# SPANEX™

General Usage Manual

Span Software Consultants Limited

Version: 06.0 Product Number: SPOS-001

Revision: 1st March 2015 Manual Ref: SPX-02-017

CONTENTS Page



# 1 Introduction

This manual describes usage of SPANEX for general purposes, other than specifically for Job Restart and Job Networking and Scheduling. See the SPANEX Restart and Job Networking Guide for full details on these major facilities.

This manual contains general information concerning the SPANEX JCL requirements and options and the use of the other features of SPANEX, and descriptions of the organization of SPANEX programs and macros.

All SPANEX parameter formats and options are described, and details of the effects and implications of each option, and of combinations of options, are given. Implementation details on the SPANEX security and user program features are also contained in this manual.

Maintenance information for SPANEX, such as the Link-Edit requirements for SPANEX modules, will be found in the SPANEX Installation and Maintenance manual. Some examples are given of the use of SPANEX in a standard environment.

The SPANEX Job Restart and Job Networking facilities are covered separately in the SPANEX Restart and Job Networking Guide (manual ref SPX-03). Information on these facilities is provided in this manual where relevant to the use of other parts of SPANEX, but the reader is referred to the SPANEX Restart and Job Networking Guide for full details.

For an extensive introduction to the SPANEX Scheduling and Restart features, the SPANEX Scheduling Beginning User's Guide, manual ref SPX-12, should be used.

| SPANEX Manuals | Order No |
|---|---|
| SPANEX General Usage Manual | SPX-02 |
| SPANEX Restart and Job Networking Guide | SPX-03 |
| SPANEX Scheduling Beginning User's Guide | SPX-12 |
| SPANEX Automated Data Areas Manual | SPX-04 |
| SPANEX Messages and Codes Manual | SPX-05 |
| SPANEX Terminal User's Guide | SPX-07 |
| SPANEX Installation and Maintenance Manual | SPX-09 |
| SPANEX Documentation Index | SPX-10 |
| Span Macros Manual | SPZ-02 |
| Span Service Routines Manual | SPZ-03 |
| SPSMFINF User Manual | SPI-01 |

## 1.1 Basic SPANEX Concepts

SPANEX is a product that provides many facilities in many different areas of system operation. Major functions include an automatic Job Scheduling system, based upon dependent job control within the various jobs of a suite or application system, together with comprehensive calendar-based facilities; a sophisticated Job Restart system providing intelligent restart without manual intervention for the majority of job failures; a means of implementing authorized program functions with all authorization control performed by SPANEX; notification to the system operator of any failures in any job and action based upon the failure that occurred; and many more features.

SPANEX functions as a logical extension of the Operating System Initiator, and provides many additional services to the user program. As a monitor or "front-end" processor, SPANEX detects and intercepts any failure of a user program and takes the appropriate action as specified by parameter. By this technique, SPANEX provides a simple and consistent set of error-reporting mechanisms, and avoids the necessity for wide-ranging and complicated messages and error codes within user programs.

The SPANEX Job Restart system implements the concept of building restart logic into each user Job at the design stage. The logic of the restart of each Job is defined to SPANEX such that a failed Job need only be re-submitted for execution, and recovery programs will be automatically run as necessary, depending upon the point of failure, before SPANEX retries the execution of the Jobstep where the error occurred. This principle removes all the problem analysis load from operations staff and passes responsibility for defining recovery logic to the applications development groups. The SPANEX technique also provides a standardized restart/recovery implementation that can be used throughout an installation, removing any need for customized restart or control systems.

The SPANEX Job Scheduling and Networking facilities extend this restart logic into the area of inter-job dependencies. A SPANEX Job Network can support any complexity of job relationships, with each Job being automatically scheduled by SPANEX as its pre-requisite jobs or events complete successfully. Job suites can be tailored by simple SPANEX utility commands so that different combinations of jobs can be run on different occasions with no difficulty. Using the SPANEX Calendar facilities, completely automatic configuration of jobs can be achieved, supporting any number of job combinations, and an unlimited number of calendars. Various different scheduling control implementations are provided for within the SPANEX system. Job Network control can be performed by an individual user group, each group responsible for its own job suites; or can be the responsibility of an individual time-sharing user via the SPANEX terminal interface; or a specialized group of scheduling staff may be in charge of the scheduling of all job suites or networks in the installation. SPANEX commands are provided for all these organizational methods, and security facilities prevent unauthorized execution of SPANEX utility commands. SPANEX also provides an application program interface that allows user programs to make and implement scheduling decisions.

As an execution monitor, SPANEX provides many further facilities and services to the user program and to the installation. These facilities include secure control of the Operating System Authorized Program Facility, printing of Input/Output and Paging statistics, a "Step Cancel" facility, TSO options, secure control of non-swappability for MVS, and many other features.

## 1.2 Standard SPANEX Features

The following is a list of SPANEX features, not necessarily in any order of importance.

- Automatic Suite/Job/Step Restart facility
- Automatic Job Scheduling and Networking (Dependent Job Control)
- Automatic Calendar processing, to control job selection based on date or “day-type”
- Job Scheduling and Network Operations from Batch, Time-Sharing terminal, MCS operator console or dedicated SPANEX terminal
- support of different techniques for the input of jobs to the system - techniques may vary between Job Networks
- great flexibility in defining variable combinations of jobs required for each execution of an application system
- temporary or one-off changes can be made to production JCL without affecting production JCL libraries
- full-screen real-time display of Job Network status
- full-screen display of jobs selected by Calendar processing for any date in the future or past
- run-time statistics for individual jobs and for application systems
- user-modifiable 3270 Program Function Key support and online HELP data
- wallchart produced showing job inter-relationships
- HALT and PROCEED commands allowing Job Network execution to be interrupted at any point, and resumed when required
- up to eight arbitrary external events which may be defined for each job, to control scheduling according to any user-chosen factors
- ability to “catalogue” frequently-used sets of SPANEX commands and invoke with a single command
- Password protection of sensitive Network control commands
- many optional exit routines to permit Network execution to be monitored or modified and to provide Network security
- all abnormal conditions automatically notified to the operator
- operator acknowledgement of failures can be ensured
- user program exception processing specified by parameter
- multiple criteria for determining the successful or unsuccessful completion of application and utility programs
- batch environment provided for user programs under TSO
- minimal JCL changes or parameter requirements
- extends user EXEC statement “PARM” field to 255 bytes (by use of optional extra DD statement)
- all actions fully documented via optional “Message Log”
- Global Log for a Job Network enables recording of all significant events in the execution of a Job suite with searching by SPANEX command
- “TASKLIB” facility allows TSO “Dynamic STEPLIB”
- “SPANLIB” facility allows easy testing of new SPANEX versions
- secure authorization control for selected programs, batch or TSO
- secure non-swappability control for MVS without PPT updates
- optional MCS console display for information messages
- MCS console for SPANEX messages selectable via Route Code specification

- optional STOP and MODIFY command processing by SPANEX
- optional operator authorization of jobstep execution
- operator can CANCEL with DUMP, even if no Dump DD statement
- “Span Product” facility enhances other Span products and provides facilities for user programs
- aids for testing of TSO Command Processors
- “NOTIFY” option informs TSO users of batch job results (and status of Job Network)
- low overhead - CPU time used printed on “Message LOG” dataset using Span Software's SPSMFINF product (MVS only)
- full paging and EXCP statistics from SPSMFINF (MVS only)
- optional installation exit routine allows tailoring
- macros provided to aid user program development

## 1.3 SPANEX Release Numbering Strategy

Each new Release or Version of SPANEX is identified by means of the SPANEX Version Number, which is printed at the beginning of the SPANEX Message Log, and which appears as a compressed literal string in every SPANEX module.

The structure of this Version Number is that of two numeric digits, a decimal point, and one further numeric digit. An increase in the portion of the Version Number before the decimal point indicates that the format of the SPANEX ICB (the major SPANEX control block) has changed, and that all SPANEX modules have been re-assembled to take account of this change of format. SPANEX will detect any accidental mixing of levels, and will issue an appropriate message and an Abend if an incompatibility is found.

All user-written Exit Routines, and user routines which make use of the #SPXUMSG or #SPXMSG SPANEX macros should be re-assembled when the major SPANEX Version Number changes. Any change to the minor SPANEX Version Number (the digit after the decimal point) does not require any re-assembly of these modules, for the format of SPANEX control blocks is guaranteed unchanged in this situation.

Restart Control Modules for the SPANEX Restart and Job Networking facilities will never need to be re-assembled (except to take advantage of any future new features) because the SPANEX Version Number is included in the RCM and will be checked by SPANEX when the format of the RCM is being analyzed.

## 1.4 Changes for Version 06.0

SPANEX is undergoing a continuous cycle of development and enhancement. The new and improved features of Version 06.0 of SPANEX over the previous Version (05.1) are listed here.

- New options of the LINKAGE parameter of #SPANTRY allow the specification of the storage location of dynamic save areas.
- New CPRIGHT2 option of the #SPZFLD macro for a mixed-case copyright notice value.
- New format option for SPZPARSE fields that allows format-checking of character name values (ie alphabetic first character, followed by alpha-numeric characters).
- All SPANEX modules can now be located above the 16-Megabyte line.

## 1.5 Changes for Version 05.1

SPANEX is undergoing a continuous cycle of development and enhancement. The new and improved features of Version 05.1 of SPANEX over the previous Version (05.0) are listed here.

- Support for z/OS.
- Support for JES job numbers up to J0999999.
- New dump-formatting service routine for printed output (SPZFDUMP).
- Support for 64-bit applications is included in the #SPANTRY, #SPANXIT, #SPAMODE and related macros.
- Support for relative branching, and assembler modules without base registers, is included in the #SPANTRY, #SPANXIT and related macros.
- All software maintenance has been consolidated into the SPANEX code, and many reliability and service improvements have been made throughout the product.
- Minor editorial and technical changes have been made throughout the SPANEX manuals.

## 1.6 Changes for Version 05.0

The new and improved features of Version 05.0 of SPANEX over the previous Version (04.6) are listed here.

- Support for "Year 2000 compliance". All SPANEX reports, displays and date formats now use 4-digit year indications. All scheduling and calendar functions support the running of jobs across the boundary from 1999 to the year 2000.

- SPANEX Restart and Networking features now allow a SPANEX KSDS Catalog to be defined with JCL DD statements. This allows individual users or groups to have their own SPANEX Catalog datasets, if the VSAM KSDS Catalog is implemented when SPANEX is installed.

- SPANEX Quicknet now propagates SPXCAT1 and SPXCAT2 DD statements to submitted jobs, in the same manner as previously supported for TASKLIB and STEPCAT DD statements. This allows the use of user-defined SPANEX VSAM KSDS catalogs, in addition to the optional system-wide catalog specification.

- SPANEX now supports RACF checking of all MVS and JES commands issued from within SPANEX job control features. Authorisation to issue commands can be given to SPANEX without allowing the user assigned to a job to have that authority.

- SPANEX Quicknet now detects and reports Abend codes from all jobsteps, rather than simply reporting that an Abend occurred.

- SPANEX Quicknet retrospective condition code checking now allows specific Abend codes to be interrogated when determining whether a jobstep ended successfully.

- An MVS dump is no longer produced if the "Cancel" option is given to the SPANEX OPT=D jobstep start-up message.

- SPZPARSE now supports TSO- or IDCAMS-style command syntax, as an alternative to the traditional "keyword=value" style. Command and parameter format is dynamically specified via a new parameter of the #SPZPARS macro.

- The features provided in the #SPANTRY and #SPANXIT macros for MVS/ESA and OS/390 programming using the Linkage Stack may now be executed is ASC AR mode.

- New DATE4 field type in the #SPZFLD macro supports for 4-digit year notation for Year 2000 compliance.

- New SYSID field type in the #SPZFLD macro supports inclusion of the SMF System ID in program output.

- New USERID field type in the #SPZFLD macro supports inclusion of the owning user ID in program output.

- New version of the SOB used by SPOUTPUT eliminates all 3-byte addresses from the SPOUTPUT interface. All modules sharing a SOB must be assembled with the same version of the SPANEX macros, but SPOUTPUT supports existing modules compiled with all previous versions of the macros.

- Minor editorial and technical changes have been made throughout the SPANEX manuals.

## 1.7 Changes for Version 04.6

The new and improved features of Version 04.6 of SPANEX over the previous Version (04.5) are listed here.

- SPANEX installation, and all SPANEX macros now support the High-Level Assembler. The High-level Assembler is now the default assembler for the SPANEX installation process (#SPXGEN macro).

- SPANEX Quicknet now propagates STEPCAT DD statements to submitted jobs, in the same manner as previously supported for TASKLIB DD statements. This allows SPANEX restart catalog information to be held in an MVS user catalog.

- Arbitrary internal limits for the number of jobs and jobsteps in a SPANEX Network have been substantially increased. Limits are now 4000 jobs and 16000 jobsteps in a Network. These limits can be increased further if required by means of a small user modification to SPANEX.

- SPZSORT can now sort in-storage records up to 32767 bytes in length.

- New features are provided in the #SPANTRY and #SPANXIT macros for MVS/ESA programming using the Linkage Stack.

- The #SPANTRY macro now support ESA-style module entry and exit linkage, and allows user-defined identification text to be added to a module entry point.

- All SPANEX executable macros for Span Service Routine functions now support user programs running in AR mode on MVS/ESA systems.

- New “Double-underline” feature is provided by SPOUTPUT formatting.

- Minor editorial and technical changes have been made throughout the SPANEX manuals.

## 1.8 Changes for Version 04.5

The new and improved features of Version 04.5 of SPANEX over Version 04.4 are listed here.

- Automatic Calendar feature. A comprehensive implementation is provided of automatic calendar processing within SPANEX. This is integrated with all previous facilities, and can be selected for use on a Job Network basis or on an individual job basis. No changes are required to existing applications.

- The NETSTART command can optionally produce lists of jobs that will be executed and/or not executed for this run of a Job Network. This is used in conjunction with the Calendar feature, so that operators can know which jobs will be run and which omitted.

- "Job Confirmation Sheets", similar to the existing Job Check Sheets, can be produced for hypothetical runs of any Network, in the future or in the past. This is based on SPANEX Calendar definitions, and is available only for Job Networks that have calendar support defined.

- New QUICKNET, QUICKJOB and QUICKSTP macros supercede the earlier macros supported by the SPANEX Quicknet feature, although existing SPANEX Quicknet RCMs need not be altered. Similarly there are new SPXRCM, SPXJOB and SPXSTEP macros to supercede the previous #SPXRCM, #SPXJOB and #SPXSTEP macros, and a new macro, SPXNET, is provided that is a synonym for SPXRCM. These new macros help to make SPANEX RCM definitions more readable, and are easier to enter on some foreign-language 3270-terminal keyboards.

- SPANEX Quicknet now supports Automatic Step Restart, by the use of the new SCANOPT=4 option. This provides many of the facilities of full SPANEX Automatic Restart, but without some of the validation performed by SPANEX if all restart steps are "OPT=M" steps.

- For JCL that is held on a library dataset, the member name of the JCL for a given job may be different from the jobname. There is currently still a limitation that each job must reside in a separate member.

- The support for Procedure Steps in user JCL is improved and completed. There are now no limitations on repeated executions of PROCLIB or in-stream procedures within a SPANEX job, even when full use is made of SPANEX Restart. The User Check exit, SPXUCHEK, is also improved to handle all situations of procedure steps.

- The STATUS command now provides a list of all defined SPANEX Networks, together with their status, if the "NET=/ALL" option is specified. Previously, this was available only if SPANEX Native VSAM KSDS support was used for the Catalog datasets. This feature is not provided for CVOL Catalog datasets.

- There is a new RESET option of the NETSTART command, to reset Job Network execution statistics.

- The new "OPT=Z" SPANEX parameter option allows the SPX000I message (issued by SPANEX at the end of each execution) to be suppressed. This option is supported both for batch and TSO executions of SPANEX.

- Checking of parameters in RCM definitions has been tightened up, and more error messages are given for dubious situations that would previously have gone unnoticed.

- Span Product SPSMFINF is bundled with SPANEX in Release 4.5. SPSMFINF allows SPANEX to report on CPU usage for itself and for user programs, and also to give statistics on EXCPs, Paging and Swapping. SPSMFINF has been improved so that it no longer executes in Supervisor mode for MVS/XA and MVS/ESA SCPs.

- All SPANEX Job Submit Routines that use a JCL library now support the two-tier structure for JCL storage libraries previously introduced with Quicknet. Temporary JCL changes may be made to production jobs without changing the production JCL libraries, and SPANEX will automatically use the changed JCL until it is removed from the temporary library.

- SPANEX cross-CPU integrity is improved by a new technique to force an I/O operation to the "Reserve" volume (see #SPXGEN RESVOL parameter) as part of the serialization process. This closes a small timing loophole where the "Reserve" volume is dedicated to SPANEX serialization, when this volume is not used by SPANEX or other tasks in the system.

- Some of the SPANEX error messages for problems during the use of SPANEX Job Restart are clarified, to aid in user problem determination.

- A further message text option is provided for SPANEX message editing, allowing a longer amount of variable text to be included in "canned" messages.

- The #SPAMODE macro now supports use of the ARMODE facility of MVS/ESA.

- New SCP types (MVS/XA and MVS/ESA) are supported by the #SPANCHK and #SPANTRY macros.

- Assembler H is now the default for the SPANEX installation process (#SPXGEN macro).

- The #SPXGEN macro allows SPANEX to be defined for MVS/ESA systems.

# 2 How to Invoke SPANEX

## 2.1 Organization of SPANEX Parameters

SPANEX, when used as a monitor for the execution of other programs, can be executed as a Batch Jobstep or as a TSO Command Processor. Additionally, the SPANEX Utility function, which is an execution of SPANEX with some special parameters indicating the utility requirements, can be executed as a Batch Jobstep, as a Started Task, as a TSO Command Processor, or, via an interface routine which may need to be supplied by the user, under the control of any Teleprocessing Monitor or time-sharing system.

Parameters are passed to SPANEX via the "PARM" field of the EXEC JCL statement used to invoke SPANEX, or on the TSO command if SPANEX is invoked in TSO foreground or under the control of the batch Terminal Monitor Program (MVS). Parameters to SPANEX can be divided into a number of separate types.

The first type of parameter is the description of the program that SPANEX is being used to invoke. The first positional parameter, if SPANEX is being passed a JCL EXEC-type parameter or a TSO Command Processor parameter, is the name of the program to be executed; this is always required except for some uses of the SPANEX Restart and Job Networking facilities, and is always required for TSO because of restrictions of the TSO command syntax (a dummy program name may be used under TSO for an execution of the SPANEX Utility). Following the program name is a second positional parameter (in batch) which specifies the highest return code that the designated program can issue which should not be recognized by SPANEX as a failure of that program (this value may also be specified via the "ACCRC" keyword operand for batch or for TSO; use of "ACCRC" also allows the specification of an "acceptable abend code"); this parameter is always optional. The "CHKEXIT" parameter in batch or TSO allows the specification of a user routine to check the execution of the program executed by SPANEX; this routine can determine whether or not the execution of the program was successful, and can pass this information to SPANEX for processing. These values are the only parameters that are used to describe the executed program to SPANEX.

The next class of SPANEX parameter is the "Abnormal Termination Action" that SPANEX is required to take if the executed program fails. This class of parameter is recognized by SPANEX as each possible value is a fixed keyword. Possible values are "ABEND", "ABEND=abendcode", "NOTIFY", "NOTIFY=userid", "SETRC=value", "CANCEL", "ACK", "SPXDUMP". The meanings of these various values, and which can be used for batch and which for TSO, are described in the following sections of this chapter. All of these parameters are optional, and SPANEX has a default action which is taken if none of these values is specified. If the executed program fails in batch, SPANEX issues an ABEND with a code of User 4095 with no dump being taken; if the executed program fails under TSO, SPANEX returns to the Terminal Monitor Program with a Return Code of 16 which may be used in conditional statements in a TSO Command Procedure (CLIST).

Another class of SPANEX parameter is the optional SPANEX processing that is required to be performed for this execution. This optional processing can take many forms, and is specified, for both batch and TSO, by means of the "OPT=" SPANEX parameter. This parameter passes to SPANEX a contiguous string of

single-byte character option codes, as described in the “Optional Processing” section of this manual. There are no restrictions as to the combinations of these options that may be requested, although in some circumstances SPANEX may choose to ignore certain options if they are obviously without meaning for the particular situation, or if some options specified are incompatible with others.

There are also some special SPANEX parameters, which include the “NET” or “RCM” parameter for the SPANEX Restart and Job Networking systems, the “TASKLIB” TSO parameter for a “Transient TASKLIB”, and the slash (“/”) which delimits the SPANEX part of the EXEC statement parameter string. Following a slash is placed the EXEC-type parameter which SPANEX is to pass to the executed user program. This user parameter is an optional positional quoted string on the SPANEX TSO command, and immediately follows the user program name.

Although there are many possible combinations of these SPANEX parameters, the parameters required to be used in any particular situation are usually very few and very simple, as the Examples at the end of this manual will show. SPANEX parameters are described in detail in the following sections of this chapter.

## 2.2 JCL for Batch Execution

For batch or started task execution of SPANEX, the program name specified in the "PGM=" parameter of the EXEC statement should specify "SPANEX", and the format of the "PARM=" parameter should be as shown below. The name of the program to be executed by SPANEX should normally be provided, but may be omitted if "OPT=U" is specified, or if "OPT=I" is specified without "OPT=M" (see Restart and Job Networking Guide, manual ref SPX-03, for details).

```
//(stepname)  EXEC  PGM=SPANEX,

                          [      [=Unnnn]]
// PARM='pgmname [,rc] [,ABEND[      ]]  [,NOTIFY[=userid]]
                          [      [=Sxxx ]]  [,N=userid       ]
                          [,CANCEL       ]
                          [,SETRC=rrrr   ]

                          [,ACK]  [,SPXDUMP]

                          [       [nnnn ]]
                          [,ACCRC=[Unnnn]]  [,CHKEXIT=modname]
                          [       [Sxxx ]]

                          [,NET=rcmname]
                          [            ]  [,OPT=xxx]  [/pp]  '
                          [,RCM=rcmname]
```

where

"pgmname" is the load module name of the program to be executed. This parameter may be omitted only in cases where no user program is to be invoked by SPANEX, ie if "OPT=U" or "OPT=UX" is specified, or if Restart Initialization ("OPT=I") is being requested without Restart Monitoring ("OPT=M") - in these cases signify the absence of the "pgmname" parameter with a comma. See the SPANEX Restart and Job Networking Guide (manual ref SPX-03) for more details on the use of options I, M, U and X.

"rc" is the highest condition code from the executed program that does not signify abnormal termination (default is 0). If the code value is specified as 4095, the highest legal condition code from a user program, then all condition code values will be treated as a good end of the user program. This value can alternatively be specified via the "ACCRC" parameter (see below). Note that, for Jobsteps executed with the SPANEX Job Restart or Networking facilities, a value for this acceptable return code can be specified via the SPXSTEP or QUICKSTP macro during the RCM generation process. In this case, any value specified in the EXEC statement parameter overrides the value specified in the RCM. See the SPANEX Restart and Job Networking Guide, manual ref SPX-03, for details of the RCM options.

"ABEND=Unnnn" specifies the User ABEND Code to be used to terminate the Jobstep in the event of abnormal termination of the executed program (default is U4095).

"ABEND=Sxxx" specifies the System ABEND Code to be used to terminate the Jobstep in the event of abnormal termination of the executed program (default is "ABEND=U4095").

"CHKEXIT=modname" specifies the name of a user module that is to be called to check the execution of the program that runs in this jobstep. This "user check exit" facility is required if more than one acceptable abend code is required, or if the combination of an

acceptable abend code and acceptable non-zero condition codes is required. Note that, for Jobsteps executed with the SPANEX Job Restart or Networking facilities, a value for the name of the user check exit module can be specified via the SPXSTEP or QUICKSTP macro during the RCM generation process. In this case, any value specified in the EXEC statement parameter overrides the value specified in the RCM. See the SPANEX Restart and Job Networking Guide, manual ref SPX-03, for details of the RCM options. The definition of the interface to user check exit modules is contained in Chapter 5 of this manual.

"SETRC=rrrr" specifies the step completion code to be issued for the Jobstep in the event of abnormal termination of the executed program (default is "ABEND=U4095").

"userid" is the TSO user ID of the user to be notified by SPANEX of the completion or ABEND of this Jobstep; the abbreviation "N=userid" may be used in place of "NOTIFY=userid"; if "NOTIFY" is specified without a userid, then the console operator is sent a message indicating if a failure occurred, and the ABEND code or return code is passed to the Operating System without modification; the maximum length of userid is 7 bytes for all MVS Operating Systems. Note that a message is sent to the designated TSO user for both normal and abnormal completion of the executed program, so that positive feedback is always received.

"ACK" specifies that the master console operator will be required to acknowledge that he has seen SPANEX error messages in the event of a failure of the executed program; this is ensured by SPANEX by issuing the SPX101A message as a WTOR, and logging the operator's reply on the SPANEX Message Log and in the JCL listing. This option can be a default during the SPANEX installation process (see the SPANEX Installation and Maintenance manual for details).

"SPXDUMP" specifies that a storage dump of the SPANEX major task is required if SPANEX has cause to issue an ABEND for the Jobstep as a result of a failure of the executed program; no dump will normally be produced of the SPANEX task if this option is not specified.

"ACCRC=nnnn" specifies an acceptable condition code value for the user program in this jobstep. This has the same meaning and effect as the "rc" parameter described above, and is included for compatibility. If both the "rc" parameter and the "ACCRC=nnnn" parameter are specified, the "ACCRC=" parameter takes precedence.

"ACCRC=Unnnn" specifies an acceptable user abend code value for the user program in this jobstep. An acceptable user abend code is the single User abend code from this step that does not constitute an error that should be recognized as a failure of the jobstep. For use with the SPANEX Job Restart and Networking features, occurrence of an acceptable abend will not cause suspension of dependent jobs. If an acceptable abend code is specified, and the jobstep does not abend, then all non-zero condition codes are recognized as errors. Multiple conditions for acceptable errors can be implemented by means of a user exit (see the description of the CHKEXIT parameter). Note that, for Jobsteps executed with the SPANEX Job Restart or Networking facilities, a value for an acceptable user abend code can be specified via the SPXSTEP or QUICKSTP macro during the RCM generation process. In this case, any value specified in the EXEC statement parameter

overrides the value specified in the RCM. See the SPANEX Restart and Job Networking Guide, manual ref SPX-03, for details of the RCM options.

"ACCRC=Sxxx" specifies an acceptable system abend code value for the user program in this jobstep. An acceptable system abend code is the single System abend code from this step that does not constitute an error that should be recognized as a failure of the jobstep. For use with the SPANEX Job Restart and Networking features, occurrence of an acceptable abend will not cause suspension of dependent jobs. If an acceptable abend code is specified, and the jobstep does not abend, then all non-zero condition codes are recognized as errors. Multiple conditions for acceptable errors can be implemented by means of a user exit (see the description of the CHKEXIT parameter). Note that, for Jobsteps executed with the SPANEX Job Restart or Networking facilities, a value for an acceptable user abend code can be specified via the SPXSTEP or QUICKSTP macro during the RCM generation process. In this case, any value specified in the EXEC statement parameter overrides the value specified in the RCM. See the SPANEX Restart and Job Networking Guide, manual ref SPX-03, for details of the RCM options.

"xxx" is a string of single-byte option codes specifying the SPANEX optional processing required for this Jobstep (see Section 3 of this manual for a list of valid option codes); the abbreviation "O=xxx" may be used in place of "OPT=xxx".

"pp" is the parameter to be passed to the executed program in "EXEC" JCL statement format (see also description of the "SPXUPARM" DD statement in Section 2.5.4 on page 27 of this manual).

"rcmname" is the load module name of the Restart Control Module for this Job; this need be specified only if "OPT=I" is also specified for a job that is not part of a SPANEX Job Network (SPANEX will know by means of keyed look-up of the SPANEX Catalog what the name of the RCM is for a Network job), but the parameter will be accepted if "OPT=U" is specified (see the SPANEX Restart and Job Networking Guide, manual ref SPX-03). Note that the "NET=" and "RCM=" parameters are synonyms and may be used interchangeably.

NOTE: Abnormal termination is considered to have occurred when the user program either ABENDs with an Abend code other than an Abend specified via the "ACCRC" parameter, or terminates with a return code greater than the value specified in the "rc" field. Abnormal termination is also recognized for a Job controlled by the SPANEX Restart or Job Networking facilities when the Job is cancelled or when an Operating System or Central Processor failure occurs.

## 2.3 SP - The SPANEX TSO Command

The TSO command to execute SPANEX is "SP". The SP command should be issued with the operands described below. The only information that must be provided in the command is the name of the program to be invoked by SPANEX, although a dummy name such as "IEFBR14" may be used if a foreground execution of the SPANEX Utility is to be performed. SPANEX will prompt for a program name if this is omitted.

Syntax:

```
                      [      [(rc)    ]]   [      [(Unnnn)]]
SP pgmname ['pp']     [ACCRC[(Unnnn)]]   [ABEND[(      )]]
                      [      [(Sxxx) ]]   [      [(Sxxx )]]

                      [CHKEXIT(modname)]   [NOTIFY(userid)]

                      [SETRC(rrrr)]    [OPT(xxx)]

                                           [NET(rcmname)     ]
                      [TASKLIB(dsname)]    [                 ]
                                           [RCMNAME(rcmname)]
```

where

"pgmname" is the load module name of the program to be executed.

"pp" is the parameter (enclosed in single quotes) to be passed to the executed program in "EXEC" JCL statement format, or, if OPT(T) is selected, the parameter may be a complete TSO command that is required to be passed to the user program (which must be written as a TSO CP) in a TSO Command Buffer. If OPT(T) is specified and no "pp" value is present, SPANEX will prompt the user for a TSO command (or take the next line of a Command Procedure (CLIST) as the command for the user program). See also description of the "SPXUPARM" DD statement in Section 2.5.4 on page 27 of this manual.

"ACCRC(rc)" specifies the highest condition code from the executed program that does not signify abnormal termination (default is 0). If the code value is specified as 4095, the highest legal condition code from a user program, then all condition code values will be treated as a good end of the user program.

"ACCRC(Unnnn)" specifies an acceptable user abend code value for the executed user program. An acceptable user abend code is the single User abend code from this program that does not constitute an error that should be recognized as a failure of the program. If an acceptable abend code is specified, and the program does not abend, then all non-zero condition codes are recognized as errors. Multiple conditions for acceptable errors can be implemented by means of a user exit (see the description of the CHKEXIT parameter).

"ACCRC=Sxxx" specifies an acceptable system abend code value for the executed user program. An acceptable system abend code is the single System abend code from this program that does not constitute an error that should be recognized as a failure of the program. If an acceptable abend code is specified, and the program does not abend, then all non-zero condition codes are recognized as errors. Multiple conditions for acceptable errors can be

implemented by means of a user exit (see the description of the CHKEXIT parameter).

"Unnnn" is the User ABEND Code to be used to terminate the command in the event of abnormal termination of the executed program (default is "SETRC(16)").

"Sxxx" is the System ABEND Code to be used to terminate the command in the event of abnormal termination of the executed program (default is "SETRC(16)").

"CHKEXIT(modname)" specifies the name of a user module that is to be called to check the execution of the user program. This "user check exit" facility is required if more than one acceptable abend code is required, or if the combination of an acceptable abend code and acceptable non-zero condition codes is required. The definition of the interface to user check exit modules is contained in Chapter 5 of this manual.

"rrrr" is the completion code to be issued for the command in the event of abnormal termination of the executed program (default is "SETRC(16)").

"userid" is the TSO user ID of the user to be notified by SPANEX of the completion or ABEND of this command; the maximum length of userid is 7 bytes.

"xxx" is a string of single-byte option codes specifying the SPANEX optional processing required for this command (see Section 3 of this manual for a list of valid option codes).

"rcmname" is the load module name of the Restart Control Module to be used for a TSO foreground execution of the SPANEX Utility (to be used in conjunction with OPT(U)). Note that the "NET" and "RCMNAME" parameters are synonyms and may be used interchangeably.

NOTE: Abnormal termination is considered to have occurred when the user program either ABENDs with an Abend code other than an Abend specified via the "ACCRC" parameter, or terminates with a return code greater than the value specified in the "rc" field.

As with all TSO Command Processors, any non-ambiguous abbreviation may be used for any of the operands of the "SP" TSO command.

See Section 7.2 on page 57 for examples of the use of SPANEX under TSO.

## 2.4 Using SPXUCHEK, the Standard User Check Exit

Supplied with SPANEX is a standard implementation of the User Check Exit facility, as described in Chapter 5 of this manual. User Check Exits are used by specifying the CHKEXIT parameter to SPANEX, either in batch or for TSO executions. The User Check Exit is invoked to check the execution of the user program running under the control of SPANEX.

The supplied standard exit, SPXUCHEK, is a generalized implementation of a checking function. SPXUCHEK uses control statements read from the SPXUCHKI DD statement to define Condition Code values and/or Abend codes that may be acceptable for this application program or which may indicate that errors have occurred. The format and options of these control statements are described below.

In order to use the SPXUCHEK facility, specify the name of the exit in one of the SPANEX parameter fields: for batch jobs, add "CHKEXIT=SPXUCHEK" to the SPANEX EXEC parm field; for TSO, add "CHKEXIT(SPXUCHEK)" as a parameter of the SP TSO command. For jobsteps using the SPANEX Restart and Job Networking functions, the CHKEXIT parameter of the SPXSTEP and QUICKSTP macros allows the check exit to be specified in the RCM. See the SPANEX Restart and Job Networking Guide for more details of RCM macro parameters. In all cases, the SPXUCHKI DD statement must also be supplied, with one or more control statements as defined below.

### 2.4.1 SPXUCHEK: CONDITION control statement

The CONDITION control statement identifies user program condition code(s) that may indicate either success or failure of the program.

format:

```
            [CODE=nnnn          ]   [,ACCEPT]
CONDITION   [                   ]   [       ]
            [RANGE=(nnnn,nnnn)  ]   [,REJECT]

            [,STEP=stepname]   [,PROCSTEP=procstepname]

            [,SET=nnnn]
```

where:

CODE= - single numeric value, 0-4095
- indicates a specific condition code value from the executed user program that is to be checked for. Either the CODE= parameter or the RANGE= parameter is required for the CONDITION statement.

RANGE= - two numeric values, 0-4095, enclosed in parentheses
- indicates a range of condition code values (specified lower value first, values separated by a comma) that is to be checked for. Any code from the user program within this range (inclusive) will be treated as a match. Either the CODE= parameter or the RANGE= parameter is required for the CONDITION statement.

ACCEPT - fixed keyword value
- indicates that the specified condition code, or range of condition codes, is to be regarded as a successful execution of the user program. The jobstep will be flagged as "good end" for scheduling and restart purposes.

REJECT - fixed keyword value
- indicates that the specified condition code, or range of condition codes, is to be regarded as a failure of the user program. The jobstep will be flagged as failed for scheduling and restart purposes, and the "abnormal termination action", as specified in the SPANEX parameters, will be taken.

STEP= - name of a jobstep
- specifies the stepname of a jobstep for which this CONDITION control statement applies. If the jobstep being checked does not have this name, the control statement will be ignored. This allows the same control statement stream to be used for checking multiple jobsteps, and is also required to identify individual jobsteps when the SPXUCHEK exit is being used for Retrospective Condition Code checking.

PROCSTEP= - name of a procedure step
- specifies the procedure stepname of a jobstep for which this CONDITION control statement applies. This is used as further qualification of the jobstep name specified by the STEP= parameter, in the case where there are duplicate stepnames in the job within catalogued or in-stream JCL procedures. If the jobstep being checked does not have the name specified by the STEP= parameter, and is not within a JCL procedure with the procedure stepname specified by this parameter, the control statement will be ignored. This allows the same control statement stream to be used for checking multiple jobsteps and multiple procedure steps, and is also required to identify specific instances jobsteps when the SPXUCHEK exit is being used for Retrospective Condition Code checking.

SET= - single numeric value, 0-4095
- specifies a condition code to be used for a successful completion of the user program. If the user program ends with a condition code included in this statement (CODE= or RANGE= parameter), then the jobstep completion code will be changed to the value specified by the SET= parameter. This allows a single "good" condition code to be generated for programs that may produce varying condition codes for a successful completion. This function is not applicable to Retrospective Condition Code checking or to failing jobsteps.

### 2.4.2 SPXUCHEK: ABEND control statement

The ABEND control statement identifies user program abend code(s) that may indicate either success or failure of the program.

format:

```
              [Unnnn]    [,ACCEPT]
ABEND    CODE=[Sxxx ]    [       ]
              [ANY  ]    [,REJECT]

         [,STEP=stepname]  [,PROCSTEP=procstepname]

         [,SET=nnnn]
```

where:

CODE= - valid Abend code, or the keyword ANY
- indicates a specific abend code value from the executed user program that is to be checked for. The abend code may be a single User or System abend code, or the keyword "ANY", indicating any user or system abend.

ACCEPT - fixed keyword value
- indicates that the specified abend code, or any abend (if ANY is specified for CODE=), is to be regarded as a successful execution of the user program. The jobstep will be flagged as "good end" for scheduling and restart purposes.

REJECT - fixed keyword value
- indicates that the specified abend code, or any abend (if ANY is specified for CODE=), is to be regarded as a failure of the user program. The jobstep will be flagged as failed for scheduling and restart purposes, and the "abnormal termination action", as specified in the SPANEX parameters, will be taken.

STEP= - name of a jobstep
- specifies the stepname of a jobstep for which this ABEND control statement applies. If the jobstep being checked does not have this name, the control statement will be ignored. This allows the same control statement stream to be used for checking multiple jobsteps, and is also required to identify individual jobsteps when the SPXUCHEK exit is being used for Retrospective Condition Code checking.

PROCSTEP= - name of a procedure step
- specifies the procedure stepname of a jobstep for which this ABEND control statement applies. This is used as further qualification of the jobstep name specified by the STEP= parameter, in the case where there are duplicate stepnames in the job within catalogued or in-stream JCL procedures. If the jobstep being checked does not have the name specified by the STEP= parameter, and is not within a JCL procedure with the procedure stepname specified by this parameter, the control statement will be ignored. This allows the

same control statement stream to be used for checking multiple jobsteps and multiple procedure steps, and is also required to identify specific instances jobsteps when the SPXUCHEK exit is being used for Retrospective Condition Code checking.

SET= - single numeric value, 0-4095
- specifies a condition code to be used for a completion of the user program with the Abend code specified by this statement. If the user program Abends with the Abend code included in this statement (CODE= parameter), then the Abend will be over-ridden by SPANEX, and the jobstep completion code will be changed to the value specified by the SET= parameter. This allows a single "good" condition code to be generated for programs that may produce varying condition codes or Abend codes for a successful completion. This function is not applicable to Retrospective Condition Code checking or to failing jobsteps.

### 2.4.3 SPXUCHEK Processing of control statements

The control statements for SPXUCHEK are processed sequentially as they are read. The first control statement which matches the completion of the user program will be used to determine the success or failure of the jobstep. Subsequent control statements will be ignored. This technique allows any combination of condition and/or abend codes to be handled.

For example, a user program always produces a condition code zero if it has worked successfully, and produces a range of different error condition codes according to different types of failure. However, the installation determines that, although most codes must be treated as errors, condition code 20 may be treated as a good completion in this case. The SPXUCHEK statements to support this would be:

```
CONDITION     CODE=20,ACCEPT
CONDITION     RANGE=(1,4095),REJECT
```

When used for Retrospective Condition Code checking, the SPXUCHEK exit will be invoked separately for each jobstep in the job that is to be checked. Therefore, the control statements may be read more than once. The STEP= and PROCSTEP= parameters are used to ensure there is no confusion over statements that apply to specific jobsteps only. If the STEP= parameter is specified, it must match the stepname being checked before any action is taken. The PROCSTEP= parameter is used to qualify the stepname if the same step appears more than once in different procedure steps within the job.

## 2.5 Optional DD statements for SPANEX

There are no mandatory DD statements for SPANEX (except for some functions of the SPANEX Utility and some Job Scheduling routines), but there are some optional DD statements which may be used to take advantage of some SPANEX facilities. The "TASKLIB" and "SPANLIB" DD statements both refer to load libraries, "SPXPRINT" defines the optional SPANEX Message Log dataset, "SPXUPARM" defines the optional EXEC statement parameter extension dataset, "SPANSNAP" defines an optional dump dataset to be used for dumping the user task(s) in the event of an Operator CANCEL of the job. See below for a description of these DD statements.

Other optional DD statements include "SPXRSCTL", "SPXNETCS" and "SPXUTLIB" which are used by the SPANEX Utility for control statement input, SPANEX Job Check Sheet output, and for access to the optional SPANEX Utility catalogued command library, respectively, "RCMAPOUT" and "WALLCHRT" which are report datasets for the RCM Map facility, "SPXGLOG" which refers to the SPANEX Job Networking facility Global Log dataset (Dynamic Allocation is optional for this dataset under MVS), and various DD statements used by some of the supplied SPANEX sample Job Submit routines for the Job Networking facility.

### 2.5.1 TASKLIB DD Statement

If a "TASKLIB" DD statement is supplied in a SPANEX Jobstep or for a SPANEX TSO foreground execution, then the library (or library concatenation) specified on this DD statement is used as a task library for the user program subtask of SPANEX (this facility is not available if OPT=C is specified). This task library replaces any "JOBLIB" or "STEPLIB" specification. If the first load module of the user program is not found in the library or libraries specified then the JOBLIB/STEPLIB/Linklist is searched, but the JOBLIB/STEPLIB will not be searched for any further modules that the user program may dynamically invoke. The "TASKLIB" parameter of the SPANEX TSO command may also be used and this will replace (for the duration of the TSO command only) any dataset allocated to the TASKLIB DD name. The TASKLIB DD statement is used in the first instance to search for any specified "user check exit" modules.

If the SPANEX Restart or Job Networking facilities are in use, then the "TASKLIB" DD statement is also used in the first instance for the Restart Control Module, for User Exit routine load modules, for Submit Routine load modules, and for the load modules associated with the SPANEX RCM Map facility; if these load modules are not found in the task library, then the JOBLIB/STEPLIB/Linklist will be searched.

### 2.5.2 SPANLIB DD Statement

If a “SPANLIB” DD statement is supplied in a SPANEX Jobstep or for a SPANEX TSO foreground execution, then the library or library concatenation specified on this DD statement is used to obtain further SPANEX modules (after the first) only. It is not used for any user modules or for any of the modules that may be obtained via the “TASKLIB” DD statement. The main use of this DD statement is for the testing of SPANEX itself, particularly in TSO foreground.

### 2.5.3 SPXPRINT DD Statement

The “SPXPRINT” DD statement must refer to a sequential output dataset, and is used to print the optional SPANEX Message (activity) Log. Many useful SPANEX messages appear only on this Log, and this DD statement should be supplied if any unexpected results are received or if, for example, data on the results of SPANEX optional processing is required, or if information on the CPU utilization of the user program or of SPANEX itself is required (available via Span Software's Product SPSMFINF). The “SPXPRINT” DD statement is required for executions of the SPANEX Utility (OPT “U”).

### 2.5.4 SPXUPARM DD Statement

The “SPXUPARM” DD statement, if specified, must refer to a sequential input dataset (or member of a Partitioned Dataset) with Fixed or Fixed-Blocked 80-byte records. The function of this is to extend the length of parameter that is to be passed to the SPANEX user program to a maximum of 255 bytes. Any user parameter information specified in the SPANEX EXEC statement PARM field will be passed to the user program first, with SPXUPARM information concatenated directly to it. The normal half-word parameter length prefix will contain the combined length of all this data when the user program first receives control from SPANEX. The format of input cards in the SPXUPARM dataset is as follows: All data in columns 1-71 (including any leading blanks) is used, a non-blank character in column 72 specifies that the following card contains continuation of the data; after all continuations have been processed, parameter data is truncated after the last non-blank character. Comment cards may be input after the last continuation card - these will be printed on the SPANEX Message Log.

### 2.5.5 SPXUCHKI DD Statement

The “SPXUCHKI” DD statement is used by the standard User Check Exit, SPXUCHEK, as described earlier in this chapter. The SPXUCHKI DD statement defines a control statement input dataset, of fixed 80-byte records, each record being a valid control statement for the SPXUCHEK exit routine.

### 2.5.6 SPANSNAP DD Statement

If a “SPANSNAP” DD statement is supplied in a SPANEX Jobstep, then, under certain circumstances, SPANEX will dump user task(s) to this dataset. The current implementation of SPANEX uses this dataset in the event of an operator CANCEL of the Job, or if the Jobstep is terminated because of CPU time limits being exceeded, in order to produce a dump of the user's areas which do not appear in any system-provided dump. A dump may also be written to this dataset if the Modify command option (OPT=F) is in effect, by means of an operator “SNAP” command (see Section 6 on page 53 of this manual).

### 2.5.7 SPXRSCTL DD Statement

The “SPXRSCTL” DD statement is fully described in the SPANEX Restart and Job Networking Guide, manual ref SPX-03, under the heading “SPANEX Utility”.

### 2.5.8 SPXNETCS DD Statement

The “SPXNETCS” DD statement is fully described in the SPANEX Restart and Job Networking Guide, manual ref SPX-03, under the descriptions of the NETSTART and CSHEET Utility commands.

### 2.5.9 SPXUTLIB DD Statement

The “SPXUTLIB” DD statement is fully described in the SPANEX Restart and Job Networking Guide, manual ref SPX-03, under the description of the INPUT Utility command.

### 2.5.10 SPXUTMST DD Statement

The “SPXUTMST” DD statement is fully described in the SPANEX Restart and Job Networking Guide, manual ref SPX-03, under the description of the INPUT Utility command.

### 2.5.11 SPXUTPAN DD Statement

The “SPXUTPAN” DD statement is fully described in the SPANEX Restart and Job Networking Guide, manual ref SPX-03, under the description of the INPUT Utility command.

### 2.5.12 RCMAPOUT DD Statement

The “RCMAPOUT” DD statement is fully described in the SPANEX Restart and Job Networking Guide, manual ref SPX-03, under the heading RCM Map Implementation.

### 2.5.13 WALLCHRT DD Statement

The “WALLCHRT” DD statement is fully described in the SPANEX Restart and Job Networking Guide, manual ref SPX-03, under the heading RCM Map Implementation.

### 2.5.14 JOBPDS/TEMPPDS DD Statements

These DD statements are used by certain SPANEX Job Submit Routines, which are fully described in the SPANEX Restart and Job Networking Guide, manual ref SPX-03.

### 2.5.15 SPXCAT1/SPXCAT2 DD Statements

These DD statements may be used to define user-specific SPANEX KSDS Catalog datasets, as described in the SPANEX Restart and Job Networking Guide, manual ref SPX-03.

### 2.5.16 PANVALET/TEMPPANV DD Statements

These DD statements are used by certain SPANEX Job Submit Routines, which are fully described in the SPANEX Restart and Job Networking Guide, manual ref SPX-03.

### 2.5.17 LIBRMAST/LIBRTEMP DD Statements

These DD statements are used by certain SPANEX Job Submit Routines, which are fully described in the SPANEX Restart and Job Networking Guide, manual ref SPX-03.

This page intentionally left blank.

# 3 SPANEX Optional Processing

| Option | Batch | TSO | CP | Function |
|---|---|---|---|---|
| A | Y | Y | Y | Requests APF Authorization for user program |
| B | N | Y | Y | Requests bypass SPANEX ESTAI/STAI processing |
| C | Y | Y | Y | Do not create a subtask for user program |
| D | Y | N | N | Requests WTOR before user program invoked |
| F | Y | N | N | Requests accept MODIFY (F) OS commands |
| H | Y | N | N | Specifies information messages to console |
| I | Y | N | N | Step Restart Initialization option |
| J | Y | N | N | Requests Job-Step task for user program |
| L | Y | Y | Y | Requests SPSMFINF statistics (MVS only) |
| M | Y | N | N | Step Restart Monitoring option |
| N | Y | N | N | Requests Jobstep non-swappable (MVS only) |
| P | Y | N | N | Requests accept STOP (P) OS commands |
| R | Y | N | N | Permit Retry of user program on request |
| S | Y | Y | Y | Invoke user program as Span Product |
| T | N | N | Y | Invoke user program as TSO CP |
| U | Y | Y | Y | Invoke SPANEX Utility, not user program |
| X | Y | Y | Y | External access to SPANEX Utility |
| Z | Y | Y | Y | Suppress SPX000I message at SPANEX termination |

## 3.1 OPT=A - Request APF Authorization

Option "A" requests that the user program be executed with an Authorized Program Facility (APF) Authorization Code of "1". The user program does not need to be Link-Edited with the AC(1) option. The "OPT=A" option is available to Batch or TSO programs and must be specified if APF authorization is required. If the "OPT=A" option is not specified the Jobstep will be executed non-authorized, even if the user program is Link-Edited with the AC(1) Linkage Editor option. In order for OPT=A to take effect, the name of the user program being executed must appear in the appropriate SPANEX Authorized Program Name Table - (SPXM0APB for batch execution, SPXM0APF for TSO foreground execution). If this is not done, the Jobstep or TSO command will be executed non-authorized. Note that the Operating System requirements for including any user program libraries in the APF library list are not affected unless the appropriate bit is set in the SPANEX Authorized Program Name Table which permits the authorized execution of modules from non-authorized libraries; in this case, only the first module need not appear in an authorized library, the fetching of second and subsequent modules is as standard in the Operating System. To overcome these restrictions the Span Product facility is available. See Section 4.3 on page 40 in this manual on the maintenance of the SPANEX Authorized Program Name Tables, and note the possible security exposure if certain options of this facility are used without care. Note also that if the user program is fetched from a non-authorized library, Register 0 on entry to the user program will contain the CDE address for the user program load module which should not be found from the normal Operating System control block chains.

## 3.2 OPT=B - Bypass SPANEX ESTAI/STAI

Option "B" requests that SPANEX bypass ESTAI/STAI processing for the user program (TSO only). This should be specified if it is required to enter the TSO "TEST" facility to debug an ABEND of the user program running under SPANEX, or to obtain a system dump of a SPANEX user program. If "OPT=B" is not specified, SPANEX will intercept any ABEND of the first user program task, and will cause the removal of all debugging information in order to perform normal SPANEX processing.

## 3.3 OPT=C - Do Not ATTACH User Program

Option "C" requests that the user program be executed within the major SPANEX task. This option is necessary, for example, if the user program uses OS Checkpoint/Restart. The user program is normally ATTACHed by SPANEX as a subtask - if "OPT=C" is specified, SPANEX will invoke the user program by means of a LINK. This option should be used only when it is necessary, as it may restrict some of the recovery features of SPANEX, and will prevent certain SPANEX features from being used.

## 3.4 OPT=D - WTOR before execution

Option “D” requests that SPANEX issue messages (SPX065I, SPX066A) to the operator before invoking the user program, requesting him to decide whether or not the user program should be executed. The operator has the option of causing an ABEND, bypassing the execution of the Jobstep, CANCELing the Job, or executing the program. The purpose of this is to allow hands-on testing of Jobs or to confirm the execution of sensitive Jobsteps.

## 3.5 OPT=F - Accept MODIFY Commands

Option “F” is the option that permits SPANEX operator commands to be issued by the console operator and accepted by SPANEX. These commands are issued to SPANEX by means of the MODIFY (F) OS console command. If “OPT=F” is specified, then SPANEX will create and enqueue its own CSCB (Command Scheduling Control Block) so that user program MODIFY commands (if used) are not affected. The name of the SPANEX CSCB that must be used for MODIFY commands is the name of the user program if this is unique in the system, or a name constructed by SPANEX; the CSCB name is notified to the operator by message SPX063I at SPANEX initialization time - this message is non-deletable on DIDOCS consoles and will be displayed in response to an “*I R” command on JES3 consoles. See Section 6 on page 53 of this manual (SPANEX Operator Commands) for further information on the use of the commands provided by the “OPT=F” facility.

## 3.6 OPT=H - Hardcopy to Console

Option “H” requests that all WTO messages issued by SPANEX during the course of execution of the user program are to be issued to the Master Console (Routcde=2). Without “OPT=H” specified, many SPANEX WTO messages are sent to ROUTCDE=11 only (Write-to-Programmer messages) and so appear only in the JCL listing for the Job.

## 3.7 OPT=I - Restart Initialization

For users of the SPANEX Restart and Job Networking facilities, Option “I” is used on the Restart Initialization step of each job to process the restart status of the job, to ensure that all pre-requisite jobs have successfully completed execution, and to determine the step at which job execution is to begin. For further details see the SPANEX Restart and Job Networking Guide, manual ref SPX-03.

## 3.8 OPT=J - Request Job-Step task for user program

Option “J” requests that the user program be executed as a job-step task under SPANEX. This function allows for the unusual requirement of a program to execute without being aware of any external code or modules residing in its Address Space. For the vast majority of user programs, this option will have no apparent effect. Note, however, that programs executed with the “OPT=J” option do not have access to SPANEX facilities or the use of SPANEX macros such as #SPXQ, #SPXRSTU, #SPXUMSG. Attempts to use any of these macros will produce an error return code, or sometimes an abend of the user program. Option J has a further function for special programs, in that a non-standard storage protect key can be assigned by SPANEX to the user program task if the appropriate requirements are met. In order for this function of OPT=J to take effect, the name of the user program being executed must appear in the SPXM0APB SPANEX Authorized Program Name Table (for batch execution only - this function is not supported under TSO). The entry for the program name must be flagged as requiring a special protect key, and the value of the key must also be specified in the table. If this is not done, the Jobstep or TSO command will be executed with the standard protect key. Note that a special protect key is an authorized function, and the Operating System requirements for including any user program libraries in the APF library list are not affected. See Section 4.3 on page 40 in this manual on the maintenance of the SPANEX Authorized Program Name Tables. Option “J” takes precedence over option “C”, if both are specified for the same jobstep. Appendix A in this manual contains a list of standard programs that are known to require the use of OPT=J when run under SPANEX.

## 3.9 OPT=L - SPSMFINF Statistics

For use of the SPSMFINF Span Software Product, which is included with SPANEX, option “L” is used to request that full statistics for the execution of the user program be printed on the SPANEX Message Log. Information printed includes Total EXCP count, Paging counts (including VIO and Swapping), and total Address Space Swap count. This feature is particularly informative for TSO commands and programs. Note that the SPSMFINF product supports only the MVS Operating System (all versions), and that this option cannot be used for other systems.

## 3.10 OPT=M - Restart Monitoring

For users of the SPANEX Restart and Job Networking facilities, Option “M” is used on each step of the job which is to be monitored by the SPANEX Restart Processor. Each “OPT=M” step must be defined in the Restart Control Module for the job. For further details see the SPANEX Restart and Job Networking Guide, manual ref SPX-03.

## 3.11 OPT=N - Mark Jobstep Non-Swappable

Option “N” requests (for an MVS system only) that the user program be executed in an Address Space that is non-swappable. This option is available to batch programs only and must be specified if non-swappability is required. The name of the user program does not need to be specified in the MVS Program Properties Table. In order for this option to take effect, however, the name of the user program being executed must appear in the appropriate SPANEX Authorized Program Name Table (SPXM0APN). If this is not done, the Address Space will be swappable.

## 3.12 OPT=P - Accept STOP Commands

Option “P” is the SPANEX “Step Cancel” option, and requests that SPANEX accept console STOP (P) commands to this job. If the system operator issues a “STOP jobname” command, SPANEX will immediately terminate the jobstep by taking the Abnormal Termination action as specified in the SPANEX parameter. This can be used to effect an operator CANCEL of a jobstep without cancelling the entire job. If it is wished to specify the means of terminating the jobstep in response to an operator command, then “OPT=F”, the MODIFY command option, should be used.

## 3.13 OPT=R - Permit Retry of user program

Option “R”, used in combination with option “F” (accept operator Modify commands) specifies that the user program executed in this batch jobstep is eligible for operator-requested retry. If option “R” is included with option “F”, the operator may issue an OS Modify command for this SPANEX jobstep requesting the “RETRY” option. The user program will then be immediately terminated by SPANEX, and then re-invoked to begin a new execution within the same jobstep.

## 3.14 OPT=S - Span Product

Option “S” requests that the user program be executed as a Span Program Product. See Section 5.1 on page 47 of this manual on the writing of Span Products (which may be performed by user installations as well as by Span Software Consultants!). Note that if the user Span Product Program is fetched from an authorized load library, Register 0 on entry to the user program will contain the user program CDE address which should not be found from the normal Operating System control block chains. Option “S” should always be specified when running under SPANEX user programs that use the #SPZSOB macro to invoke the Span Software SPOUTPUT service routine.

## 3.15 OPT=T - TSO Command Processor

Option “T” requests that the user program be executed as a TSO Command Processor. This option can be implemented only if SPANEX itself is a TSO CP (ie not invoked via a “CALL” TSO command but entered directly from the TSO Terminal Monitor Program (the batch TMP is supported) or under the SPF TSO command option). When using option “T”, SPANEX must be provided with the TSO command that is required to be passed to the user program when it is invoked; either this can be specified on the “SPANEX” or “SP” command used to invoke SPANEX by enclosing the whole user command in quotes as if it were a problem program parameter, or, if the parameter value is omitted, SPANEX will prompt for a command for the user program. If the SPANEX command is embedded in a TSO Command Procedure (or CLIST) then the next statement in the CLIST will be taken by SPANEX to be the user program command if the parameter value is omitted from the SPANEX command. The user program must be written as a TSO Command Processor if option “T” is used, or 0Cx ABENDs in the user program will probably result.

## 3.16 OPT=U - SPANEX Utility

For users of the SPANEX Restart and Job Networking facilities, Option “U” is used to execute the SPANEX Utility. This utility has a wide range of commands which are used to display or modify the Restart Status of jobs without the necessity of executing these jobs, and is used for all manual control and interrogation of a SPANEX Job Network. For further details see the SPANEX Restart and Job Networking Guide, manual ref SPX-03. The SPANEX Utility can also be executed using one of the supplied SPANEX Extended TP support routines, and use of these is documented in the SPANEX Terminal User's Guide, manual ref SPX-07.

## 3.17 OPT=X - External Access to SPANEX Utility

Option “X” permits the SPANEX user program to execute as the SPANEX Utility, and permits non-standard input/output routines to be used for the SPANEX Utility. This function is used internally for the support by SPANEX of terminals other than TSO terminals or MCS consoles. Option “X” can also be used to permit user programs to call all commands of the SPANEX Utility, but this requires a considerable amount of user coding and a product to make use of this facility may be announced by Span Software Consultants at a future date.

## 3.18 OPT=Z - Suppress SPX000I message at SPANEX termination

Option “Z” causes SPANEX to suppress the SPX000I message issued after a normal termination of the user program run by SPANEX. Sometimes, particularly when invoked within TSO or ISPF CLISTs, it is desirable to remove this message, as it can cause the user terminal screen to be reformatted and require an additional user keystroke during the execution of the CLIST. Option “Z” has no other effect, and does not alter the logical flow of SPANEX processing.

This page intentionally left blank.

# 4 SPANEX Installation Procedures

Note that the definitive information about the installation of SPANEX can be found in the SPANEX Installation and Maintenance manual.

## 4.1 SPANEX SVC

SPANEX requires the SPANEX SVC to be installed for the majority of its functions. This SVC is Type 3 for all MVS Operating Systems. For an existing SPANEX installation implementing a new release of SPANEX all that is required is to replace the SVC load module with the new SVC load module supplied with the SPANEX release. Otherwise a new SVC number must be allocated for the SPANEX SVC (Type 3 as required for MVS).

The #SPXGEN macro (see the SPANEX Installation and Maintenance manual) generates JCL to assemble and link-edit source module SPXM0050, and to include the SVC number in the #SPXGLOB source member, from where it can be obtained by all SPANEX functions. If the SVC is Type 2, the SPXMSVC0 source member of the SPANEX source library will also be assembled via the #SPXGEN macro in order to produce an SVC routine with the correct CSECT name, and then the Operating System Nucleus must be link-edited to include the SPANEX SVC. SPANEX can be generated specifying an SVC number of 000, and this will disable any SPANEX function that requires the use of the SPANEX SVC - this should not be attempted for a production version of SPANEX as many integrity-checking features are implemented by means of the SVC.

## 4.2 System Control Program

SPANEX requires to be generated for the System Control Program (Operating System, SCP) under which it is to be run. The SPANEX release tape contains object modules for the MVS Operating System only. When the SPANEX generation process is performed using the #SPXGEN macro, the “ASSEM=ALL” parameter must be specified if the SCP is not MVS, in order to ensure that all SPANEX modules are compatible with the SCP in use. If a change of SCP is to be undertaken by an existing SPANEX installation then it is recommended that all SPANEX modules are assembled for the new SCP (by using the “ASSEM=ALL” parameter of the #SPXGEN macro).

At SPANEX installation time, the SCP for which SPANEX is to be used must be specified to the #SPXGEN macro. Currently valid SCP names are MVS, MVS/ESA, MVS/XA, VS1. All SPANEX modules must be link-edited after assemblies are performed to effect a change of SCP.

## 4.3 Authorized Program Name Tables

SPANEX provides four tables of program names of programs which are eligible for special facilities when run under SPANEX. These tables are all found in source module SPXM0APF and may be included either in the "SPANEX" load module or as a separate load module "SPXM0APT" (the latter arrangement is recommended for ease of updating). In either case the tables may be updated by means of source changes followed by assembly and link-edit as appropriate, or by means of SUPERZAP to CSECT SPXM0APF. For a SPANEX system where this module is not included in the "SPANEX" load module, SPANEX will attempt to load "SPXM0APT" when checking of Authorization Name Tables is required. An ABEND will occur if this module either is not found or is found on a program library that is not APF authorized. Thus, full security is preserved to the limit of that provided by the Operating System. The isolation of the SPANEX Authorization Name Tables into a separate load module also permits other customer programs to access data contained within the tables. An assembler mapping macro (#SPXAUTH) is provided with the SPANEX system to aid the development of such user programs, and a further macro (#SPXAFLG) is provided which provides flag settings for the module attribute flags in these tables.

The remainder of this discussion assumes that the tables are provided as a separate load module SPXM0APT, which also permits easier updating and avoids the contents of the tables being visible in SPANEX storage dumps.

SPXM0APF Table of program names that may be executed with APF authorization in TSO foreground. This authorization is provided only if the user specifies OPT(A).

Format of each entry (10 bytes long)

| | | |
|---|---|---|
| 0-7 | Load module name as invoked by SPANEX | |
| 8-9 | Attribute flags: | |
| | X'8000' | Module may come from a non-APF library |
| | X'4000' | Module must be executed as a TSO CP |
| | X'2000' | Module must not be a TSO CP |

Note that the first entry in the SPXM0APF table is a dummy entry for identification purposes only and is not used by SPANEX when checking module names.

SPXM0APB Table of program names that may be executed with APF authorization in background (batch). This authorization is provided only if the user specifies OPT=A.

Format of each entry (10 bytes long)

| | | |
|---|---|---|
| 0-7 | Load module name as invoked by SPANEX | |
| 8 | Attribute flags: | |
| | X'80' | Module may come from a non-APF library |
| | X'10' | Module requires special storage protect key |
| 9 | X'F0' | Protect key value (high-order four bits of byte, allowed values 0-F) |

Note that the first entry in the SPXM0APB table is a dummy entry for identification purposes only and is not used by SPANEX when checking module names.

SPXM0APS Table of program names that may be executed as Span Products. Any program may be run as a SPANEX-defined Span Product if "OPT=S" is specified in the SPANEX parameter. This feature allows the user program to take advantage of SPANEX message facilities, etc. However, the program name must appear in this table if any SPANEX authorized functions (eg use of the SPANEX SVC) are to be used. This authorization is provided only if the user specifies OPT=S.

Format of each entry (10 bytes long)

0-7 Load module name as invoked by SPANEX

8-9 Attribute flags:

| | |
|---|---|
| X'8000' | Module may come from a non-APF library |
| X'4000' | Module must be executed as a TSO CP |
| X'2000' | Module must not be a TSO CP |
| X'6000' | Module must not be run under TSO |
| X'1000' | Do not issue log messages to record use of SPANEX SVC authorization functions |

Note that the first entry in the SPXM0APS table is a dummy entry for identification purposes only and is not used by SPANEX when checking module names.

SPXM0APN Table of program names that may be executed non-swappable (batch only, MVS only). The Address Space will be set non-swappable by SPANEX only if "OPT=N" is specified in the SPANEX parameter.

Format of each entry (10 bytes long)

0-7 Load module name as invoked by SPANEX

8-9 Attribute flags:

| | |
|---|---|
| X'8000' | Module may come from a non-APF library |

Note that the first entry in the SPXM0APN table is a dummy entry for identification purposes only and is not used by SPANEX when checking module names.

Warning: There is a security exposure if the option is used which permits setting of APF authority or of Span Product authority for modules which do not reside on an APF authorized library. It is recommended that if this facility is used it is controlled in some way. For example, a SPANEX Installation Exit routine could be implemented to ensure that programs executed in this way are fetched from the correct library or that modules loaded are indeed the ones that the system is expecting. If all modules are required by these tables to reside in an APF authorized library, then there is no security loophole whatsoever introduced by SPANEX into the system. Any security problems discovered to be introduced by SPANEX may be the subject of an error report to Span Software Consultants Limited, and a correction will be provided.

## 4.4 Installation Exit Routine

SPANEX has a facility to allow the user installation to tailor the functions and security aspects of SPANEX by means of an optional Installation Exit Routine. If used, this routine must be link-edited with the SPANEX load module "SPXM0050", and is entered twice for each SPANEX execution of a user program, once immediately before and once immediately after the invocation of the problem program. For SPANEX executions that process only the SPANEX Utility, or for which invocation of the user program is bypassed for any reason, only the second entry to the installation exit is made (ie flag SPXF4TRM will be set at the only call to the exit - see below). The installation exit routine must have a CSECT name of SPXUSERX.

Parameter list to installation exit routine:

Register 1 ➜ A(SPANEX Internal Control Block)

The routine should include the #SPXICB macro to generate the DSECTs for SPANEX control blocks, and must have addressability to the ICB (USING SPXICB,Rn) before issuing any SPANEX macros. In addition to this macro, a COPY statement must be included at the beginning of the assembly for "#SPXGLOB". All available information pertaining to the user program being executed and to the program environment is available from the ICB. Flag values that should be examined include flag SPXF4TRM in byte SPXFLG04 (which, when set, indicates that this is the second entry to the exit routine, after the successful or unsuccessful completion of the executed user program; this flag is also set at the only entry to the exit routine if execution of the user program is bypassed for any reason), and flag SPXF6ACT in byte SPXFLG06 (which, if set, indicates that the executed user program has abended in a SPANEX jobstep running with SPANEX OPT=C; otherwise the second entry to the exit routine is not made in cases where the SPANEX control program abends). The exit routine may issue any SPANEX macro and any system macro (including ABEND).

A sample installation exit routine is provided for MVS users, which sets the SMF program name for a SPANEX Jobstep to the name of the program being executed by SPANEX (instead of being "SPANEX" for all SPANEX steps). This routine is named SPXUSERX on the distributed SPANEX source library, and is offered as an example only and is not warranted in any way as being fit for any particular purpose.

## 4.5 SPANEX Load Modules and Macros

SPANEX Load Modules should be installed on a system "Link List" library to permit universal access and to protect against illegal updates. SPANEX load modules are: SPANEX (alias SP, SPXM0000), SPXM0050, SPXM0130, SPXM0140, SPXM0150, SPXM0310 (alias SPXRCCC0), SPXMSVCR, SPXRCMAP, SPXUCHEK. In addition, many sample Job Network Submit Routines and a number of SPANEX Extended TP Support modules are supplied with the SPANEX release, and these must be available as separate load modules if they are to be used; sample submit routines are named SPXNJSnn or SPXQCKnn, and Extended TP modules are named SPXFxxxx. SPANEX System Calendar modules are named SPXCALnn, and User Calendar modules may be named SPXxxxnn, where "xxx" is any 3-character string. Link-edit attributes must be as specified in the module table in Section 3 of the SPANEX Installation

and Maintenance Manual (manual ref SPX-03) or the results may be unpredictable (Link-edit JCL is generated in full by the #SPXGEN macro). Note that all SPANEX Load Modules that have the "RENT" (reentrant) attribute are candidates for the Link Pack Area for all Operating Systems.

All SPANEX macros should also be installed on a system-wide or generally-available macro library, as these are used whenever any SPANEX modules are assembled and whenever any Restart Control Module (RCM) is generated. Also, many of these macros may be found useful for normal user program development - they are fully documented in the Span Macros Manual.

## 4.6 SPANEX RESERVE Volume Serial Number

The #SPXGEN macro has the facility for specifying the Volume Serial Number of a disk volume for which a RESERVE can be issued to ensure serialization in a shared DASD environment, for the use of the SPANEX Restart and Job Networking systems. The implications of the use of this parameter are fully explored in the discussion of the "RESVOL=" parameter of the #SPXGEN macro in the SPANEX Installation and Maintenance manual. The default value of this parameter is "000000" which specifies that there is to be no SPANEX cross-CPU serialization. If the SPANEX Catalog is a CVOL Catalog, this value should be set as the Volume Serial Number of the volume containing the CVOL Catalog, as this will also be used to build the parameter lists for accessing the Catalog. The value of this parameter must be allowed to default to "000000" only if (1) there is no shared DASD in the installation; and (2) user jobs or TSO LOGON procedures do not use VSAM or ICF Catalogs via JOBCAT or STEPCAT JCL statements or the SPANEX Catalog is a native VSAM KSDS. SPANEX will not function correctly with STEPCATs unless the SPANEX Catalog is a CVOL or a native VSAM KSDS. If the SPANEX Catalog is a CVOL, STEPCATs in user jobs are supported if the volume serial number of the Catalog volume is specified via the RESVOL parameter of the #SPXGEN macro, and the #SPXGEN parameter CATALOG=CVOL is specified. The SPXM0050 module must be re-assembled and link-edited if the value of the RESERVE Volume Serial Number is changed.

## 4.7 SPANEX RACF Group and User IDs (symbols &#RACGR and &#RACUS)

As part of its implementation of job control, SPANEX may at times issue MVS and/or JES commands to the system. If the user installation has implemented RACF authority checking for the issuers of system commands, then a RACF Group and Userid must be nominated to be given authority for commands for use by SPANEX. The names of the RACF Group and User are specified on the #SPXGEN macro. Appropriate facilities of RACF (or of an equivalent security product) must be used to grant authority for all MVS and JES job control commands to this Userid. It is not necessary for this Userid to have the ability to log on to the system. SPANEX will issue all commands using the authority of this User and Group, in order to avoid any necessity for giving command authority to individual users of the SPANEX job scheduling and control features.

## 4.8 SPANEX MCS Route Codes

The #SPXGEN macro provides the facility for user installations to specify the MCS Console Route Codes that are to be used for SPANEX WTO/WTOR messages. Four variables in the #SPXGLOB source member are used for this purpose, and they are specified (via the "MCSROUT=" parameter of the #SPXGEN macro) in two groups of two: &#URTE1 and &#URTE2 (MCSROUT sub-parameters 1 and 2) both specify Route Codes to be used for messages issued by the SPANEX Utility when it is being operated from an MCS console; &#URTE3 and &#URTE4 (MCSROUT sub-parameters 3 and 4) specify Route Codes to be used for all other SPANEX WTO/WTOR messages issued by Jobs using the SPANEX Restart and Job Networking facilities (these codes will be used in addition to Route Code 2). Each of these variables must be assigned a single numeric value in the range 1-16 inclusive, corresponding to an MCS Route Code. This permits an installation to define unique Route Codes for SPANEX messages or to dedicate an MCS console to use by the SPANEX Utility. The SPANEX system as distributed has all four of the variables assigned the value 2, and Route Code 2 indicates the master console in a standard MCS environment. If any of these variables is changed, the SPXM0050 module must be re-assembled and link-edited.

## 4.9 Non-English-Language SPANEX Messages

In a standard SPANEX system all SPANEX messages are issued in English. However, the capability is available for SPANEX messages to be translated into any other language that may be required (indeed, any re-wording in English may be undertaken if so desired). It is the responsibility of the user installation to perform this translation. All SPANEX messages are held in the two SPANEX message modules, source members SPXM0070 and SPXM0140. These may be changed in any way provided the following rules are obeyed: (a) the length of each message may be increased or decreased as necessary; (b) SPANEX message numbers are fixed and cannot be altered; (c) the basic sense of each message should be preserved; (d) variable data is included in messages by SPANEX by means of fixed character strings as defined in Section 5.4 of this manual - these strings should be left unchanged in each message, although their position in each message may be altered. After changing the text of messages, modules SPXM0070 and SPXM0140 must be re-assembled and link-edited into the SPANEX load modules.

## 4.10 Elimination of undesired SPANEX Messages

Individual SPANEX messages may be completely disabled if they are never to be required. For example, the messages SPX107I and SPX108I concerned with switching of APF authority by authorized programs may be of no interest. In order to prevent a particular message being issued by SPANEX, the appropriate message table (SPXM0070 or SPXM0140) must be updated and then re-assembled and link-edited into the SPANEX load modules. The message in question should be altered in the source of the message table so that it is immediately followed by the parameter ",(NOUT)". Care should be taken that important messages are not disabled in this way.

This page intentionally left blank.

# 5 Using SPANEX Advanced Features

## 5.1 Writing SPANEX "Span Products"

SPANEX provides the facility for user programs of running as a "Span Product". This facility was primarily developed to enable Span Software products to be written in a uniform and compatible manner, but is available to any SPANEX user program. Two levels of Span Product are supported, authorized and non-authorized. A "Span Product" is authorized if the name of the program is included in the SPXM0APS table (see Section 4.3 on page 40 of this manual) and if the program conforms to the designated standards according to any flag settings in that table. An authorized Span Product program is permitted to issue the #SPXSVC macro; authorization is not required for the other SPANEX facilities.

When a program is requested to be executed by SPANEX as a Span Product (by specifying the parameter "OPT=S"), SPANEX passes to the user program a parameter list including the address of the SPANEX ICB (Internal Control Block). Provided that the user program has addressability to the SPANEX ICB, SPANEX macros may be used to call up any of the SPANEX functions described below.

Parameter format for Span Product program on initial entry:

a) if OPT=S specified,
R1 ➜ A(normal EXEC parm)
X'80',AL3(ICB)

b) if OPT=S not specified,
R1 ➜ X'80',AL3(normal EXEC parm)

Note: For a TSO foreground execution where OPT(T) is specified (as well as OPT(S)), R1 -> CPPL, where the CPPL is as passed to any TSO Command Processor but extended by one fullword, where this fullword contains the address of the SPANEX ICB.

The user program must include the #SPXICB macro to generate the DSECTs for SPANEX control blocks, must include a COPY statement for #SPXGLOB at the beginning of the module, and must have addressability to the ICB (USING SPXICB,Rn) before issuing any SPANEX macros.

## 5.2 Writing SPANEX "User Check Exit" Routines

SPANEX User Check Exit routines are modules that may be called by SPANEX to check the execution (ie success or failure) of a user program executed by SPANEX or within a SPANEX job.

SPANEX invokes a User Check Exit routine when this is requested by means of the CHKEXIT parameter in the SPANEX JCL PARM field or specified on the SP TSO command. For users of the SPANEX Restart and Job Networking facility, there is a CHKEXIT option on the SPXSTEP macro used to define a jobstep to

SPANEX, and the User Check Exit can be called either within the step where the user program is executed, or as part of the Retrospective Condition Code checking function.

The exit routine must be written in assembler language, must include the #SPXICB macro to generate the DSECTs for SPANEX control blocks, must include a COPY statement for #SPXGLOB at the beginning of the module, and must have addressability to the ICB (USING SPXICB,Rn) before issuing any SPANEX macros. The exit routine must be link-edited as a separate load module, and must be available to SPANEX from the JOBLIB/STEPLIB, or from the Operating System Link List, or in the SPANEX TASKLIB dataset (if any). For integrity reasons, the module must reside in an APF authorized library. A general-purpose User Check Exit routine, SPXUCHEK, is supplied with SPANEX, and this is described in Chapter 2 of this manual.

The parameter passed to the exit routine is as follows:

| | | | | |
|---|---|---|---|---|
| R1 ➜ | +0 | A(SPANEX ICB - mapped by #SPXICB macro) | | |
| | +4 | X'xx' | Flags: | X'80' Exit called to check this step |
| | | | | X'40' Exit called retrospectively |
| | +5 | X'xx' | Flags: | X'80' Step Abended |
| | | | | X'40' Step failed with condition code |
| | | | | X'20' Step failed because of internal user macro call |
| | | | | X'10' Step cancelled by Stop/Modify command |
| | | | | X'01' SPANEX recognizes that the step failed |
| | +6 | X'xxxx' | Reserved | |
| | +8 | C'nnnn' | Step condition code, OR | |
| | | C'Unnnn' | Step User Abend code, OR | |
| | | C'Sxxx' | Step System Abend code | |
| | +12 | A(24-byte area) | Area contains Jobname (8 bytes), Stepname (8 bytes), Procstepname (8 bytes) | |

The exit routine may perform any desired function, and must return to SPANEX with one of the following return codes in Register 15:

| | |
|---|---|
| R15=0 | Use standard SPANEX procedures for determining whether or not the user program completed successfully. |
| R15=4 | Force a good end of the user program. All errors are ignored for the purposes of SPANEX restart or job networking. |
| R15=8 | Set the program execution as a failure. SPANEX takes the "Abnormal Termination Action" as specified in the EXEC statement PARM field. |
| R15=12 | Force a good end of the user program. All errors are ignored for the purposes of SPANEX restart or job networking. In addition, the contents of Register 1 upon return from the user check exit are used as the step condition code for this step. The maximum value of this code is 4095. If returned by a user check exit called as part of Retrospective Condition Code checking, this return code has the same effect as code 4, and the contents of Register 1 are ignored. |

Note that there are some restrictions in the parameter information that is passed to the exit routine. If the exit is called from Retrospective Condition Code checking (Bit X'40' set on in the first flag byte), and the jobstep abended, then the abend code value is not available to the exit, and this parameter value is not filled in. The area addressed by the fourth word of the parameter list must be used to determine the jobname and stepname. For Retrospective Condition Code checking the exit may be called multiple times in succession, and so must be reentrant or serially reusable; the exit module will not necessarily be reloaded by SPANEX between calls.

## 5.3 SPANEX Macros

Note that SPANEX Restart and Job Networking facility macros QUICKJOB, QUICKNET, QUICKSTP, SPXJOB, SPXSTEP, SPXRCM, #SPXRDEF, #SPXRSTU are described in the SPANEX Restart and Job Networking Guide (manual ref: SPX-03). Other SPANEX macros of interest to user programs are described in full in the Span Macros manual (ref: SPZ-02).

### 5.3.1 #SPXAUTH Macro - Mapping of Authorized Name Tables

The #SPXAUTH macro is required by user modules that require access to the SPANEX Authorized Program Name Tables. A USING statement on symbol SPXAUTH provides addressability to the table headers. The DSECT generated by this macro should be based upon the start of the load module SPXM0APT. The #SPXAUTH macro has no operands. The SPXAUTH control block is included in the SPANEX Automated Data Areas manual.

### 5.3.2 #SPXICB Macro - Generate SPANEX DSECTs

The #SPXICB macro is required by user modules that are required to run as SPANEX Span Products, and by all exit routines, Job Network job submit routines, etc. A USING statement on symbol SPXICB provides addressability as required by other SPANEX macros. In addition to this macro, a COPY statement must be included at the beginning of the assembly for #SPXGLOB.

For the format and operands of the #SPXICB macro see the Span Macros manual.

### 5.3.3 #SPXICBA Macro - Access non-fixed fields in SPANEX ICB

The #SPXICBA macro is used by SPANEX modules to access fields in the SPANEX Internal Control Block that may exist at different offsets in the ICB for different System Control Programs or different SPANEX releases.

For the format and operands of the #SPXICBA macro see the Span Macros manual.

### 5.3.4 #SPXQ Macro - Locate SPANEX ICB

The #SPXQ macro is provided for user programs to enable the address of the SPANEX ICB to be obtained if this is not available from the original parameter passed by SPANEX. The #SPXQ macro also enables user modules to determine whether or not they are executing under the control of SPANEX.

For the format and operands of the #SPXQ macro see the Span Macros manual.

### 5.3.5 #SPXSVC Macro - Issue SPANEX SVC

The #SPXSVC macro is used by user modules that are authorized to run as SPANEX Span Products, and by SPANEX user exit routines of all types, to request a function of the SPANEX SVC.

For the format and operands of the #SPXSVC macro see the Span Macros manual.

### 5.3.6 #SPXUDDN Macro - Define DDNAME for #SPXUMSG Requests

The #SPXUDDN macro is used by user modules that are designed to run as SPANEX Span Products, and provides the user program with the ability to specify the DDNAME to be used for the output of messages issued by means of the SPANEX #SPXUMSG macro. The #SPXUDDN macro must be executed before any #SPXUMSG macros. If no #SPXUDDN macro is used, messages issued by means of the #SPXUMSG macro will appear on the SPANEX Message Log.

For the format and operands of the #SPXUDDN macro see the Span Macros manual.

### 5.3.7 #SPXUMSG Macro - Message Request to SPANEX

The #SPXUMSG macro is used by user modules that are designed to run as SPANEX Span Products, and provides the ability to issue messages to all SPANEX destinations including the SPANEX Message Log, using the SPANEX message editing facilities. See Section 5.4 on page 50 of this manual for a discussion of SPANEX message editing.

For the format and operands of the #SPXUMSG macro see the Span Macros manual.

## 5.4 SPANEX Message Editing

Users of the #SPXUMSG macro instruction (and of the #SPXMSG internal message macro) may optionally have their message text modified by SPANEX

before being issued. This modification is not performed in the user's area and is triggered by the inclusion by the user of pre-defined character strings within the text of his message as passed to SPANEX. This function is always performed by default but may be disabled by specifying the "FILLIN=NO" option on the #SPXUMSG and #SPXMSG macros. Note that the length of the message may be altered when this string substitution is performed.

These character strings also occur in the SPXM0070 and SPXM0140 SPANEX message definition modules, and must be left unaltered if the wording of these messages is changed by the user installation (see Section 4.9 on page 44 of this manual).

The following fixed character strings are presently supported:

| | |
|---|---|
| ABCODX | Reserved |
| CPUTIMEXXXXXXXX | Reserved |
| CSCBNAME | Task name in SPANEX OPT=F CSCB |
| CVOLXX | SPANEX Catalog Volume Serial Number |
| FORMATTEDTIMEXXXX | Reserved |
| JOBNAMEX | Current Jobname/TSO Userid |
| PGMNAMEX | Name of user program being executed |
| PROCSTEP | Current Procedure Stepname |
| RCMNAMEX | Name of current Restart Control Module |
| RSBCODE | Reserved |
| RSBDATEX | Reserved |
| RSBSTEPX | Reserved |
| RSBTIMEX | Reserved |
| SPXABC | Reserved |
| STEPNAME | Current Stepname |
| USERIDX | TSO Userid to be notified by SPANEX |
| XRELNO | Current SPANEX Release Number (nnn.n) |
| XXDATA | Reserved |
| XXDATA1XXXX | Reserved |
| XXDATA2X | Reserved |
| XXDATA3XXXXXXXXX | Reserved |
| XXDATE | Current Date (YY.DDD) |
| XXDATEFX | Current Date (DDMMMYY) |
| XXTIMEXX | Current Time (HH:MM:SS) |
| ZZDATA | Reserved |

This page intentionally left blank.

# 6 SPANEX Operator Commands

SPANEX provides a number of commands that may be issued by the console operator when option "F" is specified as a SPANEX parameter. In order to process these commands, SPANEX creates and enqueues its own CSCB to which any SPANEX commands (STOP(P) or MODIFY(F)) must be addressed. The CSCB name to be used is displayed by SPANEX via message SPX063I at SPANEX initialization time. This section details the SPANEX commands that are currently available and their functions. This facility is not available if option "C" is requested.

The general format of MODIFY commands to SPANEX is as follows:

```
MODIFY  cscbname,text
        or
F  cscbname,text
```

where "cscbname" is the SPANEX CSCB name as specified in message SPX063I, and "text" is the SPANEX command.

## 6.1 STOP Command

The "STOP" SPANEX command has exactly the same function as the OS STOP (P) command as used when "OPT=P" is specified. Execution of the user program is immediately terminated and SPANEX ends the Jobstep by taking the Abnormal Termination action as specified in the SPANEX parameter.

## 6.2 ABEND Command

The "ABEND" SPANEX command has the same effect as the "STOP" command above, but overrides the Abnormal Termination action as specified in the SPANEX parameter and causes an ABEND to be issued for the Jobstep.

## 6.3 RETRY Command

The "RETRY" SPANEX command causes SPANEX immediately to terminate the execution of the user program task running under its control. The user task is then re-attached, and a completely new execution of the user program is begun. This facility is for use in special circumstances, where a long-running user program, run under SPANEX, may need to be restarted without the overheads and delay of terminating and re-submitting the job. Note that SPANEX option "R" must be specified for the jobstep, in addition to option "F", for the use of this command to be permitted.

## 6.4 SDUMP Command

The “SDUMP” SPANEX command causes SPANEX to take an immediate SVC dump of the Address Space or partition in which it is running. This facility allows the dynamic dumping of the user storage even if no dump DD statement was provided in the JCL for the Jobstep. Processing of the Jobstep continues after the dump has been taken. Note that the “STOP” or the “ABEND” command may be combined with the “SDUMP” command if processing of the Jobstep is not to be continued after the dump has been taken; eg enter “F cscbname,SDUMP,STOP”.

## 6.5 SNAP Command

The “SNAP” SPANEX command causes SPANEX to take an immediate SNAP dump of the user tasks to the “SPANSNAP” DD statement. Processing of the Jobstep continues after the dump has been taken. Note that the “STOP” or the “ABEND” command may be combined with the “SNAP” command if processing of the Jobstep is not to be continued after the dump has been taken; eg enter “F cscbname,SNAP,ABEND”.

## 6.6 SUSPEND Command

The “SUSPEND” SPANEX command causes SPANEX to suspend temporarily execution of all user tasks in the SPANEX Address Space or Partition. This facility allows all CPU usage for this job to be halted, perhaps to permit analysis of a problem in the system caused by this or by another job. Message SPX084E is issued as a reminder to the operator that execution of this job is suspended, so that, when required, execution may be resumed.

## 6.7 RESUME Command

The “RESUME” SPANEX command causes SPANEX to resume execution of all user tasks in an Address Space or Partition which had previously been suspended by means of a SPANEX “SUSPEND” command.

# 7 Examples of SPANEX Usage

This section gives some examples of how SPANEX may be used, for both batch and TSO environments. The examples given are not intended to be comprehensive, and by no means illustrate all the possible combinations of options and features that are provided by SPANEX. See also the SPANEX Restart and Job Networking Guide (manual ref SPX-03) for further examples.

## 7.1 Batch Examples

(a)
```
//  EXEC PGM=SPANEX,
//    PARM=PGM01
```

Program PGM01 is invoked.
ABEND U4095 is issued if PGM01 terminates abnormally. A SPANEX SVC Dump will be taken if the job is CANCELed with the “DUMP” option.

(b)
```
//  EXEC PGM=SPANEX,
//    PARM=PGM01
//SYSUDUMP DD  SYSOUT=A
```

Program PGM01 is invoked.
ABEND U4095 is issued if PGM01 terminates abnormally. A SPANEX SVC dump will not be taken if the job is CANCELed with the “DUMP” option.

(c)
```
//  EXEC PGM=SPANEX,
//    PARM='PGM02,ABEND,OPT=A/HELLO'
```

Program PGM02 is invoked with a PARM of “HELLO”.
ABEND U4095 is issued if PGM02 terminates abnormally. APF authorization is set (see Section 4.3 on page 40 of this manual).

(d)
```
//  EXEC PGM=SPANEX,
//    PARM='PGM03,SETRC=16,OPT=AN'
```

Program PGM03 is invoked.
Return Code 16 is set for the Jobstep if PGM03 terminates abnormally. APF authorization is set and the MVS Address Space will be set non-swappable (see Section 4.3 on page 40 of this manual).

```
(e)         //  EXEC PGM=SPANEX,
            //    PARM='PGM04,8,NOTIFY,OPT=C'
```

Program PGM04 is invoked within the SPANEX major task.
Return Code of 8 or less from PGM04 is treated as normal completion. The operator is notified if PGM04 terminates abnormally.

```
(f)         //  EXEC PGM=SPANEX,
            //    PARM='PGM05,NOTIFY=USER01/TESTPARM'
```

Program PGM05 is invoked with a PARM of "TESTPARM".
TSO user "USER01" is sent a message detailing the return/ABEND code from PGM05. ABEND U4095 is issued if PGM05 terminates abnormally.

```
(g)         //  EXEC PGM=SPANEX,
            //    PARM='PGM06/ABC'
            //SPXUPARM DD  *
            DEF
            /*
```

Program PGM06 is invoked with a PARM of "ABCDEF".
ABEND U4095 is issued if PGM06 terminates abnormally.

```
(h)         //  EXEC PGM=SPANEX,
            //    PARM='PGM07,CANCEL'
```

Program PGM07 is invoked.
The job is CANCELed by SPANEX if PGM07 terminates abnormally.

```
(i)         //  EXEC PGM=SPANEX,
            //    PARM='PGM08,OPT=D'
            //TASKLIB  DD  DSN=LOADLIB,DISP=SHR
            //SPANSNAP DD  SYSOUT=A
```

Operator authority is requested for the execution of program PGM08. Program PGM08 is invoked from library "LOADLIB" with that library set up as a Task Library, replacing any STEPLIB or JOBLIB that is specified. A SPANEX SNAP dump is taken if the job is CANCELed with the "DUMP" option or if the Jobstep is terminated because of excessive CPU utilization. ABEND U4095 is issued if PGM08 terminates abnormally.

```
(j)         //  EXEC PGM=SPANEX,
            //    PARM='SPANPGM1,OPT=S'
            //SPXPRINT DD  SYSOUT=A
```

Program SPANPGM1 is invoked as a Span Product (see Section 4.3 and Section 5.1 of this manual).
ABEND U4095 is issued if SPANPGM1 terminates abnormally. The SPANEX Message Log is produced.

## 7.2 TSO Examples

### 7.2.1 Obtaining a dump for a SPANEX User Program ABEND

(i) Allocate a SYSUDUMP DD statement:

```
ALLOC F(SYSUDUMP)DA(DUMP.LIST)SP(5 5)CYL
```

(ii) Enter the SP command specifying option “B”:

```
SP userpgm 'parm' OPT(B)
```

(iii) When the message:

```
SP ENDED DUE TO ERROR
READY
```

is received, press “ENTER” (or “RETURN”) without entering any other character

(iv) After message SPX009I is received, a dump will be found on dataset DUMP.LIST

### 7.2.2 Further TSO Examples

(a)
```
SP PGM01
```

Program PGM01 is invoked.
Return Code 16 is set if PGM01 terminates abnormally.

(b)
```
SP PGM02 'HELLO'
```

Program PGM02 is invoked with a PARM of “HELLO”.
Return Code 16 is set if PGM02 terminates abnormally.

(c)
```
SP PGM03 OPT(A)
```

Program PGM03 is invoked.
Return Code 16 is set if PGM03 terminates abnormally. APF authorization is set (see Section 4.3 on page 40 of this manual).

(d)
```
SP PGM04 OPT(T)
```

Program PGM04 is invoked as a TSO Command Processor.
The user is prompted for the TSO command to be processed.
Return Code 16 is set if PGM04 terminates abnormally.

(e)
```
SP PGM04 'PGM04 TEST' OPT(T)
```

Program PGM04 is invoked as a TSO Command Processor, and is passed the TSO command “PGM04 TEST”.
Return Code 16 is set if PGM04 terminates abnormally.

(f)
```
ALLOC FILE(TASKLIB) DA('LOADLIB') SHR
     SP PGM05 SETRC(24)
```

Program PGM05 is invoked from library “LOADLIB” with that library set up as a Task Library.
Return Code 24 is set if PGM05 terminates abnormally.

```
(g)   SP PGM06 ACCRC(20)
```

Program PGM06 is invoked.
Return Code of 20 or less is treated as normal completion. Return Code 16 is set if PGM06 terminates abnormally.

```
(h)   ALLOC FILE(SPXPRINT) DA(*)
           SP PGM07
```

Program PGM07 is invoked.
Return Code 16 is set if PGM07 terminates abnormally. The SPANEX Message Log is displayed (without page headings) on the user's terminal.

```
(i)   ALLOC F(SPXRSCTL) DA(*)
           SP IEFBR14 OPT(U) RCM(RCM01)
```

The SPANEX Utility (for the Restart and Job Networking facilities) is invoked in TSO foreground, with Restart Control Module RCM01 to be accessed for the processing of commands. The TSO terminal will be used for Utility command input.

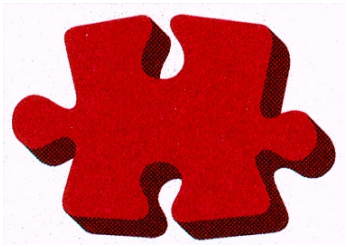
```
(j)   SP PGM08 SETRC(24) TASKLIB('LOADLIB')
```

Program PGM08 is invoked from library “LOADLIB” with that library set up as a Task Library.
Return Code 24 is set if PGM08 terminates abnormally.

# Appendix A Modules that require OPT=J when run under SPANEX

This table lists standard modules that have been found to require the SPANEX OPT=J parameter in order to execute successfully under the control of SPANEX. This table should always be checked if problems or abends are found when running user programs or IBM utilities under SPANEX.

| Module Name | Function | Comments |
|---|---|---|
| DFSUOCU0 | IMS/VS or IMS/ESA Online Change Utility | Also requires OPT=A |
| DSNUTILB | IBM DB2 Utility | Must be defined in SPXM0APB table, and requires Storage Protect Key 7 (see section 4.3 on page 40 of this manual) |
| GVRESTOR | CA-FAVER Restore | No other options required |



This manual is published by

Span Software Consultants Limited

Little Moss, Peacock Lane

High Legh

Knutsford

Cheshire

WA16 6PL

England

Tel: +44/0 1565 832999

Fax: +44/0 1565 830653

Email: spanex@spansoftware.com

www.spansoftware.com

to whom all comments and suggestions should be sent.