Chapter Twenty One

THE SURPAS SOFTWARE ENVIRONMENT

Introduction

The SuRPAS software environment is supported by a number of utilities and facilities that enable the programmer to be productive while maintaining control over his or her code and associated code modules.

This chapter examines these utilities and facilities, providing an understanding of when and how to use them. These utilities and facilities include:

- The Code Management System (CMS)
- The Nightly Release Process
- The Production Package (FTKPKG)
- The Release Process
- Tape Backup (FTAPE)

Objectives

Programmers will be familiar with and be able to use the utilities associated with the SuRPAS software environment, including:

- The ability to check software in and out using CMS
- An understanding of the nightly release process, retrofitting code, and the release process.
- The tape backup facility and the FTAPE commands

Code Management System (CMS)

Overview

CMS is the homegrown software version control and code movement system of SuRPAS. It handles code module reservation, preventing others from trying to modify the same code in any version of SuRPAS. It is also one of the critical players in the SuRPAS workflow process, working with WebFTK and the email system to help track code for a specific FTK.

This section looks at:

- SuRPAS code and version control
- Access to the Code Management System
- Software retrofits as a part of the Nightly Release Process
- WIPs and DIFFs and how CMS uses these techniques

The Code Management System

The CMS option in SuRPAS Workbench provides access to the SuRPAS Code Mangement System. CMS provides access to file delivery and the software trees. SuRPAS supports 10 trees (blank or zero through 9) with 5 levels per tree. The five levels include the three visible levels (Production, Test, and Development) and two unseen client delivery tree levels (A client and B client).

CMS can be accessed through SuRPAS WorkBench main menu screen or directly from a VMS prompt, as seen in the following example:

```
$ CMS
```

Either of the above options results in the display of the CMS release system screen seen in Figure 21-1 on the next page. The screen is made up of two components; the software trees at the top of the screen; and the selection menu at the bottom. The menu supports ten options:

1. Check OUT a module
2. Check IN a module
3. Remove the WIP or DIFF status of a module
4. Move software
5. Accept an auto-merge
6. Compile a module
7. Obsolete a Module
8. Convert Screens to XOR
9. Software Retrofits

? For Help

Return to exit the screen

Option #1 - reserves this code module under your name (TLA) and place a copy in your directory. No one else can check out this code module (at any release level) until you check this code back in to CMS. The file in the SuRPAS source code directory is renamed with "**_WIP_*tla***" appended to the file extension.

Option #2 - updates and unreserves the code module (after the next nightly release). The code module is placed in a WIP$ directory until the nightly release process moves it to the appropriate directory.

Option #3 - unreserves the code module without updating the code. This occurs immediately. This option also allows the removal of a hold status from a code module.

Option #4 - moves code between software levels and reserves it at that level.

Option #5 - accepts an auto-merge of code modules.

Option #6 - compiles a specified code module and all required include files.

Option #7 - marks a file as obsolete in the CMS database.

Option #8 - converts screen files to XOR format.

Option #9 - implements software retrofitting on a checked-in module.

```
                PFPC Global Fund Services Release System

13.1R   N/A     12.2O   12.2R     12.2a  12.1R  11.2O  11.2R  N/A    ADP

  P       -       P2      P3       P4     P5     P6     P7     -      P9
  T       -       T2      T3       T4     T5     T6     T7     -      T9
  D       -       D2      D3       D4     D5     D6     D7     -      -



  1)  Check OUT a module
  2)  Check IN a module
  3)  Remove WIP or DIF status of a module
  4)  Move software
  5)  Accept an auto-merge
  6)  Compile a module
  7)  Obsolete a module
  8)  Convert screens to XOR
  9)  Software retrofits


  ENTER OPTION (? FOR HELP RETURN TO EXIT):
```

Figure 21-1 - CMS Main Screen

- CMS is available to users based on user permissions as defined in FALLOGIN.COM.

- CMS option 4 moves software from one slot in the software version tree to another slot in that same software version tree

- CMS will do immediate DIFFs as it is moving the software, including older dates over newer dates.

- When CMS is used to check in a file for a particular FTK, a note is made in the notes section in WebFTK. It has the form:

CMS: *filename software sequence*

Code Check-out

CMS Option #1 allows an authorized user to check out code from the controlled source code repository, The user must provide the software version of the software module to be checked out and, of course, the filename and extension. If the file is not already reserved, the user then takes ownership of the file (for all versions) and it is checked out of CMS. The file is automatically copied to the user's default directory and the file in the controlled source directory is renamed with "**_WIP_*tla***" appended to the file extension, where ***tla*** represents your three-letter-acronym and **WIP** is an acronym for Work In Progress.

The Code Management System maintains control of a number of different types of files. When a file is requested, only those file types supported by CMS can be checked out. The following are filetypes that are eligible for checkout:

File Type Extension	**File Description**
.FOR	Fortran Source Code
.COM	DCL Command Procedure
.CFD	SuRPAS Common File Definitions/layouts
.PAR	SuRPAS Parameter Record Definitions/layouts
.CRD	SuRPAS Card Definitions/layouts
.FRM	Instructions to the Fortran Screen Builder
.FDL	File Descriptors
.JDL	Job Descriptors
.FIL	SuRPAS File Definitions
.ERX	SuRPAS Errors with Descriptions

Note that for certain items in the above list, only a record exists or the file itself resides in an OpenVMS library (.OLB or .TLB). In order to accommodate the check out process, the item will be written to the FAL$MISC directory. The file in this directory will be given the appropriate name and extension and reside there until the item is checked in (CMS Option #2).

If the file to be checked out was already reserved at any software version (tree), CMS will not allow that code file to be checked out. If this is the case, CMS will respond to the checkout request with an appropriate message and also the full name of the file including the "**_WIP_*tla***" extension. This provides you with the TLA of the current owner of the checked out file. You can contact the current owner of the file and negotiate a schedule for when the file will be checked back in so you can then check it out.

Code Check-In

When modification to a checked out file is completed, it can be checked back in using CMS Option #2. In order to check a code file in to CMS an authorized user must provide the FTK number along with the file name and extension. All code files must be checked in for a particular project.

If this is a new file (being checked in for the first time) a series of additional questions must be answered, including the software destination, software version tree, and whether or not this code is a software tool.

When checking a file in to CMS be sure to provide appropriate comments describing the changes made to the file. This set of comments will be stored along with the recording of the file update.

The file that is checked in is copied from your default directory to the corresponding WIP directory with "***_tla_#####***" added to the file extension, where ***tla*** is your three-letter acronym and ***#####*** is the FTK number of the project to which this code is associated. The source code file that was checked out and renamed in the source directory retains its "***ext_WIP_tla***" extension. This expanded extension will remain in place until the updated (checked-in) file is processed during the nightly release process or through the use of the SAMEDAY procedure.

The SAMEDAY Procedure

The SAMEDAY procedure allows a user, under very controlled circumstances, to run the nightly release process activities on a particular FTK during the day. This capability may be used when a bug fix needs to be implemented in less than 24 hours or when checked-in code is needed prior to running the nightly release process. SAMEDAY is not accessible from the normal SuRPAS toolset (WorkBench, SuRPAS, Code Management System, etc.). It can only be run from a privileged account. Manual recompile and linking is preferable, along with the appropriate manual retrofitting. SAMEDAY should can be used for FTK packaging to a client. It uses an unusually large amount of system resources (processor intensive) and will impact system access by other users.

Remove CMS Status

CMS Option #3 allows a user to reverse the checking out of a file from the CMS code repository when the user discovers that no code changes are necessary. It can also be used if the result of a DIF during retrofitting leaves the code file with a DIF status attached. In this case if no further coding is required to resolve the DIF, Option 3 can remove the hold status from the code file (see the Retrofit section which follows).

Option #3 requires that you supply not only the file name and extension, but also the software version tree, where the code status is to be reversed. When supplying the file name, enter the original extension name excluding the extra extension information appended during CMS check out (**_WIP_*t7a***). At completion, the filename will revert back to its original name in the source directory.

Code Movement

CMS Option #4 supports the moving of a code file or a series of related code files from one software version tree slot (**D**evelopment or **T**est) to a higher slot in the same software version tree (**T**est or **P**roduction). When moving a code file or a series of related code files it is necessary to provide the FTK number related to this/these file(s).

CMS will prompt you as to whether to attempt to move each of the modules that have been changed as part of the project. You have the option of choosing to manually move any of the modules listed. For each code module selected CMS will display a code difference (DIF) between the destination software and the one you have chosen to move. The DIF is done in real-time as a part of the Option #4 request. Once you have examined the results of the DIF for a specific module you can then confirm your desire to move the code module. You should only move the module if your changes to the code are the only ones that show up in the DIF. Otherwise you need to manually make those changes in the target software to resolve any unexpected DIFs.

Once CMS has prompted you for each module in the FTK that you intend to move, you have now built a move list. The move list is saved to your default directory in the event that the move request is not completed successfully.

CMS will then attempt to move each module in the move list and ask for confirmation. It is possible that during the actual move, you will discover that one or more modules are currently checked out to other users. You can still move the unreserved modules, but you'll need to wait until the other modules are freed up in order to complete the move. This may cause an unstable software environment, depending on what software version level you are moving code to and whether or not this code is going to be used prior to completing the entire move. The modules that did not move successfully still remain in your move list. The next time you access Option #4 you will be prompted to use this move list or review all modules again.

Software Retrofit Screens

Option #9 displays the software retrofit path in any or all of the software version slots in the trees. When option #9 is selected the user can display all of the trees or a specific slot in a tree. Figure 21-2a shows a full tree display. Figure 21-2b displays the retrofit path for a single tree slot.

```
                        - Sw Retrofits -


                  P7 > T7,D7,T6,D6,T5,D5,T4,D4,T3,D3,T2,D2,D
                  P6 > T6,D6,T5,D5,T4,D4,T3,D3,T2,D2,D
                  P5 > T5,D5,T4,D4,T3,D3,T2,D2,D
                  P4 > T4,D4,T3,D3,T2,D2,D
                  P3 > T3,D3,T2,D2,D
                  P2 > T2,D2,D
                  T7 > D7,D6,D5,D4,D3,D2,D
                  T6 > D6,D5,D4,D3,D2,D
                  T5 > D5,D4,D3,D2,D
                  T4 > D4,D3,D2,D
                  T3 > D3,D2,D
                  T2 > D2,D
                  D7 > D6,D5,D4,D3,D2,D
                  D6 > D5,D4,D3,D2,D
                  D5 > D4,D3,D2,D

                             *** MORE ***


   ENTER OPTION (RETURN FOR NEXT PAGE, ENTER SW, X = EXIT, M = MAIN MENU):
```

Figure 21-2a - CMS Full Software Retrofit Screen

```
(ENTER SW, RETURN FOR ALL SW, P = PREV PAGE, X = EXIT, M = MAIN MENU)
   ENTER OPTION: P5

   P5 > T5,D5,T4,D4,T3,D3,T2,D2,D

   ENTER OPTION (ENTER OTHER SW, RETURN FOR ALL SW, X = EXIT, M = MAIN MENU):
```

Figure 21-2b - CMS Single Slot Software Retrofit Screen

The Nightly Release Process

Overview

The nightly release process releases modules checked in to the Code Management System (CMS) during the past 24 hours. This section discusses the functions performed by the Nightly Release Process, including software retrofitting and the handling of DIFs.

Functions of the Nightly Release Process

The nightly release process runs during non-peak development hours. There are a number of functions accomplished during the process. These functions include:

- Verifies that the modules compile cleanly
- Links the modules into the software version tree under which the module is checked in
- Automatically retrofits or moves the modules down the software version tree to the lower (or later, more current) software version slots
- Verifies that the modules will not overlay any changes to lower level software version slots (conflicts must be resolved)
- Saves the original modules in the WIP$ARCHIVE directory for that slot
- Writes a record to the FAL$LIBRARY:HSCMS.DAT history file

Software Retrofits

Software retrofits automate code changes in multiple versions of SuRPAS. The retrofit process built into CMS saves the programmer from what could be a very repetitive task and could lead to the inadvertent introduction of bugs.

Retrofitting happens automatically, across all software libraries, therefore it is required that a particular file only be changed in one software library at a time. Automatic retrofitting only works when the target software looks like your source software did before you changed it. Be aware that your retrofitted code is not explicitly re-tested. It is hoped that the combination of unit test and QA testing will catch any version-specific bugs. Regression testing of each future version will most likely also exercise this change.

If code fails retrofitting at any version level, the programmer will get a "DIFF" email indicating where the code differences are. A DIFF is the result of using the OpenVMS DIFFERENCE utility. The DIFF utility compares two files and provides a line by line set of differences between the two. The programmer will need to manually change the target software (which may have its own retrofits), which may cause more DIFFs, and so on.

Retrofit paths are chosen because of the relationships between the versions of software. If a retrofit occurs at the (P)roduction level, future versions of SuRPAS are retrofitted at the (T)est and (D)evelopment levels (but not the Production level). If a retrofit occurs at the (T)est level, future versions of SuRPAS are retrofitted at the (D)evelopment level only (not at either the Test or Production levels). The software that is a target of a retrofit is a superset of the software that was the source of the retrofit.

How to Retrofit Code

You checked code modules into CMS yesterday, went home thinking all was well, then walked into the office today and you have two messages from the "timer" with the subjects "**`Merge YOURFILE D`**" and "**`Unable to retrofit file: YOURFILE in D`**" sitting in your inbox ("***YOURFILE***" being the name of your code file and "***D***" representing a software version level). Why did you get these? These messages are telling you that the file you were trying to replace in the higher software version level was different than the file in the lower software ***before*** you even checked your file in or out. What could cause that to happen?

Retrofit errors can occur for a variety of reasons, but they most commonly occur as the result of the following scenario: Someone else changes the file at the D software version level, then you try to make a change to the T software version level hoping that your changes will retrofit down to D. CMS recognizes that something is different and does not let you overwrite someone else's changes. Thus you are sent a retrofit error notice.

The following steps address how to retrofit code. These steps assume that code is retrofitted from the lowest software version slot to the higher version slot or the lower software tree to higher software tree. In other words, software version 11.1 is lower than 11.2 and P is lower than T which is lower than D, etc.

Steps For Retrofitting Code

1. Look at the DIFF report. You'll find it in that file's normal directory (FAL$MAIN, FAL$SUB, etc), named ***filename*.FOR_DIF_*tla*_######**, where "***filename***" is the name of your file, "***tla***" is your three-letter-acronym, and "**######**" is the FTK number.

2. It is possible that you received a DIFF solely because of a difference in comments, an extra blank line, or something else in one of the software modules that does not change the functionality of the program. If this is the case, just use CMS Option #3 to remove the DIFF status.

3. The other possibility is that there is a real DIFF. As stated earlier, this usually results from modifications to a later or higher version of code. In this case, the programmer needs to evaluate the differences and decide which modifications to integrate into the final code module. If most of the differences are a result of the new code then it might be better to integrate the prior modifications into the new code. If most of the differences are a result of the prior modifications, then it would be better to integrate the new code into the later software's version. In either case, the procedure is to check out the code from the lowest software tree that had a DIFF and check in the retrofitted version to the same software version level. For instance, you may have DIFFs in Test and Development software levels so you check out the code from T, retrofit your changes from the P software level, and check in the code to the T level again. This will solve the retrofit issue for the lowest software tree, but the next day the programmer may have to perform the same process on one or more of the higher software trees. Each day the programmer may have to continue this until all software version trees are complete without DIFFs.

4. In some cases, CMS will send you mail that there is an “auto-merge” available. You can accept the auto merge from CMS by choosing Option #5 for that module and it is should auto retrofit your change. This is by no means perfect. It is always best to check the code in WIP$MAIN or WIP$SUB, make sure the changes are ok, and make sure it compiles without an error. An automerge will place a note in the routine if the auto merge fails that will show up as a warning or error during compile.

5. When performing a manual retrofit you should include a CMS-prompted comment during the CMS check-in process which contains the word “Retrofit”. Placing this comment in the code can create DIFFs and retrofitting problems itself. **It is good practice to remove this comment from the code in WIP$MAIN/WIP$SUB** especially when working with a routine that is constantly being modified like FAL$MAIN:PAINSMNT.FOR. **This is also good practice when checking in a routine that you never actually modified at all.**

Helpful Hints

Some helpful hints regarding code differences. When retrofitting code, avoid using the comment “retrofit” in the code itself. If you receive a DIFF message and want to replace one software version level’s code with that of another software version level, use the CMS move, Option #4, so that you don’t leave a retrofit message in the target software module. This option works whether you are moving up or down the tree. If you choose to use the CMS automerge, Option #5, make sure that you only merge your changes in the intended code module, not someone else’s, and make sure that no “%DIFF” messages are left in the code (which prevent compilation). This option has recently changed so that it displays the differences between the newly merged code and the old code before accepting the automerge. You should check out the module only if you need to integrate your changes with other changes in the code. In this case, check out the module from the target software, make your changes, and check it in again with a descriptive message (other than “retrofit”). This will help reduce the number of differences between the software modules.

The Release Process

Overview

SuRPAS is updated at the client site twice yearly via releases in the spring and fall. The creation of these releases is the result of an ongoing activity called The Release Process. The Release Process is managed by a project group within the manufacturing department and headed up by Elizabeth (Liz) Mutschler (LEM). This section discusses the concepts and phases of the Release Process.

Release Process Concepts

Projects - FTKs

All software added to the SuRPAS product is accomplished through software ***projects*** and these projects are managed and tracked through the FAL tracking system (***FTK***) and assigned an FTK number.

Projects that represent new enhancements are distributed only as a part of a ***product release*** (also referred to as an ***on-release***). Under normal circumstances there are two releases each year; one in the spring; one in the fall. These releases are packaged on tape and installed at the client site.

Any enhancements made to the product in between the two yearly releases are referred to as ***off-releases*** and are the result of client-specific (billed) requests or internally required updates (tax releases dictated by IRS and regulatory requirements). These releases are either packaged on tape or using the FTKPKG tool and distributed ahead of the twice-a-year scheduled releases.

Production ***"Bug" Fixes*** are the result of bugs found in previously released software and are managed by the Programming/product Support Group (PSG). Bug fix releases are packaged using the FTKPKG tool and are distributed immediately. Bug fixes are prioritized based on the nature of the problem and given a deadline from "as soon as possible" to approximately 8 to 10 days.

New Release

SuRPAS releases are complete packages including all project/FTK software included in the last full release plus all projects/FTKs developed since the last release.

Some clients only take code "on release", that is, they only take code updates twice each year (except for bug fixes). These clients are "Plan A" clients.

Other clients take packages of enhancements between releases. These clients are "Plan B" clients.

Installs

Projects/FTKs code enhancements may contain code and procedures that in some way change the characteristics and/or layout of a database, a database record, or a field in a database record. These changes may also include changes that affect parameter settings or the organization of known SuRPAS files. In order to include this code in SuRPAS, client databases will need to be updated to reflect these characteristic changes without affecting the integrity of the client's database. When a database must be upgraded to support SuRPAS enhancements code to manage this upgrade must be created and tested. This upgrade code is referred to as an ***Install.***

Upgrades

A compilation of installs that move a database from one release level to another.

The Release Committee

The Release Committee oversees the creation and delivery of a specific product release. The Release Committee is made up of members of both the manufacturing group and the delivery group. There is a Release Committee for each version release. The committee is made up of the following roles:

Manufacturing Group

- Team Manager
- Project Manager
- Quality Assurance
- Development

Delivery Group

- Programming/product Support Group
- Hardware Support Group
- Documentation
- Training
- Customer Service

Phases of the Release Process

Building The Code

Code is developed in the D level tree. Modified code is retrofitted to appropriate lower tree levels. All code DIFFs are resolved at all tree levels as soon as possible.

Testers verify installs to maintain the integrity of client databases. Code is tested at all appropriate tree levels. Test plans and regression test plans are updated as needed. Documentors and trainers are notified of any need for additional documentation or end-user training.

The release committee oversees development and testing to guarantee the integrity of the database. It also manages software integrity by making sure that all code is retrofitted to the new release test tree, that there are no missing files from the release, and that no code from inappropriate FTKs is included in the release.

Regression Testing

Regression testing is a three to four week testing period. There are nine basic test categories in regression testing. All code is fully recompiled and linked and all new coding is frozen except for bug fixes.

Release The Code

All code is copied to the appropriate P tree. The release code is copied to tape for delivery to client sites.

Upgrade Clients

Clients are upgraded on a staggered schedule. The release code is installed on the client's T tree. When the client completes acceptance testing, the release code can be upgraded to the P tree.

Release Support

Released code is under warranty through the version warranty period. The warranty period ends two weeks after the last bug is reported. Released software under warranty is supported by the manufacturing group. After the warranty period is over the software release is supported by the Customer Service Group (CSG).

The Production Package (FTKPKG)

Overview

The FTK Package is used to build the software delivery package that is sent to clients when an FTK project is completed.

This section discusses:

- The FTKPKG Application
- The FTKPKG Process

The FTKPKG Application

The FTK Package(FTKPKG) is the tool that is used to build the software production package that is sent to clients when an FTK project is complete. The tool is used when a SuRPAS functionality is to be delivered outside of the normal version release schedule. The package created contains object code, linking instructions, and file layouts, but no program source code.

The combination of WebFTK, CMS, and the internal email system work together to collect all of the module changes and notify participants as project code and project schedules are updated. When CMS is used to check in a file for a particular FTK, or SDMS is used to check in an FTK-related document, a note is made in WebFTK. These notes are automated and can be recognized as they take the form "**CMS:*filename software sequence***".

The WebFTK notes will be used by FTKPKG to determine which modules to recompile and package into the final project delivery media. A manually entered note into the WebFTK technical notes section can have the same effect.

FTK project packaging using the FTKPKG tool is initiated in the WebFTK system. The FTK project members develop and test project code and then check the code into the Code Management System (CMS). The CMS tracked code is then delivered using FTKPKG (off-release delivery) or via release tapes (on-release delivery).

The FTKPKG tool will compile the source files available in the current software to make this package. Object code library's objects are not used. They are recompiled from the source also. Packages are always built from the **P**roduction level of the software version tree.

The FTKPKG Process

When executing the FTKPKG tool you may need to attach the qualifier /WIP. This will be necessary if some of the code modules are located in one of the WIP directories. Files are located in the WIP directories if they have been checked in to CMS but have not yet successfully made it through the nightly release process.

When executing the FTKPKG tool include the FTK number as the only parameter on the command line. The syntax for the FTKPKG command is:

$ FTKPKG #####

where ***#####*** is the FTK number. You will be prompted for each module to include. Answer "Y" for each module to include. Enter an "A" to include all modules affected by an INCLUDE file change. You can then enter any extra module that needs to be included, or just press enter to continue to the next set of prompts.

FTKPKG will then prompt you for processing instructions. The next prompt will require a "Y" to relink all effected main program code. A prompt will request instruction as to whether or not a custom installation is required. Answer "N" to this, under normal circumstances. At the Description prompt, enter a brief description and then Press Enter. Finally, answer "Y" to begin the packaging process.

A delivery form prompt will require you to enter a Printer Queue Name (e.g. HPJET4). Wait for the package to build, and then check for the text "FTKPKG Done" on the screen, to confirm successful completion. Once FTKPKG has completed, retrieve the printed delivery form and obtain the necessary signatures from your manager and the project QA tester to confirm that delivery has been approved. The Program/Product Support Group (PSG) is responsible for the actual delivery of the package to the client. The completed paperwork (also referred to as the TOC - table of contents) must be taken to PSG.

Note: INCLUDE files that are referenced by source code must also be a part of the FTKPKG module list, especially if they have changed as the result of coding the project solution. If the change is being developed off-release, and the include change can cause software issues for other packages, you must check the module out of CMS, and then back in to CMS to create an association between the FTK and the code module.

Note: If you change an Event additional parameters screen (JQE*.FOR routine) you must also include the JBSIDJQE.FOR, JBDISJQL.FOR, and JBXLIST.FOR routines.

Tape Backup (FTAPE)

Overview

The Alpha Cluster and Network includes three tape jukeboxes with access to eleven tape drives. These drives are available to SuRPAS developers for code and database backup. This section discusses these tape drives, the FTAPE commands to manage these tape drives, and the SuRPAS events that provide database backup, restore, and listing functionality.

Using the FTAPE Drives

The FTAPE drives are located in the three robotic jukeboxes located in the Data Center in Pittsburgh. FTAPE allows the user to systematically load and unload tapes for the purposes of backing up or restoring a database to specified slots. Each slot consists of the following information:

Owner: The person responsible for tracking the use of the slot.
Permitted Users: Users, other than the owner, who have access to load/unload the tape.
Protection: The tape is RO (Read-Only) or RW (Read-Write).
Description: Brief description of the information the slot contains.

Viewing the Tape Drives

There are 360 tape slots available to the eleven tape drives connected to the three robot managed, jukeboxes. Each tape holds approximately 40 million blocks. The 360 tape slots are spread over the three jukeboxes as follows:

Slots	Jukeboxes	Tape Drives
000 - 263	Jukebox #1	Tape Drives 5, 7, 9
300 - 347	Jukebox #2	Tape Drives 10, 11, 12, 13
400 - 447	Jukebox #3	Tape Drives 14, 15, 16, 17

Special client-specific tapes use tape drives 18, 19 and 20. Contact HSG for access to these drives

The following is an example of slots, as displayed by the FTAPE LIST command:

Slot	Owner	Prot	Permitted Users	Description
000	TAZ	RO	*	BACKDBTAPE db17 v8.2off Post 12/22
001	TAZ	RO	*	Master BACKDBTAPE v8.2 pre12/22 2/2
002	TAZ	RW	KLK,RAD,JXN,TLO	FTC17770A backup
003	TAZ	RW	KLK,JXN,RAD,KAS,VAP	DB1-FTC_V8.2_AFTER_JS

There are currently three jukeboxes. The slot numbers and tape devices/names are specific to the jukebox.

Slot Range	Tape Drive	Jukebox
000 – 263	Fal_Tape5, Fal_Tape7, Fal_Tape9	Jukebox1
300 – 347	Fal_Tape10, Fal_Tape11, Fal_Tape12, Fal_Tape13	Jukebox2
400 – 447	Fal_Tape14, Fal_Tape15, Fal_Tape16, Fal_Tape17	Jukebox3

FTAPE Commands

The following are FTAPE commands and their descriptions. The commands are executed from the $ prompt. In the commands the **xxx** represents the slot number, the **zzz** represents the user's TLA, and the **n** represents the FTAPE drive number.

FTAPE LIST Displays all slots with the pertinent information. Other list commands include:

- **FTAPE LIST SLOT=xxx** (lists a single slot)
- **FTAPE LIST SEARCH=desription** (searches description field and lists all tapes with a match)
- **FTAPE LIST OWNER=zzz** (lists all tapes for a particular owner)

FTAPE PERMIT xxx "zzz, zzz, zzz" Allows specific users (or all by using an *) access to the tape. Note that the TLA list must be in quotes. Only the owner of the slot can authorize users.

FTAPE LABEL xxx "description" Creates a new description for the specified tape slot. Description must be in quotes. Only the owner and authorized users of the slot can change the description.

FTAPE USERS Displays the users of both jukebox tape drives and network tape drives with the pertinent information.

FTAPE LOAD xxx Takes a tape from the specified slot and inserts it into the first available tape drive. Currently there are 11 drives available across the three jukeboxes:

Jukebox #1: FAL_TAPE5, FAL_TAPE7, and FAL_TAPE9
Jukebox #2: FAL_TAPE10, FAL_TAPE11, FAL_TAPE12 and FAL_TAPE13
Jukebox #3: FAL_TAPE14, FAL_TAPE15, FAL_TAPE16 and FAL_TAPE17

FTAPE LOAD xxx FAL_TAPEn The user can specify a tape drive in which to load and if the drive is not available, the commands ends, thus eliminating the search of drives.

FTAPE UNLOAD FAL_TAPEn Takes a tape from the specified drive and returns it to its original location. The user can only select a tape drive that is currently in use by that user or owner. You can also use the command **FTAPE UNLOAD** to display a list of drives which are loaded with tapes. The command will then ask you to enter the FAL_TAPEn to unload.

FTAPE SWAP FAL_TAPEn xxx Used to take a loaded tape and replace it with one from another slot. Essentially, this does an FTAPE UNLOAD of a slot and then a FTAPE LOAD for the new slot. This is generally used if your backup goes beyond the first tape and you need to load a second tape.

FTAPE PROTECT xxx Allows the owner to protect tapes, RO for read-only or RW for read-write. This does not write protect the tape itself. If you load a read-only tape into a tape drive, FTAPE changes the protection on the tape drive so that you cannot write to the tape drive.

FTAPE INSERT xxx Takes a tape from the input port and inserts it into the specified slot and updates the database appropriately. The destination slot must be empty. If it is not empty, a message will be displayed to that effect and no action will take place.

FTAPE EJECT xxx Takes a tape from the specified slot and ejects it to the jukebox's output port and updates the database accordingly. The destination slot must not be empty. If it is empty, a message will be displayed to that effect and no action will take place.

FTAPE HELP Displays the above commands and a brief description.

FTAPE Notes

All tapes are unloaded automatically at 8:00 pm every night if they are not in use. This automatic unload does not update the FTAPE database which shows the tape is no longer in use. If your tape has been automatically unloaded and you get the message that tape is in use by (user) since (date) then run the following command: **FTAPE UPDATEDB xxx**. The UPDATEDB command will modify the database to mark the specified slot as "unloaded". Be aware that if this command is used inappropriately, it could cause damage to the tape database. Therefore, use this command only if you are certain the tape is unloaded.

If an abend occurs during a backup or restore and the message 'allocated to another user' or ' not software enabled' appears, it means the tapes were not properly dismounted. The steps to correct the problem must be executed in the same OpenVMS session that the abend occurred:

```
$ dismount/unload fal_tapen
$ deallocate fal_tapen
```

The deallocate may tell you that the drive is not allocated, this is not a problem - just ignore it. You can then unload your FTAPE and continue as usual. If the problem continues, then contact the Help Desk.

Using the MULTIBAKUP, MULTILIST, and MULTIRESTR Events

Prior to 1999, you could only back up one database to an FTAPE. This made testing rather slow because at time you would have to change tapes frequently and, as you will find out, the ROBOT is not very fast. However, new events have been created that allow:

- the backup of multiple savesets to a single FTAPE
- the listing of all savesets on an FTAPE
- the restoration of a specific saveset or savesets from an FTAPE

These are all done very quickly and easily using the “MULTI” events.

Backing Up Multiple Savesets to an Ftape

1. Load the desired Ftape* to save to. (Keep in mind the drive that it is loaded into)
2. LNM to the database that you want to back up.
3. Queue up the MULTIBAKUP event from SuRPAS
4. At the additional parameters screen you must enter a saveset description. This identifies your backup as a unique saveset on the tape and you will be able to find it again by entering part or all of that description during a list or restore.
5. The next parameter is Initialize Tape. If you select Y then any other savesets on the tape will be overwritten. Selecting N will just add your saveset to the end of the tape.
6. A normal backup does not back up the directories FAL$LOAD, SFB$LOAD, nor FAL$DATYER, but by placing a Y next to these 3 parameters, you may chose to back them up with your saveset. Be advised that restoring from a saveset that has these directories backed up will delete anything that was in these directories before the saveset. In other words, if you chose to back them up, you do not have the option not to restore them.
7. Hit PF1 and let the event begin. When prompted, enter the FAL_TAPE drive number that your tape is in.
8. When the event completes, a copy of your database will be in a saveset on the Ftape.

Listing All the Savesets on an Ftape

1. Load the Ftape that contains the savesets you wish to list.
2. Queue up the MULTILIST event from SuRPAS
3. At the additional parameters screen you do not need to enter anything and you may hit PF1 to continue, however, if you wish to narrow down your search for savesets on the tape you may enter a User ID, a Client ID, a Database version, and/or a Saveset description. Entering any or all of these will force the event to try and match a specific saveset or savesets.
4. Hit PF1 and let the event begin. When prompted, enter the FAL_TAPE number drive that your tape is in.
5. The event will cycle through the tape and find all matching savesets, tell you how many savesets matched, then output a report containing that information to FAL$REPORT:SAVESETS.RPT. If no matches are found, the event will not abend, but you will be notified that no savesets matched your criteria.

Restoring A Saveset from an Ftape

1. Load the FTAPE containing the saveset you wish to restore.
2. LNM to the database you want to restore into.
3. Queue up the MULTIRESTR event from SuRPAS.
4. At the additional parameters screen you do not need to enter anything and you may hit PF1 to continue, however, if you wish to find the desired saveset more quickly you may enter information into any or all of these fields.
5. Hit PF1 and let the event begin. When prompted, enter the FAL_TAPE drive number that your tape is in.
6. The event will cycle through the tape and if and when it finds a match to your search criteria it will prompt for you to restore. You may choose **Y** to restore, **N** to look for the next match, or EXIT to stop searching (and restore nothing).
7. If the event gets to the end of the tape and no matches were found, the event will not abend but you will be notified that no restore was completed.
8. Once you say **Y** to a saveset, it will begin restoring into that database slot. When the event terminates, you may work in that database as you would after any normal database restore.