytes. This offset is relative to the base address of the entry, or the beginning of the first field in the entry header.

Table 47. Directory information entry output–SAV and RST Commands

Offset (bytes)		Type (in bytes)	Set by[1]	Field
Decimal	**Hex**			
0	0	BINARY(8)	S/R	See the table in Entry header information for more format details.
8	8	BINARY(4)	S/R	Directory identifier offset [2]
12	C	BINARY(4)	S/R	Number of object links processed successfully in directory
16	10	BINARY(4)	S/R	Number of object links processed unsuccessfully in directory
20	14	BINARY(4)	S/R	Starting volume identifier offset [3]
24	18	BINARY(8)	S/R	Total size (in K) of object links processed successfully in directory
32	20	BINARY(4)	R	Number of directory levels created by restore

Notes:

1. **Set by column.** The following column values indicate which operations write the content of the field into the output:

Value	Condition
S	Save operation set this field.
R	Restore operation set this field.
S/R	Either operation set this field.
(blank)	Not set by either operation. The associated field is set to zero for numeric fields, blank for character fields, or empty for variable-length character fields.

2. **Format of directory identifier.** You can find the start of the directory identifier using the **Directory identifier offset** field. The directory identifier consists of a length followed by the directory name. The directory fields are not repeated.

   The CCSID of the directory name can be found by using CCSID of data field from the Command information format.

Type (in bytes)	Content	Field
BINARY(4)	S/R	Directory identifier length
CHAR(*)	S/R	Directory identifier

3. **Format of starting volume identifier.** You can find the first entry using the **Starting volume identifier offset** field. The volume identifier consists of a length followed by the volume name. The volume fields are not repeated.

Type (in bytes)	Content	Field
BINARY(4)	S/R	Starting volume identifier length
CHAR(*)	S/R	Starting volume identifier

Object link information entries

Object link information entries are output in the format described in the table below. The *Entry type* value in the entry header determines if the entry associated with the header is a object link information entry.

The system associates a coded character set identifier (CCSID) with all data including object link names. This association is maintained across all save and restore operations.

For each field, an offset is specified in bytes. This offset is relative to the base address of the entry, or the beginning of the first field in the entry header.

Table 48. Object link information entry–Output from SAV and RST Commands

Offset (bytes)		Type (in bytes)	Set by[1]	Field
Decimal	**Hex**			
0	0	BINARY(8)	S/R	See the table in Entry header information for more format details.
8	8	BINARY(4)	S/R	Object link identifier offset[2]
12	C	BINARY(4)	R	Object link identifier after restore operation offset[3]
16	10	BINARY(4)	S/R	Starting volume identifier offset[4]
20	14	BINARY(4)	S/R	Object link error message replacement identifier offset [5]
24	18	BINARY(4)	S/R	Object link size
28	1C	BINARY(4)	S/R	Object link size multiplier
32	20	BINARY(4)	S/R	ASP at time of save operation
36	24	BINARY(4)	R	ASP after restore operation
40	28	CHAR(10)	S/R	Object link type
50	32	CHAR(8)	S/R	Save active date/time
58	3A	CHAR(10)	S/R	Object link owner at time of save
68	44	CHAR(10)	R	Object link owner after restore
78	4E	CHAR(50)	S/R	Object link text
128	80	CHAR(1)	R	Object link security message
129	81	CHAR(1)	S/R	Object link status
130	82	CHAR(7)	S/R	Object link error message ID
137	89	CHAR(1)	S/R	Object link data
138	8A	BIN(8)	(blank)	Reserved
146	92	CHAR(1)	S/R	Allow checkpoint write
147	93	CHAR(10)	S/R	ASP device name at time of save operation
157	9D	CHAR(10)	R	ASP device name after restore operation
167	A7	CHAR(1)	S	In mounted UDFS
168	A8	CHAR(4)	(blank)	Reserved
172	AC	BINARY(4)	S/R	Journal information required for recovery offset[6]
176	B0	BINARY(4)	S/R	Journal receiver information required for recovery offset[7]
180	B4	BINARY(4)	S/R	Mounted file system information offset[8]
184	B8	BINARY(4)	S/R	Number of private authorities saved
188	BC	BINARY(4)	R	Number of private authorities restored

Notes:

1. **Set by column.** Each value in this column is set when:

Value	Condition
S	Save operation set this field.
R	Restore operation set this field.
S/R	Either operation set this field.
(blank)	Not set by either operation. The associated field is set to zero for numeric fields, blank for character fields, or empty for variable-length character fields.

2. **Format of object link identifier.** You can find the start of the object link identifier using the **Object link identifier offset** field. An object link identifier will consist of a length followed by the object link name. The object link fields are not repeated.

   The CCSID of the object link name can be found by using CCSID of data field from the Command information format.

Type (in bytes)	Content	Field
BINARY(4)	S/R	Object link identifier length
CHAR(*)	S/R	Object link identifier

3. **Format of object link identifier after restore operation.** You can find the start of the object link identifier after the restore operation by using the **Object link identifier after restore operation offset** field. An object link identifier will consist of a length followed by the object link name. The object link identifier fields are not repeated.

   The CCSID of the object link name can be found by using CCSID of data field from the Command information format.

Type (in bytes)	Content	Field
BINARY(4)	S/R	Object link identifier after restore operation length
CHAR(*)	R	Object link identifier after restore operation

4. **Format of starting volume identifier.** You can find the first entry by using the **Starting volume identifier offset** field. The volume identifier consists of a length followed by the volume name. The volume identifier fields are not repeated.

Type (in bytes)	Content	Field
BINARY(4)	S/R	Starting volume identifier length
CHAR(*)	S/R	Starting volume identifier

5. **Format of object link error message replacement identifier.** You can find the start of the object link error message replacement identifier using the **Object link error message replacement identifier offset** field. An object link error message will consist of a length followed by a name. The error message replacement identifier fields are not repeated.

Type (in bytes)	Content	Field
BINARY(4)	S/R	Object link error message replacement identifier length
CHAR(*)	S/R	Object link error message replacement identifier

6. **Format of journal information required for recovery.** You can find the start of the entry by using the **Journal information required for recovery offset** field. Journal information required for recovery will consist of a length followed by the journal path name. The journal fields are not repeated.

The CCSID of the journal receiver path name can be found by using CCSID of data field from the Command information format. For information about converting this name, see the iconv API.

Type (in bytes)	Content	Field
BINARY(4)	S/R	Journal information required for recovery - path name length
CHAR(*)	S/R	Journal information required for recovery - path name

7. **Format of journal receiver information required for recovery.** You can find the start of the entry using the **Journal receiver information required for recovery offset** field. Journal receiver information required for recovery will consist of an ASP device name, a length, and the journal receiver path name. The journal receiver fields are not repeated.

   The CCSID of the journal receiver path name can be found by using CCSID of data field from the Command information format. For information about converting this name, see the iconv API.

Type (in bytes)	Content	Field
CHAR(10)	S/R	Journal receiver information required for recovery - ASP device name
CHAR(2)	(blank)	Reserved
BINARY(4)	S/R	Journal receiver information required for recovery - path name length
CHAR(*)	S/R	Journal receiver information required for recovery - path name

8. **Format of mounted file system information.** You can find the start of the mounted file system information using the **Mounted file system information offset** field. The mounted file system information consists of a length followed by the name.

Type (in bytes)	Content	Field
BINARY(4)	S/R	Mounted file system name length
CHAR(*)	S/R	Mounted file system name

Trailer information entry

The trailer information entry is output in the format described in this topic. The *Entry type* value in the entry header determines if the entry associated with the header is a trailer information entry. The trailer information entry is the last entry in the output created by save (SAV) or restore (RST) commands.

For each field, an offset is specified. This offset is relative to the base address of the entry, or the beginning of the first field in the entry header.

Table 49. Trailer Information entry–Output from SAV and RST Commands

Offset (bytes)		Type (in bytes)	Set by[1]	Field
Decimal	Hex			
0	0	BINARY(8)	S/R	See the table in Entry header information for more format details.
8	8	BINARY(4)	S/R	Volume identifier offset[2]
12	C	BINARY(4)	S/R	Complete data
16	10	BINARY(4)	S/R	Number of object links processed successfully
20	14	BINARY(4)	S/R	Number of object links processed unsuccessfully
24	18	BINARY(8)	S/R	Total size (in K) of object links processed successfully
32	20	BINARY(4)	S/R	Number of media files
36	24	BINARY(4)	S/R	Media file offset[2]

Notes:

1. **Set by column.** The following column values indicate which operations write the content of the field into the output:

   Value **Condition**

   S Save operation set this field.

   R Restore operation set this field.

   S/R Either operation set this field.

   (blank)
   Not set by either operation. The associated field is set to zero for numeric fields, blank for character fields, or empty for variable-length character fields.

2. **Format of volume identifier.** You can find the first entry by using the **Volume name offset** field to get to the **Number of volume identifiers** field. The **Number of volume identifiers** field is not repeated.

Type (in bytes)	Content	Field
BINARY(4)	S/R	Number of volume identifiers

   Then, moving to the first volume identifier. A volume identifier consists of a length followed by the volume name. The **Volume identifier length** and the **Volume identifier** fields are repeated for each volume identifier.

Type (in bytes)	Content	Field
BINARY(4)	S/R	Volume identifier length
CHAR(*)	S/R	Volume identifier

3. **Format of media file.** The media file fields are repeated for each media file.

Type (in bytes)	Content	Field
BINARY(4)	S/R	Media file length
BINARY(4)	S/R	Media file sequence number
BINARY(4)	S/R	Number of media file device names
BINARY(4)	S/R	Media file device name offset
BINARY(4)	S/R	Number of media file volume identifiers
BINARY(4)	S/R	Media file volume identifier offset

4. **Format of media device name.** The media file device name fields are repeated for each media file device name.

Type (in bytes)	Content	Field
BINARY(4)	S/R	Media file device name length
CHAR(*)	S/R	Media file device name

5. **Format of media file volume identifier.** The media volume identifier fields are repeated for each media file volume identifier.

Type (in bytes)	Content	Field
BINARY(4)	S/R	Media file volume identifier length
CHAR(*)	S/R	Media file volume identifier

Output sequence

This table shows the sequence of entries in the output when you specify INFTYPE(*ALL) or INFTYPE(*ERR)

Table 50. Output sequence 1 for SAV and RST commands

Output sequence 1
Command information
Directory information for directory 1 Object link information for object link 1 . . . Object link information for object link N
Directory information for directory 2 Object link information for object link 1 . . . Object link information for object link N
Directory information for directory N Object link information for object link 1 . . . Object link information for object link N
Trailer information

When you specify INFTYPE(*ALL), the output contains an object link entry for all object links (both successful and unsuccessful). When you specify INFTYPE(*ERR), the output contains an object link entry only for unsuccessful links.

The table below shows the sequence of entries in the output when you specify INFTYPE(*SUMMARY):

Table 51. Output sequence 2 for SAV and RST commands

Output sequence 2
Command information
Directory information for directory 1
Directory information for directory 2
Directory information for directory
Trailer information

When you retrieve information from the output format for object links, you must use the entry length that the system returns in the header information format of each entry. The size of each entry might include padding at the end of the entry. If you do not use the entry length, the result might not be valid. The entry length can be used to find the next entry. The trailer entry is always the last entry.

Field descriptions

This information describes possible values for the save (SAV) and restore (RST) output fields.

| **Allow checkpoint write (ALWCKPWRT)**

Indicates whether an object was saved while updates to the object might have occurred. The possible values are:

'0' No updates occurred to the object while the object was being saved.

'1' The object was saved with the SAVACTOPT(*ALWCKPWRT) parameter and the

corresponding system attribute for the object was set. Updates to the object might have occurred while the object was being saved. See "Additional save-while-active option (SAVACTOPT) parameter" on page 135 for more information.

ASP after restore operation
The auxiliary storage pool (ASP) of the object link when it was restored. The possible values are:

1 System ASP

2–32 Basic user ASPs

33–255 Independent ASPs

ASP device name after restore operation
The auxiliary storage pool (ASP) device name of the object link when it was restored. Possible values are:

***SYSBAS**
System and basic auxiliary storage pools

device name
Name of the independent auxiliary storage pool

ASP at time of save operation
The auxiliary storage pool (ASP) of the object link when it was saved. Possible values are:

1 System ASP

2–32 Basic user ASPs

33–255 Independent ASPs

ASP device name at time of save operation
The auxiliary storage pool (ASP) device name of the object link when it was saved. The possible values are:

***SYSBAS**
System and basic auxiliary storage pools

device name
Name of the independent auxiliary storage pool

Command
The command that was used when the operation was performed.

The possible values are:

SAV Save operation

RST Restore operation

Complete data
Indicates whether all of the data for the save or restore operation was in fact saved or restored. This trailer data element can inform you as the completeness of the system description contained in the rest of the output generated by the operation.

The possible values are:

0 The data is not complete. One or more directory information or object link information entries were not written to the user space or byte stream file. This can occur when a user space object link is used and more than 16MB of information about the save or restore operation is generated. This situation occurs only when the save or restore operation processes a very large number of object links. If this situation occurs, you should consider using a stream file to store your output information.

1 The data is complete. All of the information about the save or restore operation is contained in the output.

CCSID of data
The CCSID of the data that is stored in this output.

Data compacted
Indicates whether the data was stored in compacted format.

The possible values are:

'0' The data is not compacted.

'1' The data is compacted.

Data compressed
Indicates whether the data was stored in compressed format.

The possible values are:

'0' The data is not compressed.

'1' The data is compressed.

Device name
The name of a device used to perform the save or restore operation. The filed contains either the name of a device, the name of a media definition or the name of a save file that was used to perform the operation. The length of the name is defined by Device name length and the CCSID is defined by the CCSID of data field.

Device name length
The length of the **Device name** field.

Device name offset
The offset to the field.

Directory identifier
The name of the directory that the object was saved from or where the object was restored.

Directory identifier length
The length of the **Directory identifier** field.

Directory identifier offset
The offset to the **Directory identifier length** field.

End change date
The value that was specified for the end change date when the save operation was performed.

The possible values are:

***ALL** No end change date was specified.

end date
The end change date that was specified on the save operation. The date is in YYMMDD format, is left justified, and is padded with blanks.

End change time
The value that was specified for the end change time when the save operation was performed.

The possible values are:

***ALL** No end change time was specified

end time
The end change time that was specified on the save operation. The time is in HHMMSS format, is left justified, and is padded with blanks.

Entry length
The length of this list entry.

Entry type
Indicates the type of data that is contained in this list entry.

The possible values are:

1 This list entry contains command level information. Use the command information format to map out the data for this list entry.

2 This list entry contains directory-level information. Use the directory information format to map out the data for this list entry.

3 This list entry contains link level information. Use the object link information format to map out the data for this list entry.

4 This list entry contains trailer information. Use the trailer information format to map out the data for this list entry.

Expiration date
The expiration date of the media.

The possible values are:

***PERM**
The data is permanent.

expiration date
The expiration date that was specified on the save operation. The date is in YYMMDD format, is left justified, and is padded with blanks.

File label
The file label of the media file the save or restore operation is using. For a save or restore operation that uses a save file, this field is blank.

File label length
The length of the **File label** field.

File label offset
The offset to the **File label length** field.

Information type
Shows you the type of information that was saved with this operation. (INFTYPE parameter on SAV command).

The possible values are:

'1' Summary information and information about each object link that was processed was saved (*ALL).

'2' Summary information and information about object links that were not successfully saved or restored was saved (*ERR).

'3' Only summary information was saved (*SUMMARY).

In mounted UDFS
Shows whether the object was in a mounted user-defined file system (UDFS) during the save operation.

The possible values are:

'0' The object was not in a mounted UDFS during the save operation.

'1' The object was in a mounted UDFS during the save operation.

Journal information required for recovery offset
The offset to the **Journal information required for recovery - path name length** field. This field will be 0 for objects that were not journaled at the time of the save operation.

Journal information required for recovery - path name
The path name of the journal required to recover the object. The object must be journaled by this journal before an Apply Journaled Changes (APYJRNCHG) can successfully restore the object.

Journal information required for recovery - path name length
The length of the **Journal information required for recovery - path name** field.

Journal receiver information required for recovery offset
The offset to the **Journal receiver information required for recovery - ASP device name** field. This field will be 0 for objects that were not journaled at the time of the save operation.

Journal receiver information required for recovery - ASP device name
The name of the disk pool device that contains the library containing the journal receiver required to recover the object.

Journal receiver information required for recovery - path name
The path name of the first journal receiver in the journal receiver chain needed to recover the object. The object must be journaled to this journal receiver before an Apply Journaled Changes (APYJRNCHG) can successfully restore the object.

Journal receiver information required for recovery - path name length
The length of the **Journal receiver information required for recovery - path name** field.

Media file device name
The name of a device used to perform the save or restore operation. The field contains either the name of a device or the name of a save file that was used to perform the operation. The length of the name is defined by **Media file device name length** and the CCSID is defined by the **CCSID of data field**.

Media file device name length
The length of the **Media file device** name field.

Media file device name offset
The offset to the first **Media file device name** field for this media file.

Media file length
The length of the **Media file** field.

Media file offset
The offset to the first **Media file** field.

Media file sequence number
The sequence number of the media file. The value is 0 if the **Media file device name** is not a tape device.

Media file volume identifier
The name of a volume used during the save or restore operation. The length of the name is defined by the **Media file volume identifier length** and the CCSID is defined by the CCSID of data field.

Media file volume identifier length
The length of the Volume identifier field.

Media file volume identifier offset
The offset to the first Media file volume identifier field for this media file.

| **Mounted file system information offset**
| The offset to the **Mounted file system name length** field. If this field is 0, then either a file system was not mounted over this directory during the save operation or *NONE was specified on the rebuild mounted file system (RBDMFS) parameter on the restore operation.

| **Mounted file system name**
| The name of the file system that was mounted over this directory.

Mounted file system name length
The length of the **Mounted file system name** field.

Number of device names
The number of Device name fields.

Number of directory levels created by restore
When the parent directory of an object being restored does not exist and CRTPRNDIR(*YES) is specified, the restore will create the parent directory. This field will indicate the number of levels of the parent directory that the restore created. For example, if '/a/b/c/stmf' is restored and '/a/b' does not exist, the restore will create '/a/b' and '/a/b/c' and the Number of directory levels created by restore field will be 2.

Number of media files device names
The number of Media file device names contained in this media file.

Number of media volume identifiers
The number of Media file volume identifiers contained in this media file.

Number of media files
The number of Media files processed during the save or restore operation.

Number of object links processed successfully in directory
The number of object links that were successfully saved or restored for this directory.

Number of object links processed unsuccessfully in directory
The number of object links that were not saved nor restored for this directory.

Number of object links processed successfully
The total number of object links successfully saved or restored for the entire save or restore operation.

Number of object links processed unsuccessfully
The total number of object links saved nor restored for the entire save or restore operation.

Number of private authorities restored
The number of private authorities restored for the object.

Number of private authorities saved
The number of private authorities saved with the object.

Number of records
A number interpreted for a given value as follows:

n The number of records saved or restored because a *SAVF device or save file was included among the devices or files saved or restored.

0 The number of records saved or restored because a *SAVF device or save file was not included among the devices or files saved or restored.

Number of volume identifiers
The number of volumes used during the save or restore operation.

Object link data
Indicates whether the data for this object was saved with the object. The possible values are:

'0' The object's description was saved, but the object's data was not saved.

'1' The object's description and the object's data was saved.

Object link error message ID
The message ID of an error message that was issued for this link.

Object link error message replacement identifier
The error message replacement identifier from the link error message.

Object link error message replacement identifier length
The length of the **Object link error message replacement identifier**.

Object link error message replacement identifier offset
The offset to the **Object link error message replacement identifier length** field.

Object link identifier after restore operation
The name of the object link after it was restored.

Object link identifier after restore operation length
The length of the **Object link identifier after restore operation** field.

Object link identifier after restore operation offset
The offset to the **Object link identifier after restore operation length** field.

Object link identifier
For a save operation, the name of the object link that was saved. For a restore operation, the qualified object link name that was saved (including the directory and object link identifier).

Object link identifier length
The length of the **Object link identifier** field.

Object link identifier offset
The offset of the **Object link identifier length** field.

Object link owner after restore
The name of the object link owner's user profile when the object link was restored.

Object link owner at time of save
The name of the object link owner's user profile when the object link was saved.

Object link security message
Indicates whether a security message was issued for this object link during a restore operation.

The possible values are:

'0' No security messages were issued.

'1' One or more security messages were issued.

Object link size
The size of the object link in units of the size multiplier. The true object link size is equal to or smaller than the object link size multiplied by the object link size multiplier.

Object link size multiplier
The value to multiply the object link size by to get the true size. The value is 1 if the object link is smaller than 1 000 000 000 bytes, 1024 if it is between 1 000 000 000 and 4 294 967 295 bytes (inclusive). The value is 4096 if the object link is larger than 4 294 967 295 bytes.

Object link status
Indicates whether the object link was successfully processed.

The possible values are:

'0' The object link was not successfully saved or restored.

'1' The object link was successfully saved or restored.

Object link text
The text description of the object link.

Object link type
The type of the object link.

Participating save operations
The number of save operations that work together to synchronize their data, using the same synchronization ID as this operation.

Private authorities requested
Indicates whether the save operation specified that private authorities should be saved with the objects. The possible values are:

'0' PVTAUT(*NO) was specified.

'1' PVTAUT(*YES) was specified.

Restore date/time
The time at which the object links were restored in system timestamp format. See the Convert Date and Time Format (QWCCVTDT) API for information about converting this timestamp.

Restore system serial number
The serial number of the system on which the restore operation was performed.

Restore release level
The release level of the operating system on which the object links were restored. This field has a VvRrMm format, containing the following:

Vv The character V followed by a 1-character version number

Rr The character R followed by a 1-character release number

Mm The character M followed by a 1-character modification number

Save active
Indicates whether object links were allowed to be updated while they were being saved.

The possible values are:

0 SAVACT(*NO)—Object links were not allowed to be saved while they were in use by another job.

1 SAVACT(*YES)—Object links were allowed to be saved while they were in use by another job. Object links in the save might have reached a checkpoint at different times and might not be in a consistent state in relationship to each other.

-1 SAVACT(*SYNC)—Object links were allowed to be saved while they were in use by another job. All of the object links and all of the directories in the save operation reached a checkpoint together and were saved in a consistent state in relationship to each other.

Save active date/time
The time at which the object link was saved while active in system timestamp format. See the Convert Date and Time Format (QWCCVTDT) API for information on converting this timestamp.

Save active option
Indicates which options were used with save-while-active. The possible values are:

***NONE**
SAVACTOPT(*NONE) was specified. No special save-while-active options were used.

***ALWCKPWRT**
SAVACTOPT(*ALWCKPWRT) was specified. This enabled objects to be saved while they were being updated if the corresponding system attribute was set. Refer to "Additional save-while-active option (SAVACTOPT) parameter" on page 135 for more information.

Save date/time
The time at which the object links were saved in system timestamp format. See the Convert Date and Time Format (QWCCVTDT) API for information about converting this timestamp.

Save release level
The release level of the operating system on which the object links were saved. This field has a VvRrMm format, containing the following:

Vv The character V is followed by a 1-character version number.

Rr The character R is followed by a 1-character release number.

Mm The character M is followed by a 1-character modification number.

Save system serial number
The serial number of the system on which the save operation was performed.

Sequence number
The sequence number of the file on media. The value will be 0 if the save media is not tape. If tape device was not specified for the DEV parameter, this field will be set to 0.

Start change date
The value that was specified for the start change date when the save operation was performed.

The possible values are:

***LASTSAVE**
The save includes object links that have changed since the last time they were saved with UPDHST(*YES) specified on the save operation.

***ALL** No start change date was specified.

Start date
The start change date that was specified on the save operation. The date is in YYMMDD format, is left justified, and is padded with blanks.

Start change time
The value that was specified for the start change time when the save operation was performed.

The possible values are:

***ALL** No start change time was specified.

Start time
The start change time that was specified on the save operation. The time is in HHMMSS format, is left justified, and is padded with blanks.

Starting volume identifier
For an object link, the name of the first volume, on which this object link was saved.

For a directory, the name of the first volume, on which this directory was saved. Saved content can be saved across several volumes.

Starting volume identifier length
For either the starting volume of a directory or an object link, the length of the **Starting volume identifier**.

Starting volume identifier offset
The offset to the **Starting volume identifier length**.

Synchronization ID
The name that was used to synchronize checkpoints for more than one save-while-active operation.

Target release level
The earliest release level of the operating system on which the object links can be restored. This field has a VvRrMm format, containing the following:

Vv The character V is followed by a one-character version number.

Rr The character R is followed by a 1-character release number.

Mm The character M is followed by a 1-character modification number.

Total size (in K) of object links processed successfully
The total size of the object links saved or restored successfully. This field is part of the trailer information entry created when a SAV or RST command is running.

Total size (in K) of object links processed successfully in a directory
The total size of the object links saved or restored successfully in the directory. This field is part of the directory information entry created when a SAV or RST command is running.

Volume identifier
The name of a volume used during the save or restore operation. The length of the name is defined by **Volume identifier length** and the CCSID is defined by the **CCSID of data field**. If a tape drive was not specified for the DEV parameter, this field will be set to 0.

Volume identifier length
The length of the **Volume identifier** field.

Volume identifier offset
The offset to the start of the **Volume identifier length** field.

Related information

Convert Date and Time Format (QWCCVTDT) API

Interpreting output from save commands

This topic contains a list of links to save commands or APIs that you can use to direct output to an output file.

- QSRSAVO - Save object API
- SAVCFG - Save configuration
- SAVCHGOBJ - Save changed objects
- SAVLIB - Save library
- SAVOBJ - Save object
- SAVSAVFDTA - Save save file data
- SAVSECDTA - Save security data
- SAVSYS - Save system
- SAVSYSINF- Save system information

The following topics describe the output information that these commands create. To specify an output file, you must have *CHANGE authority to the database file and *USE authority to the library. The system needs an *EXCLRD lock on the database file. Click the command above that applies to the information that you want to save. The control language (CL) provides descriptions for the three parameters that allow you to direct save output to an output file: File to receive output (OUTFILE), Output member options (OUTMBR), and Type of output information (INFTYPE).

Related reference

"Determining objects that the system saved (save messages)" on page 7
This information describes how save messages work and what information is available from the output files.

Save operation output file information

This table shows the format for the save operation (QASAVOBJ) output file information. Unused fields, fields that are not set, contain a value of zero for numeric fields and blanks for character fields.

Table 52. Save operation (QASAVOBJ) output file information

Identifier	Type	Field
SROCMD	CHAR(10)	Save command
SROINF	CHAR(10)	Information type
SROSYS	CHAR(8)	System
SROSRL	CHAR(6)	Save Release Level

Table 52. Save operation (QASAVOBJ) output file information (continued)

Identifier	Type	Field
SROLIB	CHAR(10)	Library Name
SROASP	ZONED(2)	Library ASP number
SROSAV	ZONED(6)	Objects saved
SROERR	ZONED(6)	Objects not saved
SROSEQ	ZONED(4)	Sequence number
SROLBL	CHAR(17)	File label
SROVOL	CHAR(60)	Volume identifiers
SROSVT	CHAR(13)	Save date/time
SRONAM	CHAR(10)	Object name
SROMNM	CHAR(10)	Member name
SROTYP	CHAR(8)	Object type
SROATT	CHAR(10)	Object attribute
SROSIZ	ZONED(15)	Size
SOOWN	CHAR(10)	Owner
SROSTA	CHAR(1)	Status
SROMSG	CHAR(7)	Error message
SROSWA	CHAR(13)	Save while active date/time
SROTXT	CHAR(50)	Text
SRODEV	CHAR(40)	Device names
SROSVF	CHAR(10)	Save file name
SROSFL	CHAR(10)	Save file library name
SROTRL	CHAR(6)	Target release
SROSTF	CHAR(1)	Storage
SROACP	CHAR(1)	Save access paths
SROSFD	CHAR(1)	Save file data
SROCMP	CHAR(1)	Data compressed
SROCOM	CHAR(1)	Data compacted
SRORFD	CHAR(7)	Reference date
SRORFT	CHAR(6)	Reference time
SROEXP	CHAR(7)	Expiration date
SROXVM	CHAR(390)	Extra volume identifiers
SROPGP	CHAR(10)	Primary group
SROSQ2	ZONED(10)	Large sequence number
SROMIT	CHAR(1)	Objects omitted
SROFMT	CHAR(1)	Save format
SROMFN	ZONED(3)	Media file number
SROTMF	ZONED(3)	Total media files
SROMDN	CHAR(10)	Media definition name
SROMDL	CHAR(10)	Media definition library name
SROVLC	ZONED(3)	Volume count

Table 52. Save operation (QASAVOBJ) output file information (continued)

Identifier	Type	Field
SROVLL	ZONED(3)	Volume length
SROVLD	CHAR(2400)	Volume identifiers (complete)
SROOPT	CHAR(256)	Optical file
SROAS1	CHAR(10)	ASP name
SROAS2	ZONED(5)	ASP number
SROTSZ	PACKED(21)	Total size saved
SROPRT	CHAR(1)	Partial transaction exists
SROJN	CHAR(10)	Journal name
SROJL	CHAR(10)	Journal library name
SROJRN	CHAR(10)	Journal receiver name
SROJRL	CHAR(10)	Journal receiver library name
SROJRA	CHAR(10)	Journal receiver ASP
SROPFL	CHAR(10)	Spooled file name
SROPFN	ZONED(6)	Spooled file number
SROPJB	CHAR(10)	Spooled file job name
SROPUN	CHAR(10)	Spooled file user name
SROPJN	CHAR(6)	Spooled file job number
SROPJS	CHAR(8)	Spooled file job system name
SROPCD	CHAR(7)	Spooled file creation date
SROPCT	CHAR(6)	Spooled file creation time
SROPQN	CHAR(10)	Spooled file output queue name
SROPQL	CHAR(10)	Spooled file output queue library
SROPUD	CHAR(10)	Spooled file user data
SROPFT	CHAR(10)	Spooled file form type
SROPPG	PACKED(11)	Spooled file pages
SROPCP	ZONED(3)	Spooled file copies
SROPSZ	PACKED(15)	Spooled file size
SROPXD	CHAR(7)	Spooled file expiration date
SROPVA	CHAR(1)	Private authorities requested
SROSYN	CHAR(10)	Synchronization ID
SROSYO	ZONED(2)	Participating save operations

Related reference

"Information in output files" on page 158
Most save commands create output that shows what the system saved. Depending on which command you use, you can direct this output to a printer (OUTPUT(*PRINT)), a database file (OUTPUT(*OUTFILE)), a stream file, or a user space.

Related information

Restore operation output file information

Field descriptions

This information describes the fields in the QASAVOBJ (save operation) output file.

ASP name
The auxiliary storage pool (ASP) device name of the object when it was saved. Possible values are:

***SYSBAS**
System and basic auxiliary storage pools

device name
Name of the independent auxiliary storage pool

ASP number
The auxiliary storage pool (ASP) of the object when it was saved. The possible values are:

1 System ASP

2–32 Basic user ASPs

33-255
Independent ASPs

Data compacted
Indicates whether the data was stored in compacted format. The possible values are:

'0' The data is not compacted.

'1' The data is compacted.

Data compressed
Indicates whether the data was stored in compressed format. The possible values are:

'0' The data is not compressed.

'1' The data is compressed.

Device names
The name of the devices used to perform the save or restore operation. The field contains a list of device names. Each device name is CHAR(10) and there can be 1-4 devices listed.

Error message ID
The message ID of an error message that was issued for this object or library.

Expiration date
The expiration date of the media file. The possible values are:

***PERM**
The data is permanent.

expiration date
The expiration date that was specified on the save operation. The date is in CYYMMDD format.

Extra volume identifiers
This field contains a list of extra volume IDs beyond the first 10 volumes. It contains volume names for volumes 11-75. Each entry is CHAR(6). This is a variable-length field.

File label
The file label of the media file used by the save operation. For a save that uses a save file, this field is blank.

Information type
Shows you the type of information that was saved with this operation. (INFTYPE parameter). The possible values are:

***ERR** The list contains information about the command, an entry for each library, and an entry for each object that was not successfully saved

***LIB** The list contains a library entry for each library requested to be saved.

***MBR**
The list contains an entry for each object or, for database files, each member requested to be saved.

***OBJ** The list contains an entry for each object requested to be saved.

Note:

1. The SAVSYS command does not support the INFTYPE parameter. The output contains one record for each media file that is written.
2. The SAVSAVFDTA and SAVSYINF commands do not support the INFTYPE parameter. The output contains one record for the SAVF that is saved.
3. The SAVCFG and SAVSECDTA commands do not support the INFTYPE parameter. The output is type *OBJ.

Journal library name
The name of the library that contains the journal to which the object is journaled.

Journal name
The name of the journal to which the object is journaled.

Journal receiver ASP
The name of the auxiliary storage pool (ASP) that contains the earliest journal receiver needed for applying journal changes when recovering the object.

Journal receiver library name
The name of the library that contains the earliest journal receiver needed for applying journal changes when recovering the object.

Journal receiver name
The name of the earliest journal receiver needed for applying journal changes when recovering the object.

Large sequence number
The sequence number of the file on media. The value will be 0 if the save media is not tape.

Library ASP name
The auxiliary storage pool (ASP) device name of the object when it was saved. Possible values are:

***SYSBAS**
System and basic auxiliary storage pools

device name
Name of the independent auxiliary storage pool

Library ASP number
The auxiliary storage pool (ASP) of the object when it was saved. The possible values are:

1 System ASP

2–32 Basic user ASPs

-1 Independent ASPs. The actual independent ASP number is contained in the **ASP number** field.

Library name
The name of the library that contains the objects that were saved.

Media definition library name
The name of the library that contains the media definition used in the save operation.

Media definition name
The name of the media definition used in the save operation.

Media file number
A number to identify this media file when a library is saved in parallel format. This field is only valid if the **Save format** field is '1' (save format is parallel). The value is 0 if the save media is not tape.

Member name
The name of the database file member that was saved. This field is blank if the object is not a database file, or if INFTYPE(*MBR) was not specified, or if the record is the summary record for the database file.

Object attribute
The attribute of the object that was saved.

Object name
The name of the object that was saved.

Objects not saved
The total number of objects that were not saved for the library.

Objects omitted
Indicates whether any objects were omitted from the save operation. The possible values are:

'0' No objects were omitted from the save operation.

'1' Objects were omitted from the save operation.

Object type
The type of the object.

Objects saved
The total number of objects saved successfully for the library.

Optical file
The name of the optical file used by the save operation. For a save that does not use optical, this field is blank. This is a variable-length field.

Owner
The name of the object owner's user profile when the object was saved.

Partial transaction exists
Indicates whether this object was saved with one or more partial transactions. If you restore an object that was saved with partial transactions, you cannot use the object until you apply or remove journal changes. To apply or remove journal changes, you need the journal identified by the **Journal name** field and the journal receivers starting with the one identified by the **Journal receiver name** field. The possible values are:

'0' The object was saved with no partial transactions.

'1' The object was saved with one or more partial transactions.

Participating save operations
The number of save operations that work together to synchronize their data, using the same synchronization ID as this operation.

Primary group
The name of the primary group for the object that was saved.

Private authorities requested
Indicates whether private authorities were requested to be saved with the objects. The possible values are:

'0' PVTAUT(*NO) was specified.

'1' PVTAUT(*YES) was specified.

Reference date
The value that was specified for the reference date when the save operation was performed. The possible values are:

***SAVLIB**
All changes since the last SAVLIB was specified.

reference date
The reference date that was specified on the save operation. Objects changed since this date are saved. The date is in CYYMMDD format.

Reference time
The value that was specified for the reference time when the save operation was performed. The possible values are:

***NONE**
No reference time was specified

reference time
The reference time that was specified on the save operation. The time is in HHMMSS format.

Save access paths
Indicates whether access paths were requested to be saved during the save operation. The possible values are:

'0' Access paths were not requested to be saved during the save operation.

'1' Access paths were requested to be saved during the save operations.

Save command
The command that was used when the operation was performed. The possible values are:

SAVCFG
Save configuration operation

SAVCHGOBJ
Save changed objects operation

SAVLIB
Save library operation

SAVOBJ
Save object operation

SAVSAVFDTA
Save save file data operation

SAVSECDTA
Save security data operation

SAVSYS
Save system operation

Save date/time
The date and time at which the data was saved. The date and time are in CYYMMDDHHMMSS format.

Save file name
The name of the save file used in the save operation.

Save file data
Indicates whether save file data was requested to be saved during the save operation. The possible values are:

'0' Save file data was not requested to be saved during the save operation.

'1' Save file data was requested to be saved during the save operations.

Save file library name
The name of the library that contains the save file used in the save operation.

Save format
Indicates whether the data was saved in serial or parallel format. The possible values are:

'0' The save format is serial.

'1' The save format is parallel.

Save release level
The release level of the operating system on which the objects were saved. This field has a VvRrMm format, containing the following:

Vv The character V is followed by a 1-character version number.

Rr The character R is followed by a 1-character release number.

Mm The character M is followed by a 1-character modification number.

Save while active date/time
The date and time at which the data was saved while active. The date and time are in CYYMMDDHHMMSS format.

Sequence number
The sequence number of the file on media. This field only contains values between 0 - 9999. If the sequence number is larger than 9999, this field contains a value of -5 and the sequence number value in the **Large sequence number** field should be used. The value is 0 if the save media is not tape.

Size The size of the object.

Spooled file copies
The number of copies of the spooled file.

Spooled file creation date
The date when the spooled file was created.

Spooled file creation time
The time when the spooled file was created.

Spooled file expiration date
The expiration date of the spooled file.

Spooled file form type
The form type of the spooled file.

Spooled file job name
The name of the job that owns the spooled file.

Spooled file job number
The number of the job that owns the spooled file.

Spooled file job system name
The name of the system where the job that owns the spooled file ran.

Spooled file name
The name of the spooled file.

Spooled file number
The number of the spooled file in the job that owns it.

Spooled file output queue library
The name of the output queue library that contained the spooled file.

Spooled file output queue name
The name of the output queue that contained the spooled file

Spooled file pages
The number of pages in the spooled file.

Spooled file size
The size of the spooled file.

Spooled file user data
The user data for the spooled file.

Spooled file user name
The name of the user who owns the spooled file.

Status Indicates whether the object saved successfully. The possible values are:

'0' The object did not save successfully.

'1' The object saved successfully.

Storage
Indicates whether storage was requested to be freed after the save operation. The possible values are:

'0' STG(*KEEP) was specified on the save operation to keep storage for the objects saved.

'1' STG(*FREE) was specified on the save operation to free storage for the objects saved.

Synchronization ID
The name that was used to synchronize checkpoints for more than one save-while-active operation.

System name
The name of the system on which the save operation was performed.

Target Release
The earliest release level of the operating system on which the objects can be restored. This field has a VvRrMm format, containing the following:

Vv The character V is followed by a 1-character version number.

Rr The character R is followed by a 1-character release number.

Mm The character M is followed by a 1-character modification number.

Text The text description of the object.

Total media files
The total number of media files created for a library saved in parallel format. This field is only valid if the **Save format** field is '1' (save format is parallel). The value is 0 if the save media is not tape.

Total size saved
The total size of all of the objects saved for this library.

Volume count
The number of volume identifiers in the **Volume identifiers (complete)** field.

Volume identifiers
The list of volume identifiers that are used during this save operation. The list can contain from one to 10 volumes. If more than 10 volume were used, see the **Extra volume identifiers** field.

Volume identifiers (complete)
The list of volume identifiers that are used during this save operation. The list can contain from one to 75 volumes. See the **Volume count** field to tell how many volume identifiers are in the list. This field is a variable-length field.

Volume length
The length of each volume identifier in the **Volume identifiers (complete)** field.

Retrieving the device name from save completion messages

The CL program retrieves the device name from the CPC3701 message (found in positions 126 through 135 of the message data) and uses the information to determine which device is used by the next save command.

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7

 1.00              PGM
 2.00              DCL        &MSGDATA *CHAR LEN(250)
 3.00              DCL        &MSGID *CHAR LEN(7)
 4.00              DCL        &DEV *CHAR LEN(10)
 5.00              DCL        &DEV1 *CHAR LEN(10) VALUE(TAP01)
 6.00              DCL        &DEV2 *CHAR LEN(10) VALUE(TAP02)
 7.00              SAVLIB     LIB(LIB1) DEV(&DEV1 &DEV2) ENDOPT(*LEAVE)
 8.00  L00P:       RCVMSG     RMV(*NO) MSGDTA(&MSGDATA) MSGID(&MSGID)
 9.00              IF         (&MSGID *NE CPC3701) GOTO L00P /* Compltn */
10.00              CHGVAR     &DEV %SST(&MSGDATA 126 10) /* Device name */
11.00              IF         (&DEV *EQ 'TAP01') DO /* Last was TAP01   */
12.00              CHGVAR     &DEV1 'TAP01' /* Set for first device */
13.00              CHGVAR     &DEV2 'TAP02' /* Set for second device */
14.00              ENDDO      /* Last was TAP01 */
15.00              ELSE       DO /* Last was not TAP01 */
16.00              CHGVAR     &DEV1 'TAP02' /* Set for first device */
17.00              CHGVAR     &DEV2 'TAP01' /* Set for second device */
18.00              ENDDO      /* Last was not TAP01 */
19.00              SAVLIB     LIB(LIB2) DEV(&DEV1 &DEV2) /* Save Lib 2 */
20.00              ENDPGM
```

If any objects cannot be saved, the operation attempts to save remaining objects and sends an escape message (CPF3771 for single libraries, CPF3751/CPF3778 for more than one library, and CPF3701 for save operations to save files) stating how many objects were saved and how many were not. To continue with the next library, the Monitor Message (MONMSG) command must be used to handle the escape condition. The format of the message data for the CPF3771 message is similar to the CPC3701 message and also identifies the last device used.

The SAVCHGOBJ command operates in a similar manner, but uses CPC3704 as a completion message, CPF3774 as an escape message for single libraries, and CPC3721 or CPF3751 for multiple libraries. For save operations to save files, these messages are CPC3723 as a completion message and CPF3702 as an escape message. These messages also contain the last device or save file used in the message data.

Displaying status messages when saving

This program sends a message to the external (*EXT) program message queue if any objects cannot be saved.

```
PGM              /* SAVE SOURCE */
SAVLIB           LIB(SRCLIB) DEV(TAPE01) PRECHK(*YES)
MONMSG           MSGID(CPF0000) EXEC(DO)

SNDPGMMSG        MSG('Objects were not saved - Look at the job +
                 log for messages') TOPGMQ(*EXT)
SNDPGMMSG        MSG('SRCLIB library was not backed up') +
                 TOPGMQ(xxxx)
RETURN
ENDDO
ENDPGM
```

Code license and disclaimer information

IBM grants you a nonexclusive copyright license to use all programming code examples from which you can generate similar function tailored to your own specific needs.

SUBJECT TO ANY STATUTORY WARRANTIES WHICH CANNOT BE EXCLUDED, IBM, ITS PROGRAM DEVELOPERS AND SUPPLIERS MAKE NO WARRANTIES OR CONDITIONS EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT, REGARDING THE PROGRAM OR TECHNICAL SUPPORT, IF ANY.

UNDER NO CIRCUMSTANCES IS IBM, ITS PROGRAM DEVELOPERS OR SUPPLIERS LIABLE FOR ANY OF THE FOLLOWING, EVEN IF INFORMED OF THEIR POSSIBILITY:

1. LOSS OF, OR DAMAGE TO, DATA;
2. DIRECT, SPECIAL, INCIDENTAL, OR INDIRECT DAMAGES, OR FOR ANY ECONOMIC CONSEQUENTIAL DAMAGES; OR
3. LOST PROFITS, BUSINESS, REVENUE, GOODWILL, OR ANTICIPATED SAVINGS.

SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF DIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, SO SOME OR ALL OF THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation

Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, IBM License Agreement for Machine Code, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

This Backing up your system publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of IBM i5/OS.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

AIX
Domino
i5/OS
IBM
IBM (logo)
Integrated Language Environment
Lotus
OS/400
POWER5
POWER6
Redbooks
System i
System i5
System Storage
System x
System/36
Tivoli
WebSphere
z/OS

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Terms and conditions

Permissions for the use of these publications is granted subject to the following terms and conditions.

Personal Use: You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these publications, or any portion thereof, without the express consent of IBM.

Commercial Use: You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED ″AS-IS″ AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Printed in USA