EspBatch

User Manual

READ ME FIRST

Copyright Reserved ©

Disclaimer

Cronus Consulting (Pty) Ltd hereby disclaims any and all guarantees and warranties for the correct use and application of the ESP software.

Cronus Consulting (Pty) Ltd reserves the right to revise and make changes to the software and the content of this document from time to time without obligation to notify any person of the changes.

Ownership

The ESP Products are developed by, and is fully owned by Cronus Consulting (Pty) Ltd.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose

EspBatch provides the effortless implementation, monitoring and control of batch processing in the Open Systems environment. Functionality that would typically be found in a mainframe type environment and that is not readily available on Open Systems platform is provided for. EspBatch makes use of a common mechanism to control batch jobs independent of the method used to submit the batch job, this ensures ease of use and reduces operator training. By using EspBatch developers and operators are shielded from the underlying complexities of the Open Systems command line environment, as scripts generation and return code validation are handled from within the product.

1.2 Product Overview

Batch job submission can be activated using one or a combination of the following methods:

- Online to Batch: User/Developer requests according to online parameter specification
- Request Jobs: Submission of pre-defined Batch jobs on a ad-hoc basis

Job classes can be limited on the number of jobs running concurrently as well as operational times when a job class is available.

Work file definition allows for files with a disposition of New, Mod (append to an existing file), Old and Print. GDG (Generation Data Groups) specification allows for multiple copies of a single file.

Reports can be routed to multiple destinations with individual specification of report name, class, disposition and the number of copies. The report class specified provides added security as sensitive data can be allocated to a class with limited user accessibility. Reports can be printed to disk or routed to a given E-mail address.

Dynamic parameter substitution at run-time provides additional flexibility and reduces changes to the predefined batch jobs.

Dynamic allocation of batch functions and security at function level determine who has access to specific functions.

Security is provided at function and option level, with additional security provided for SCL User libraries.

2. MENU OVERVIEW AND FUNCTION SELECTION

Once a user logs on the **EspBatch** menu will be displayed with all functions to which the user has access. The following information is displayed on the menu screen

Screen Layout:

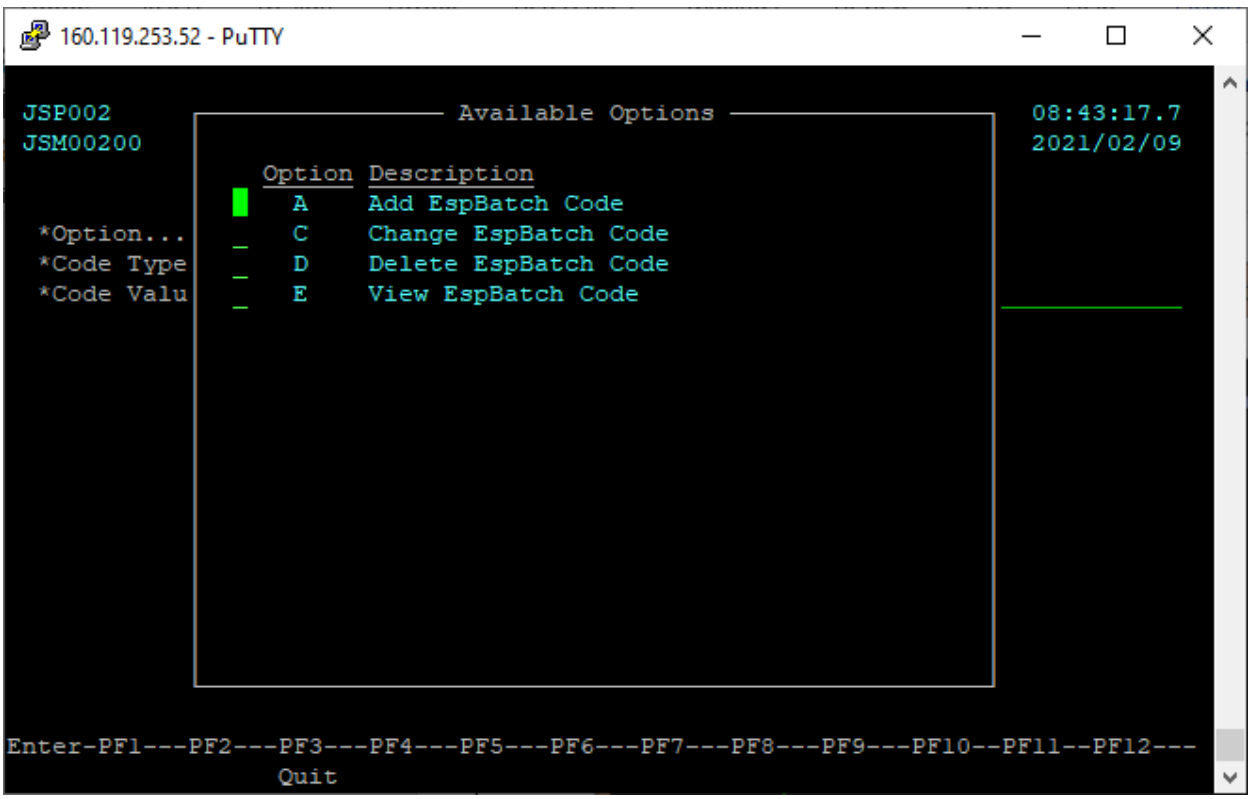
- Top line - Object name, main heading, product version and system time.
- Second line - Object name, menu name and system date.
- Lines 4 to 18 - Available menus/functions.
- Line 20 - Default Printer (If the user has not specified a default printer, then the value of Printer 1 defined on **JS002 Maintain - EspBatch Codes** with Code Type/Value **REPORT/DEFAULTS** is displayed).
  The User Id of the current user. **MENUADM** is displayed if the user has administrator privileges.
- Line 21 - Function selection area.
- Line 22 and 23 - Available function keys.

Function keys:

- PF2 - Invoke user exit **ESPUX001**.This user exit can be customized for your environment. It should contain code that transfers the user to the correct system location after exiting **EspMenu**. Function key is enabled if *"**Enable User Exit**" is set to **Y**es using function **MA001** **Maintain - EspMenu Setup**.*
- PF3 - Return to Main/Previous Menu
- PF7 - Page Backward
- PF8 - Page Forward
- PF9 - Set up default printers
- PF12 - Logoff

Function / Menu selection can be done as follows:

- All menu and function selection is cursor sensitive and can be accomplished by positioning the cursor on the line containing the desired menu or function and then pressing <ENTER>.
- Typing the menu or function name in the function selection area. Parameters required by the function can be typed in the data area, which will be passed to the function.



Once a function is selected that contains an OPTION field, the user will be required to enter a valid option, the option entered by the user will be validated against the predefined access set up for that user, thus only allowing the user to perform actions at function level according to their security profile. If the

option field is omitted and the <ENTER> key is pressed, a window containing valid options will be displayed allowing the user to choose the desired option. Option selection can be accomplished by marking the desired option or by placing the cursor in the line containing the option and pressing <ENTER>.

The **JS4*** range of functions are used to maintain and monitor EspAuto jobs.

3. FUNCTION OVERVIEW

3.1 JS002 Maintain - EspBatch Codes

This function is used to maintain codes that are specific to the run environment of all batch processes and should be administered by the person responsible for setting up and maintaining the system run environment. The values are populated during the installation process.

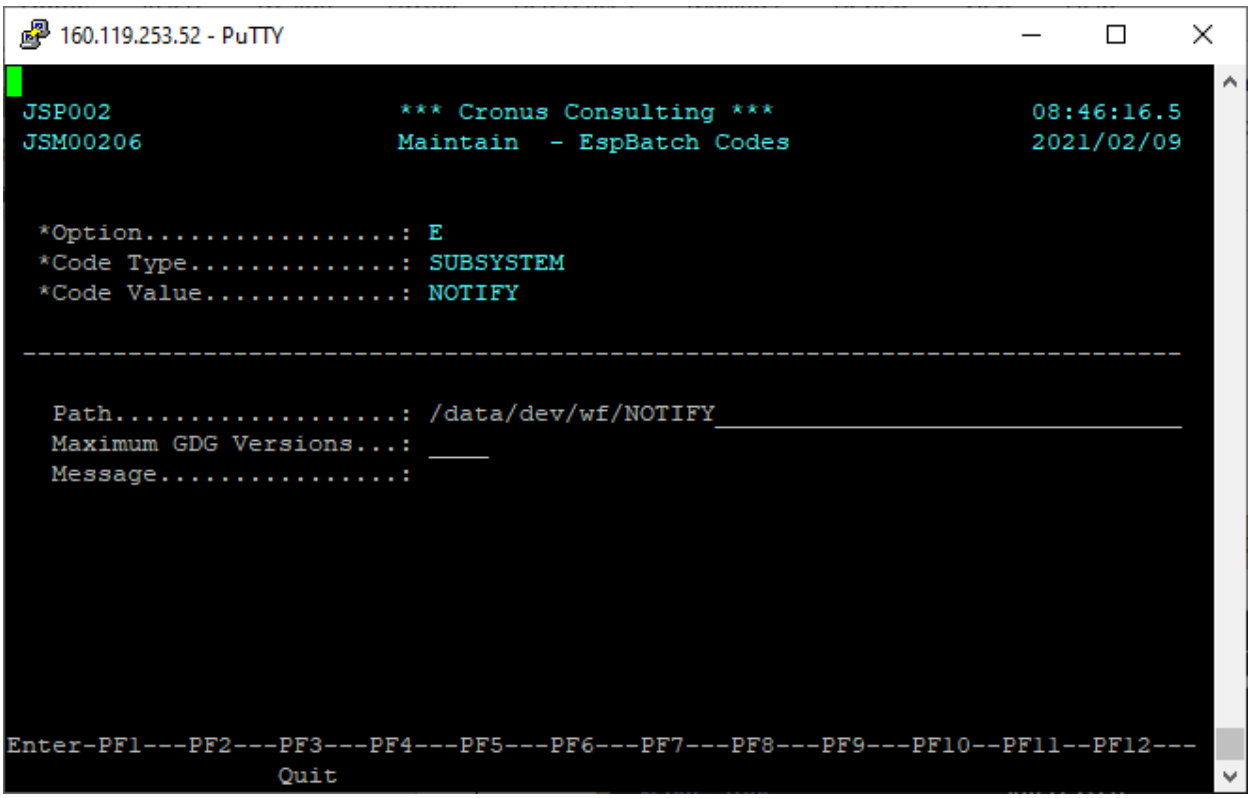
Function Options:

A - Add EspBatch Code

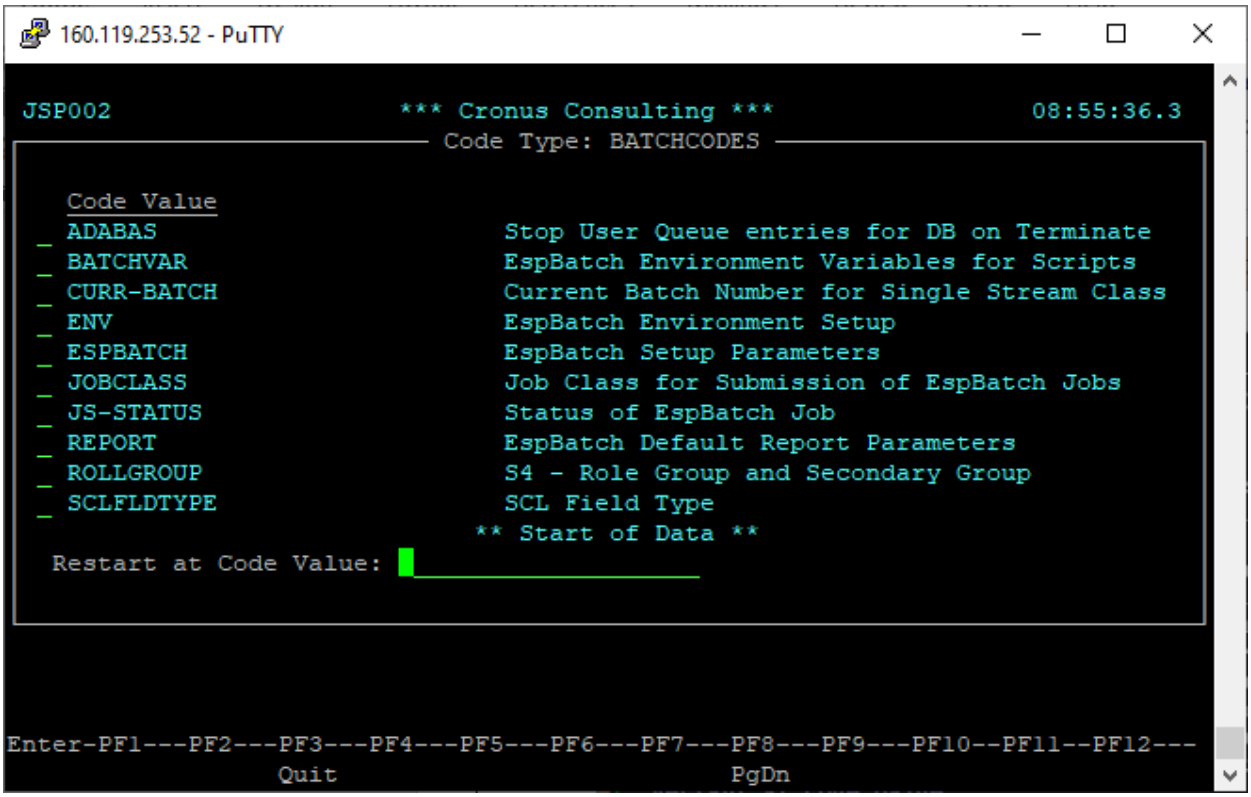
C - Change EspBatch Code

D - Delete EspBatch Code

E - View EspBatch Code



The following code types can be specified within **EspBatch**:



ADABAS: This code type is used to define a list of all the Adabas Database ID's for which the user queue entry must be stopped when terminating a batch job. A Code Value of **STOPUQ** must be entered.

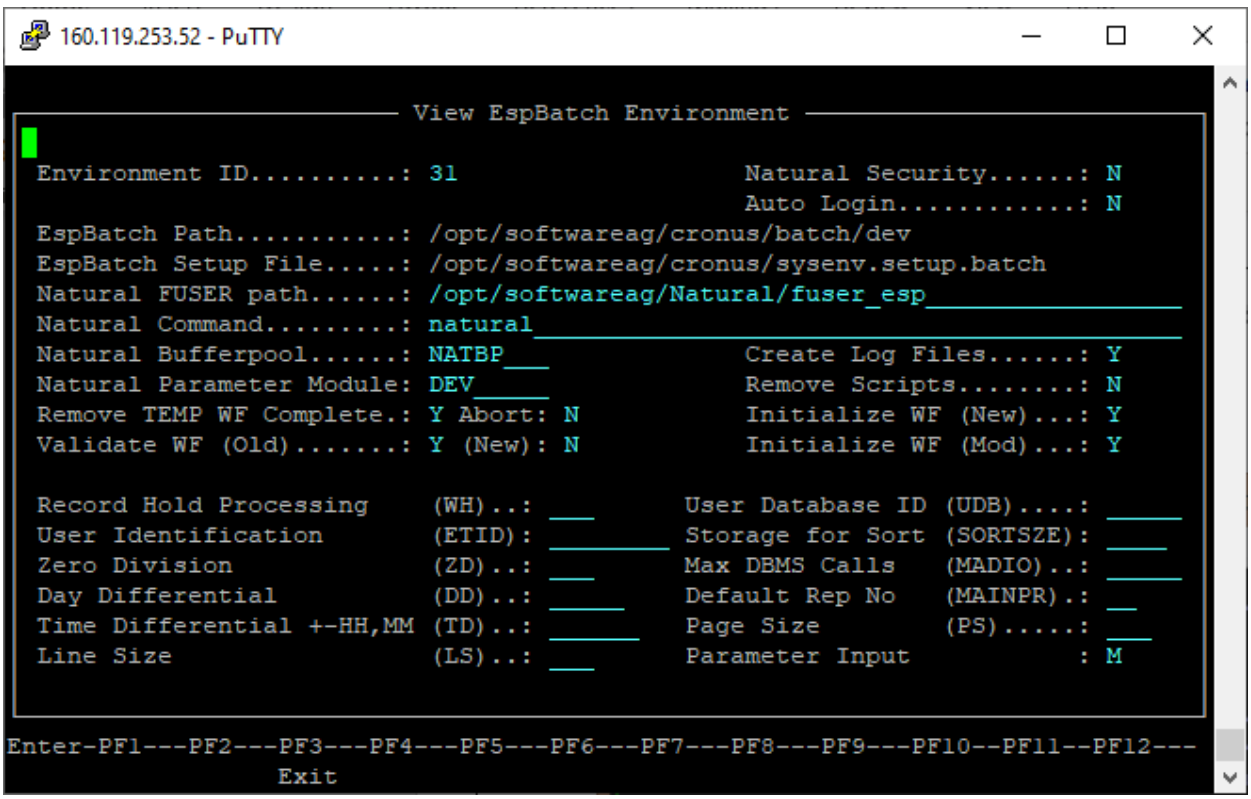
BATCHVAR: Allows for user defined environment variables. The code value contains the name of the environment variable. These environment variables are included in the job submission scripts generated by EspBatch. The values of these environment variables can be referenced within batch programs by making use of the **CALL 'GETENV'** statement. The following environment variables are required by EspBatch and may not by deleted:

- ADASCRIPTS: Location of Adabas utility scripts.
- BATCHMODE: Valid value are YES or NO. If set to YES natural batch jobs are executed in true batch mode. A value of NO will execute batch jobs in simulated batch mode.
- SIGNEMAIL: Valid values are (Y)es and (N)o. If set to (**Y**)es user exit **ESPUX012** is invoked. The content defined within the user exit is appended to the body of the email message.
- SKIPEMAIL: Valid values are: "SEND" and "SKIP". If the value is set to SKIP emails are not sent.

- SKIPFTP: Valid values are: “SEND” and “SKIP” If the value is set to SKIP execution of the FTP will not take place.
- TRIMFTP: Valid values are (Y)es and (N)o. If set to (**Y**)es, trailing spaces are removed from files before they are transferred via FTP using ESPPFTPB.

CURR-BATCH: The current batch code type is used internally by EspBatch for control of single stream job classes. The code value contains a value equal to the job class. An entry exists for each single stream job class and contains the Batch Number of the job that is currently being executed.

ENV: This code type is used to define run time environments used for submission of job steps. The Environment ID must be entered in the code value field and can contain a value of 1 to 254. By defining multiple environments, job steps can be submitted with unique parameter settings. When batch jobs are defined within EspBatch, the run environment can be specified per job step – if no environment is specified the default EspBatch environment is used for submission. The default EspBatch environment is defined within Code Type/Value: **ESPBATCH/SETUP**.



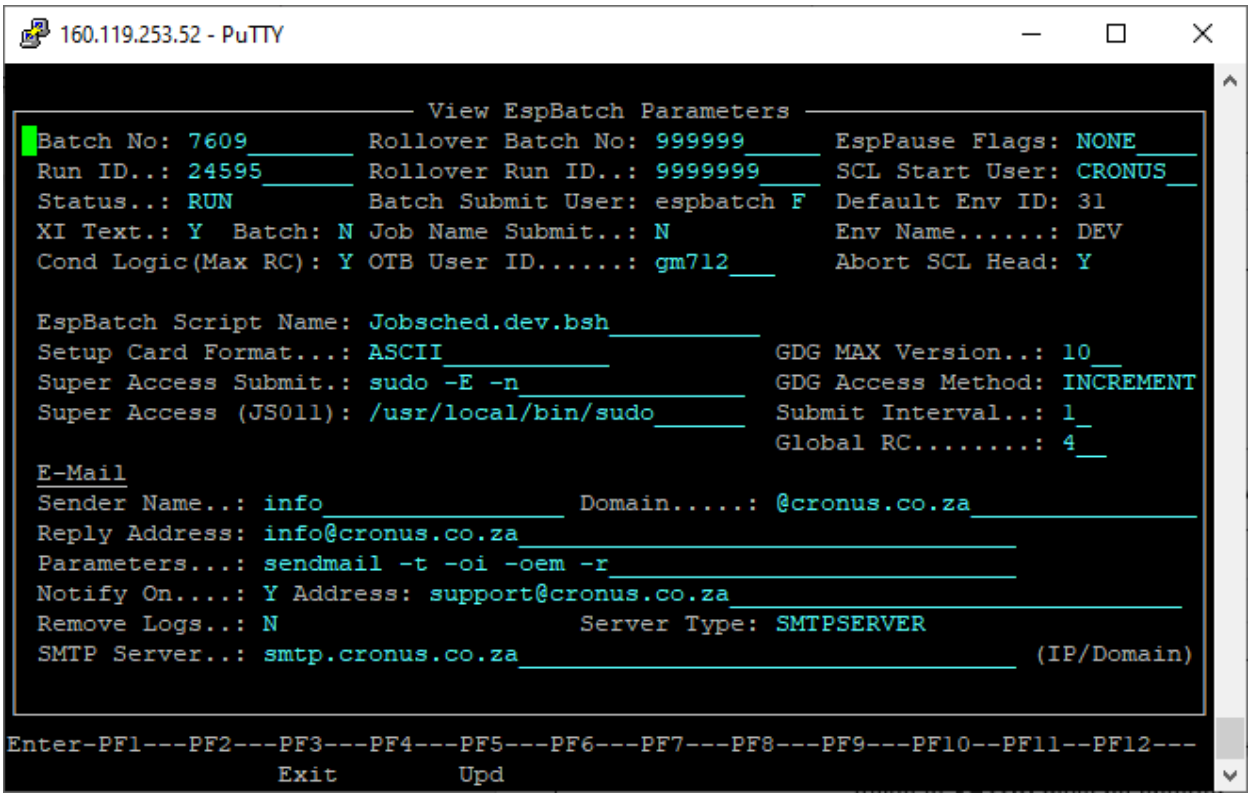
When a batch job is submitted the current Natural sessions UDB value is used to determine which EspBatch run environment is used for job execution.

Environment ID:	Numeric value used to identify the environment.
Natural Security:	Valid values are: (Y)es and (N)o. If set to (Y)es the job step will run under control of **Natural Security**.
Auto Login:	Valid values are: (Y)es and (N)o. If set to (Y)es Natural Security is invoked with **AUTO=ON**.
EspBatch Path:	Contains the file system path used by EspBatch for script generation and log file creation. The EspBatch Path is defined by environment variable **JBSJE** within the EspBatch environment file **$CRONUS/scripts/espenv.<env>.bsh**
EspBatch Setup File:	Contains the file system path and name of the EspBatch environment file. The environment file contains settings required by EspBatch which include environment variables, a link to the shared libraries and execution of the “sagenv”. The Setup File name cannot be changed.
Natural FUSER path:	Contains the file system path of the EspBatch source code. The Natural FUSER Path may only be defined for the default EspBatch environment. This field is protected for other environments.
Natural Command:	Contains the name of the natural executable. Default is **natural**.
Natural Bufferpool:	The Natural Bufferpool used when submitting jobs steps using the specified environment. The Natural Bufferpool is defined by environment variable **JBSBP** within the EspBatch environment file **$CRONUS/scripts/espenv.<env>.bsh**
Create Log Files:	Valid values are: (Y)es and (N)o. If set to (Y)es a detailed log file is created in Sub System **LOG** for each batch job. The naming convention used for these log files is **batch-id-<Batch No>.<env>**. Batch No contains the system generated batch number and env contains the value of batch variable **ENVIND**.
Natural Parameter module:	The Natural parameter module used when submitting jobs steps using the specified environment. The Natural parameter module is defined by environment variable **JBSPARM** within the EspBatch environment file **$CRONUS/scripts/espenv.<env>.bsh**

Remove Scripts:	Valid values are: (Y)es and (N)o. If set to (Y)es the script used to submit a Job step is removed after the script has completed execution. A value of (N)o retains the script after execution.
Remove TEMP work files:	Temporary work files are defined by preceding the work file name with "&&". At submission time the "&&" is replaced with the jobs batch no.
Complete:	Valid values are: (Y)es and (N)o. If set to (Y)es temporary work files are removed after completion of the batch job. A value of (N)o retains temporary work files.
Abort:	Valid values are: (Y)es and (N)o. If set to (Y)es temporary work files are removed when a batch job aborts. A value of (N)o retains temporary work files.
Initialize WF (New):	Valid values are: (Y)es and (N)o. If set to (Y)es work files that have a disposition of (N)ew are empty created at execution start time. A value of (N)o will not create the work file.
Validate WF (Old):	Valid values are: (Y)es, (N)o and (A)bend. If set to (Y)es the job step will abend if the work file does not exist and the Return Code does not meet the Condition Code specified for the job. If set to (A)bend the job step will abend if the work file does not exist, Condition Code is not evaluated. If set to (N)o the job step will not abend if the work file does not exist.
Validate WF (New):	Valid values are: (Y)es, (N)o and (A)bend. If set to (Y)es the job step will abend if the work file exists and the Return Code does not meet the Condition Code specified for the job. If set to (A)bend the job step will abend if the work file exists, Condition Code is not evaluated. If set to (N)o the job step will not abend if the work file exists.
Initialize WF (Mod):	Valid values are: (Y)es and (N)o. If set to (Y)es work files that have a disposition of (M)od are empty created at execution start time. A value of (N)o will not create the work file.
Profile parameters:	Refer to Software AG's Natural documentation for a description of the allowable natural parameter module parameter values.

Parameter Input: Valid values are: (S)ingle and (M)ulti. This parameter is used to control the method whereby parameters are stacked for input within a natural module. If (S)ingle is specified, the natural module must contain a single INPUT statement e.g. INPUT #PARM1 #PARM2. If (M)ulti is specified then each parameter must have its own INPUT statement e.g. INPUT #PARM1, INPUT #PARM2. Note that natural programs executed from library ESPSOFT are always executed with parameter input type (M)ulti. If no value is specified, the parameter input type defaults to (S)ingle for all libraries except library ESPSOFT.

ESPBATCH: This Code Type contains parameters essential to the working of EspBatch. A Code Value of **SETUP** must be entered.



Batch No: Contains the value of the last Batch Number generated. The Batch No uniquely identifies each Batch Job that is submitted. Care must be taken when changing the value

of the Batch Number as this could provide unpredictable results when submitting batch jobs.

Rollover Batch No: Contains the Batch No at which point rollover will occur. If a rollover value is entered and the last Batch No generated is equal to the value entered - Batch No generation will restart at zero. Minimum value is 2000.

EspPause Flags: Informational messages are logged during various execution stages in EspBatch. The flags set determine the how much information is supplied to EspConsole.

- **B** At start of Job step
- **E** On Error of Job step
- **C** On completion of Job step
- **R** Job step was restarted after error
- **F** Job step was “force completed”
- **T** Job step was terminated using function **JS011 Terminate - Executing Jobs Steps**
- **P** Program Requests (ESPPAUSO - user call from Natural Module)
- **NONE** No messages to Unix console

Run ID: Contains the value of the last Run ID generated. The Run ID uniquely identifies each Job step that is submitted. Care must be taken when changing the value of the Run ID as this could provide unpredictable results when submitting batch jobs.

Rollover Run ID: Contains the Run ID at which point rollover will occur. If a rollover value is entered and the last Run ID generated is equal to the value entered – Run ID generation will restart at zero. Minimum value is 5000.

SCL Start User: The SCL User Library entered is used as starting value for SCL maintenance and submission functions **JS300** & **JS310**.

Status: Protected field that displays the current status of the EspBatch job scheduler. Value **RUN** indicates that the scheduler is active, while **KILL** indicates that the scheduler has been terminated.

Batch Submit User: When a batch job is submitted an option is provided whereby the batch job can be submitted as the batch

	user specified in this field. By default, batch jobs are submitted using the User ID of the person that requested the submission. The User ID specified here should be an existing O/S User ID. If the field next to the "Batch Submit User" contains a (F)orce, all batch jobs submitted via functions JS300 and JS310 will be submitted using the "Batch Submit User" ID.
Default Env ID:	Contains the default Environment ID used by EspBatch. Care must be taken when setting up the EspBatch default environment to ensure that the parameters specified do not adversely affect the working of EspBatch. The **Default Environment ID** must contain the same number as the **Database ID** being used by EspBatch as this is required for termination of job steps using function **JS011 Terminate - Executing Jobs Steps**. If the **Default Environment ID** differs from the EspSoft database ID, the **TF** parameter in the NATPARM must be used to translate the DBID for all EspSoft adabas files. The **Default Environment ID** is defined by environment variable **JBSDBID** within the EspBatch environment file **$CRONUS/scripts/espenv.<env>.bsh**
Xi-Text:	Valid values are: (Y)es and (N)o. A value of (Y)es indicates that the product Xi-Text is installed and is being used for Print Spooling from EspBatch.
Batch:	Valid values are: (Y)es and (N)o. A value of (Y)es indicates that the product Xi-Batch is installed and is being used to run the EspBatch daemon.
Job Name Submit:	Valid values are: (Y)es and (N)o. If set to (Y)es jobs submitted via **JS300** and **JS310** will be submitted using the "SCL Name" as User ID. If the JOBNAME is not a valid O/S User ID, the user will be prompted to confirm submission using their own User ID for the submission.
Env Name:	Identifies the EspBatch run environment. The environment name can be referenced within batch programs by making use of the **CALL 'GETENV' 'ENVIND'** statement.

	The **Env Name** is defined by environment variable **JBSENV** within the EspBatch environment file **$CRONUS/scripts/espenv.<env>.bsh**
Cond Logic (Max RC):	Valid values are: (Y)es and (N)o. If set to (Y)es the highest RC produced by all executed job steps is used when evaluating Conditional Logic and returned to EspAuto or the EspBatch API. If set to (N) the RC of the last job step that executed is used when evaluation Conditional Logic and returned to EspAuto or the EspBatch API.
OTB User ID:	If a User ID is specified, online to batch (OTB) jobs are submitted using the specified User ID. A blank value submits online to batch (OTB) jobs with the user id of the user that requested the job.
Abort SCL Head:	Valid values are: (Y)es and (N)o. A value of (Y)es indicates that a SCL will abort if the return code of a job step does not satisfy the condition code specified on the SCL header. A value of (N)o indicates that all subsequent job steps will be bypassed should the return code of a given step not satisfy the condition code specified at the SCL header.
EspBatch Script Name:	Contains the name of the EspBatch script. If Xi-Batch is being used, the Title of the Job Scheduler entry within Xi-Batch must be specified.
Setup Card Format:	Contains the value of the work file format to be used when expanding @@PROC@@ statements defined as input variables within an SCL step (Refer to input variables under function **JS300 - Maintain - SCL's**. The value must be set to one of the available work file formats allowed within the Natural NATPARM.
GDG MAX Version:	Defines the maximum number of GDG occurrences per work file (Global Setting). The maximum number of GDG versions can also be defined at Sub System and file level.
Super Access Submit:	Contains the file system path and name of the executable used for privileged access required by EspBatch for submission of batch jobs e.g. (sudo, dlgexec, etc.)

GDG Access Method:	Defines the Generation Data Groups access method (INCREMENT or ROTATE). Access method INCREMENT uses M/F style GDG generation whereby the generation number is incremented by 1 for each new instance of the file until the MAX version is reached. When the MAX version is reached the generation no restarts at 1. Work files are suffixed by _Gnnnn – where nnnn denotes the GDG version. Access method ROTATE constantly renames the GDG versions, whereby the first GDG version has a suffix of .1, when the next version is created version .1 becomes version .2 and the current version takes its place. Use of GDG access method “ROTATE” is no longer supported in the current EspBatch version.
Super Access JS011:	Contains the file system path and name of the executable used for privileged access required by EspBatch to terminate batch jobs using function **JS011 Terminate - Executing Jobs Steps** e.g. (sudo, dlgexec, etc.)
Submit Interval:	The value specified is the number of seconds that EspBatch waits before checking if there are any batch jobs to submit.
Global RC:	Return code that EspBatch returns if a job step fails. Within certain utilities the return code can be specified.
Sender Name:	Contains the **name** in the from email address field.
Domain:	Contains the **domain** in the from email address field. E.g. **@domain.com**
Reply Address:	Contains the reply email for non delivery notification e.g. **info@cronus.co.za**
Parameters:	Parameters used by the sendmail program if server type is defined as LOCALMAIL. These parameters should not be changed. The default values are: **sendmail -t -oi -oem –r**
Notify On:	Valid values are: (Y)es and (N)o. If set to (Y)es a read receipt is requested from the recipient(s).
Address:	Contains the email address for response from a read receipt.
Remove Logs:	Valid values are: (Y)es and (N)o. The ESPMAILD utility generates a trace and log file in the utilscripts directory

located under the EspBatch Path. If the value is set to (Y)es these files are removed after successful completion of the job step.

Server Type: Valid values are: SMTPSERVER or LOCALMAIL.

SMTP Server: If the server type is defined as SMTPSERVER then the ip-address or dns of the SMTP server must be specified.

JOBCLASS: This Code Type is used to define available Job Classes. The code value contains the letter/number identifying the Job Class. Valid Job classes are 'A' to 'Z' and '0' to '9'. After a Job class has been added function **JS003 Maintain - EspBatch Job Classes** must be used to set up the rules governing that class.

JS-STATUS: This Code Type displays the description for each possible status that a job step can have. This is an enquiry only code type.

REPORT: This Code Type is used to specify default report parameters and to define the report heading format for both online and batch reports.

Code Values:

- **DEFAULTS**: Define default report parameters for reports 1 to 31.
- **FORMAT-BATCH**: Define report heading format for batch reports.
- **FORMAT-ONLINE**: Define report heading format for online reports.

SCLFLDTYPE: This Code Type displays the field types that can be selected when scanning SCL's using function **JS302 Scan SCL's**. This is an enquiry only code type.

SCLHEAD: This Code Type displays the field attributes that can be selected for field type **SCLHEAD** when scanning SCL's using function **JS302 Scan SCL's**. This is an enquiry only code type.

SCLREPORT: This Code Type displays the field attributes that can be selected for field type **SCLREPORT** when scanning SCL's using function **JS302 Scan SCL's**. This is an enquiry only code type.

SCLSTEP: This Code Type displays the field attributes that can be selected for field type **SCLSTEP** when scanning SCL's using function **JS302 Scan SCL's**. This is an enquiry only code type.

SCLVAR: This Code Type provides the ability to setup dynamic variables that are replaced within an SCL at submission time. The code value must contain the variable name. These variables may be references within an SCL by preceding the variable name with an '&'.

SCLWF: This Code Type displays the field attributes that can be selected for field type **SCLWF** when scanning SCL's using function **JS302 Scan SCL's**. This is an enquiry only code type.

SUBSYSTEM: This Code Type is used to define Sub Systems. A Sub System is a logical definition used within EspBatch to point to physical disk location (path) that contains work files. The code value must contain the name of the Sub System. The global GDG specification can be overwritten per sub system by entering the maximum number of GDG versions.

WF-TYPE: This Code Type displays the available work file types that can be used when defining work files in batch jobs.

Code Types can be updated at any stage by pressing **PF5**. If **PF3** is pressed and the Code Type has been changed after the last **PF5** key press a window is displayed prompting the user if changes should be saved.

3.2 JS003 Maintain - EspBatch Job Classes

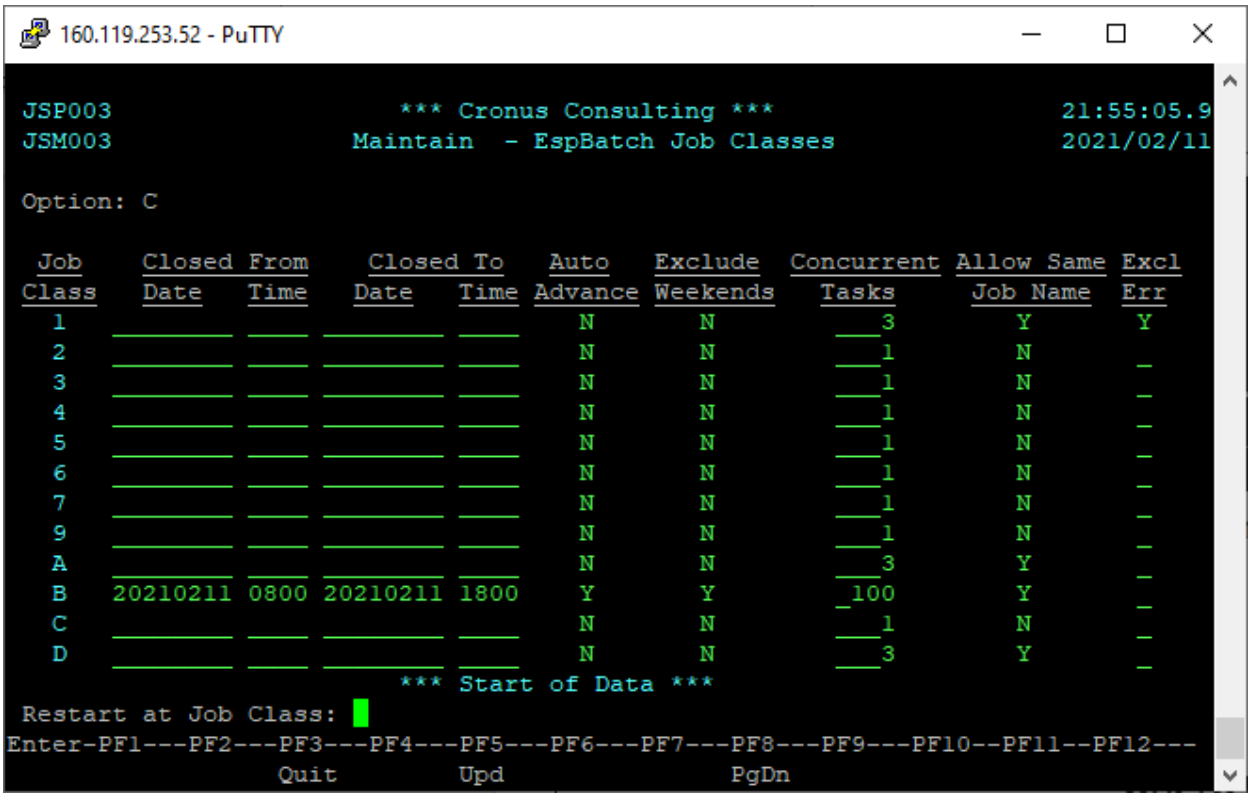
This function is used to maintain defined Job Classes added using function **JS002 Maintain - EspBatch Codes**.

Function Options:

C - Change Job Classes

E - View Job Classes

.



Job class settings:

Closed From/To Date and Time - If specified all jobs submitted in the given class will not execute during the specified period.

Auto Advance - If a Closed From/To Date/Time is specified and auto advance is set to (**Y)**es the Closed From/To Date/Time will be rolled forward on a daily basis.

Exclude Weekends - If set to (**Y)**es the date/time constraints specified for the job class will not be in effect on weekends.

Concurrent Tasks - Determines the maximum number of jobs that can execute concurrently in that job class.

Allow Same Job Name	-	If set to **Y**es, jobs that have the same name will be allowed to run concurrently for this job class. The job name is derived from the **SCL Name** defined on **JS300** – for online to batch jobs the job name is assigned to variable **#JS-BATCH-NAME**.
Excl Err	-	If set to **Y**es, jobs that are in an error status are ignored. This is useful for single stream classes to ensure that jobs with the same name will still execute when a job with the same name has aborted.

Note: If EspBatch is running under control of Xi-Batch then all Job Classes must be defined within Xi-Batch.

3.3 JS004 Maintain - EspBatch Printers

This function maintains logical printer information.

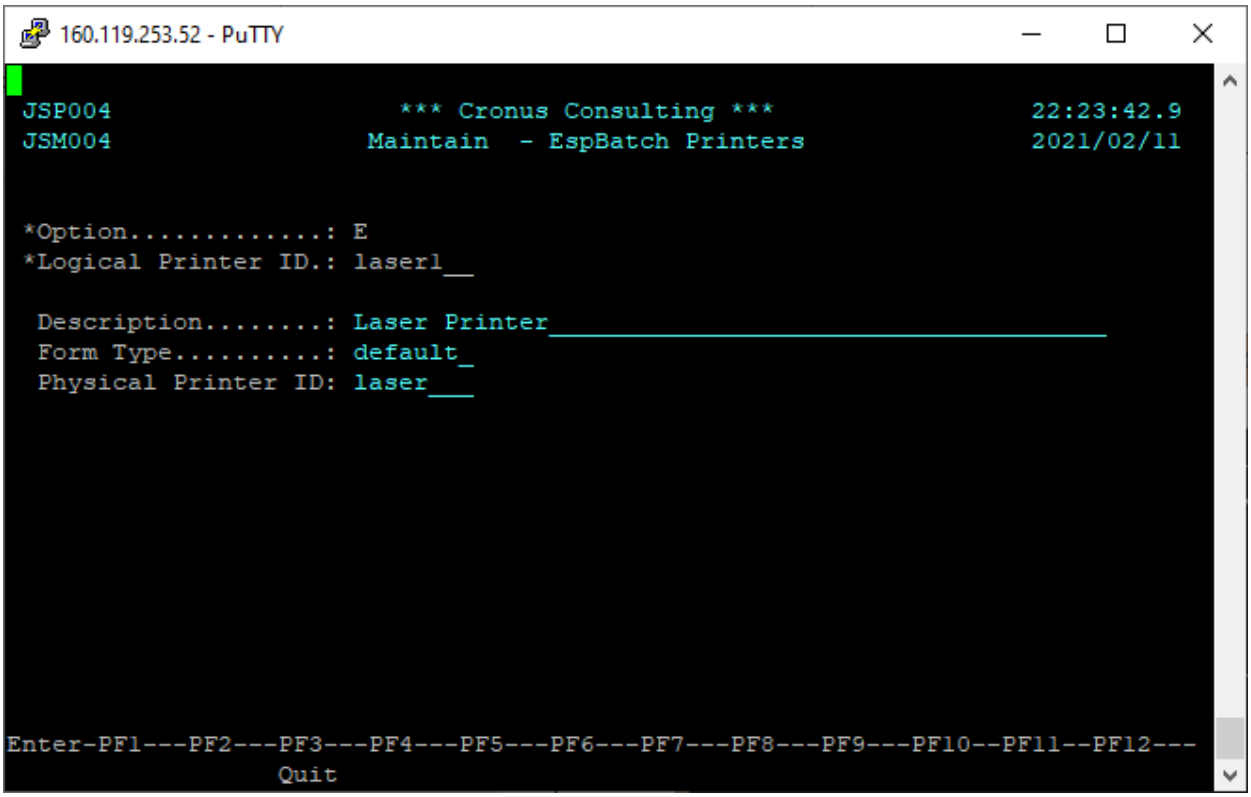
Function Options:

A - Add Printer

C - Change Printer

D - Delete Printer

E - View Printer



By defining logical printer definitions, a physical printer can be addressed using a variety of different form types and logical aliases. A form type is used to define print specifications for different types of stationery.

The following information must be entered per printer:

- Description - Brief description of the printer and its location.
- Form Type - Form Type to be used when the report is routed to the printer.
- Physical Printer ID - Physical printer name as defined in print spooler.

3.4 JS007 Maintain - GDG Versions per Work File

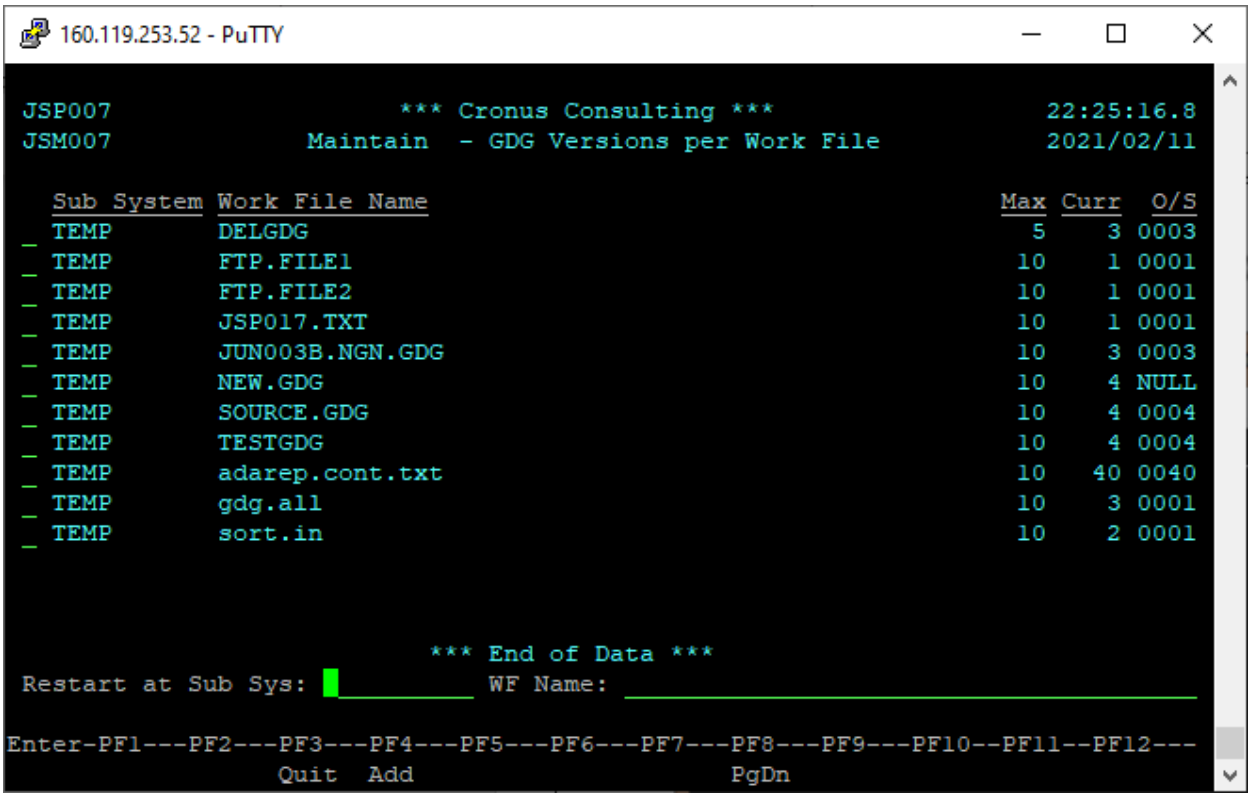
This function is used to maintain the number of GDG versions per Work File. The number of GDG versions specified on this function will override the GDG specification at Sub System and Global level.

Function Options:

A - Add GDG (Function Key **PF4** can also be used to add a new GDG)

C - Change GDG

D - Delete GDG



The following information must be entered:

- Sub System - The sub system where the work file is located
- Work File Name - Name of the work file.
- Max Versions - Maximum GDG versions for the specified work file.
- Current Ver - Current GDG version
- O/S - Latest O/S version of GDG

When a file is created as a GDG, "_Gnnnn" (where nnnn is the generation number) is appended to the work file name.

For the following file extensions ".SAG", ".ZIP", ".PDF", ".RTF" and ".GZ" "**_Gnnnn**" is inserted before the extension. E.G. **MYFILE.G0001.SAG**

If no entry exists for the Sub System/Work File combination an entry is system generated. The Max Version is set to the Global GDG Max Version value or Sub System Max GDG value if specified.

3.5 JS010 Maintain - Aborted Job Steps

This function is used to resubmit aborted batch jobs. If a step in a batch job aborts and does not satisfy the condition codes specified it will remain on this screen until it gets re-submitted or completed by means of operator intervention.

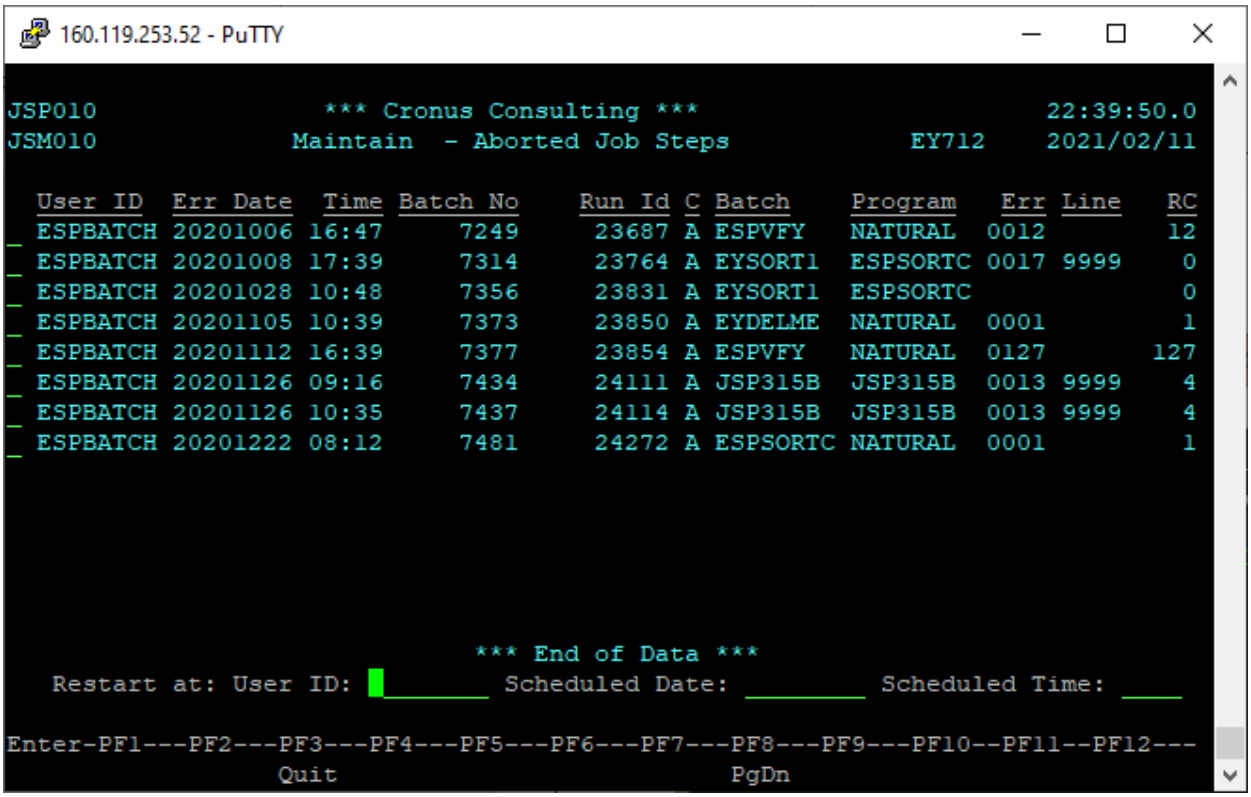
Function Options:

C - Force Complete Job Step

E - View Error Description

S - Select Job

X - Resubmit Aborted Job Step



When a job step is resubmitted, it will be rerun with the same parameters. These parameters can be modified before resubmission using function **JS050 Maintain - EspBatch Jobs**.

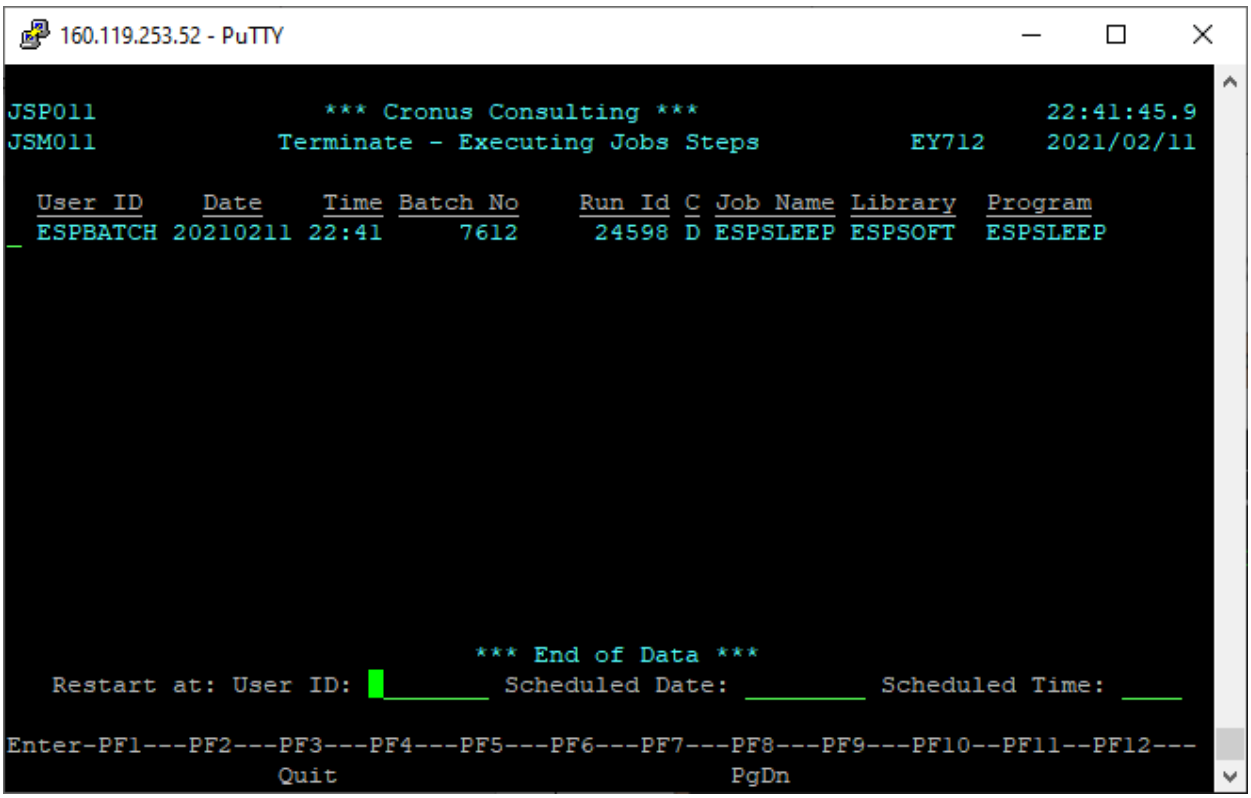
Completing a job step will change the status of the step to **SKIPPED** and the next job step will start running. If this was the last step the batch job will be completed. The job step that aborted will still display the error that caused it to abort even though its status was changed to **SKIPPED**.

3.6 JS011 Terminate - Executing Jobs Steps

This function is used to terminate a job step that is busy executing.

Function Options:

T - Terminate Job Step

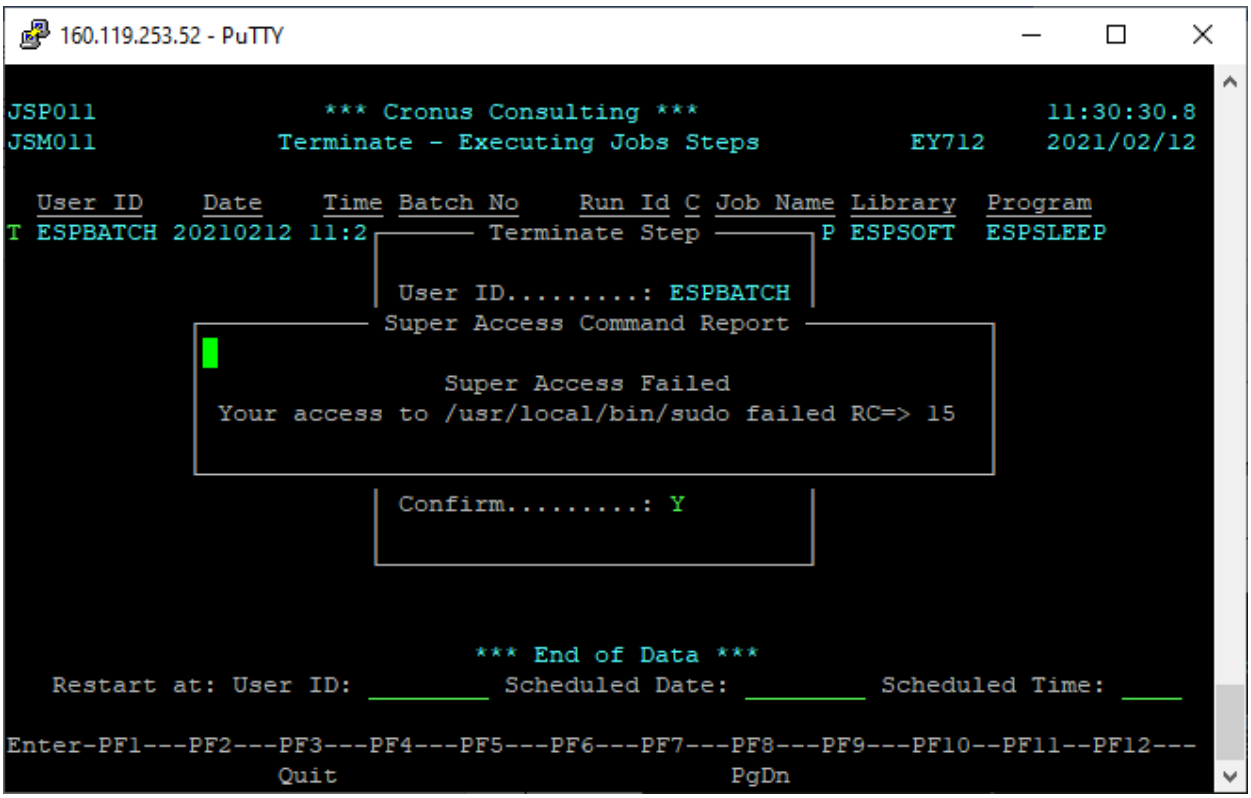


O/S command verification

To terminate a job step, the user issuing the **JS011** Terminate option must have access to the O/S command defined in the **Super Access (JS011)** field on function **JS002** with Code Type: **ESPBATCH** Code Value: **SETUP**.

The O/S terminate command is first validated before issuing the actual “kill” command.

The following message is displayed if the O/S command validation failed.



The above example illustrates the error message where the **Super Access (JS011)** is set to use O/S command "**sudo**".

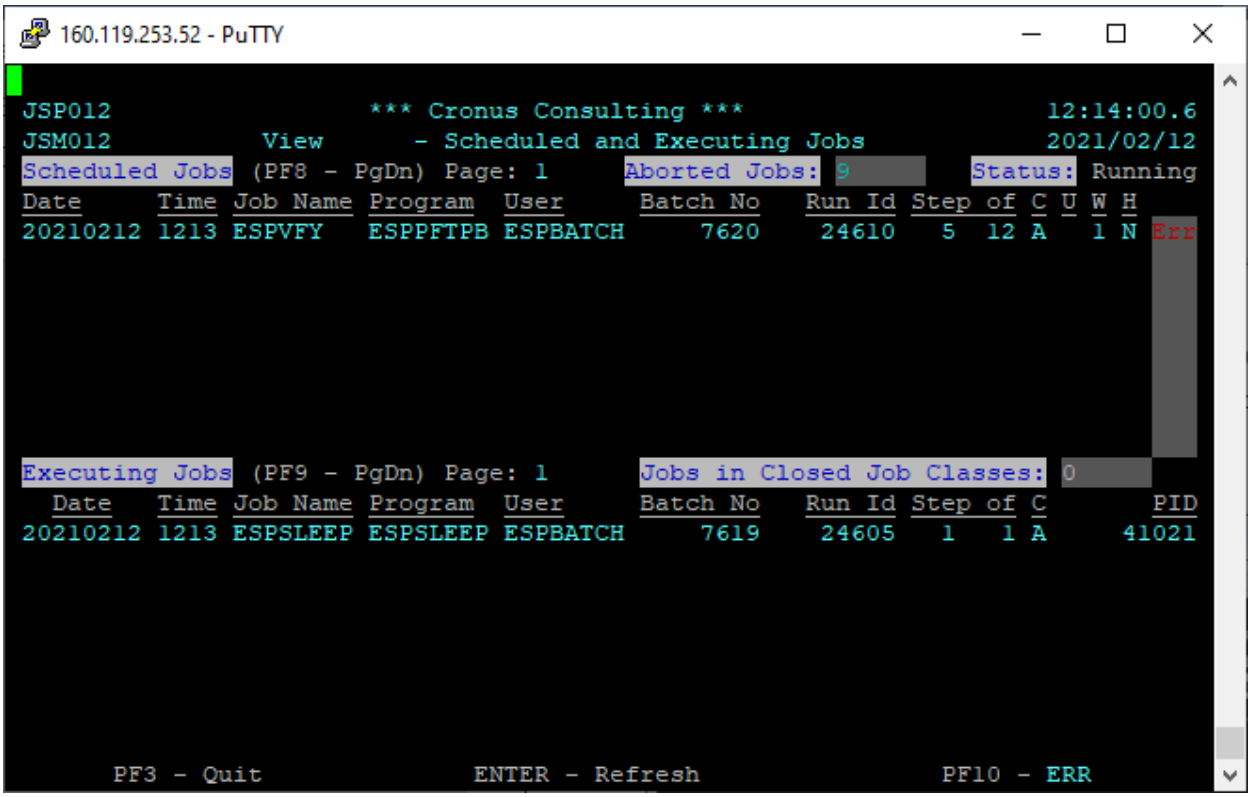
"sudo" access must be added for the user by the administrator using the "**visudo**" command. The O/S command issued to terminate the batch process using "**sudo"** is: **sudo kill -15 <PID>**

The step will terminate with EspBatch Error 7777 - Job Terminated with function JS011.

3.7 JS012 View - Scheduled and Executing Jobs

This function is used to monitor all jobs that are currently in the batch input queue(s) or are busy executing. This is a display only function – no actions can be performed.

.



The screen is divided into two sections. The top portion displays jobs that are currently in the Input Queue, while the bottom portion displays executing jobs.

Input Queue:

- Scheduled Date/Time - The Date/Time that the job was submitted.
- Job Name - The name of the Job (SCL Name)
- Program - The name of the program/utility that is waiting to run
- User - The User Id that submitted the Job
- Batch No - The Batch No assigned to the Job
- Run Id - The Run Id assigned to the Job Step
- Step of - The Step No that is currently awaiting execution and the number of steps that the Job consists of.
- C - The Class in which the Job was submitted.
- U - A '**1**' will be displayed whilst the script is being generated to execute the Step.

- W - If a '**1**' is displayed it indicates that the previous step has aborted. The current step will only start execution once the previous step has completed. To view the step that has aborted use function **JS010**.
- H - A '**Y**' indicates that the current job step is in Hold. Use function **JS014** to release the step.

Additional information displayed as part of the input queue is:

- The number of job steps that have aborted and are awaiting operator intervention.
- The status of the Jobs Scheduler (Running or Stopped). If the status of the Job Scheduler is "Stopped", it will not prohibit jobs from being submitted. Jobs will remain in the input queue until the Job Scheduler is restarted.

Use **PF8** to page forward through the input queue.

Executing Jobs:

- Start Date/Time - The Date/Time that the job started execution.
- Job Name - The name of the Job
- Program - The name of the program that is waiting to run
- User - The User Id that submitted the Job
- Batch No - The Batch No assigned to the Job
- Run Id - The Run Id assigned to the Job Step
- Step of - The Step No that is currently busy executing and the number of steps that the Job consists of.
- C - The Class the Job in running in.
- PID - The UNIX/Linux process ID.

Additional information displayed as part of the executing queue is:

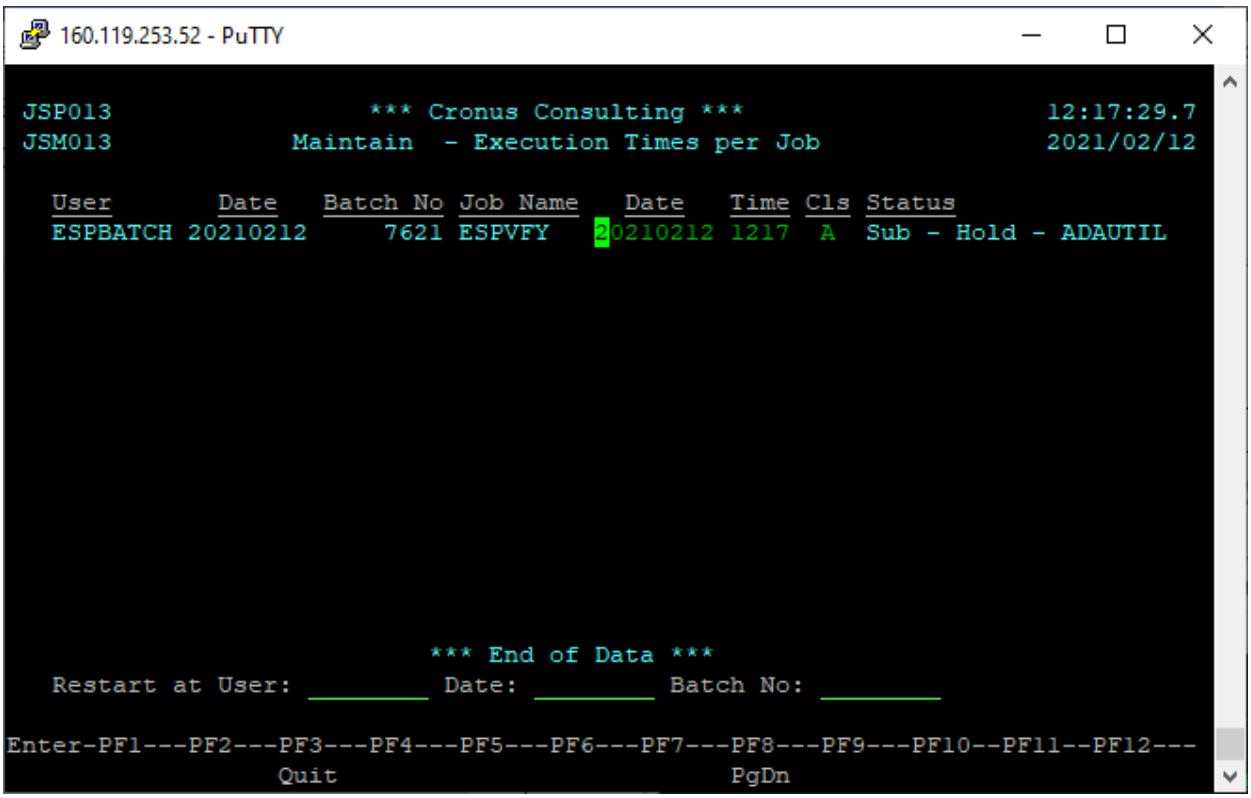
- The number of jobs that were submitted in a Class that is currently closed. Function **JS003 Maintain - EspBatch Job Classes** can be used to view closed Job Classes.

Use **PF9** to page forward through the executing Jobs.

Note: The display is only updated after a function key press or <ENTER>.

3.8 JS013 Maintain - Execution Times per Job

This function provides the ability to change the Scheduled Date, Scheduled Time and Job Class of Batch Jobs that are currently in the Input Queue. The Scheduled Date, Scheduled Time and Job Class can also be changed with function **JS050 Maintain - EspBatch Jobs** by making use of options “C” and “U”.

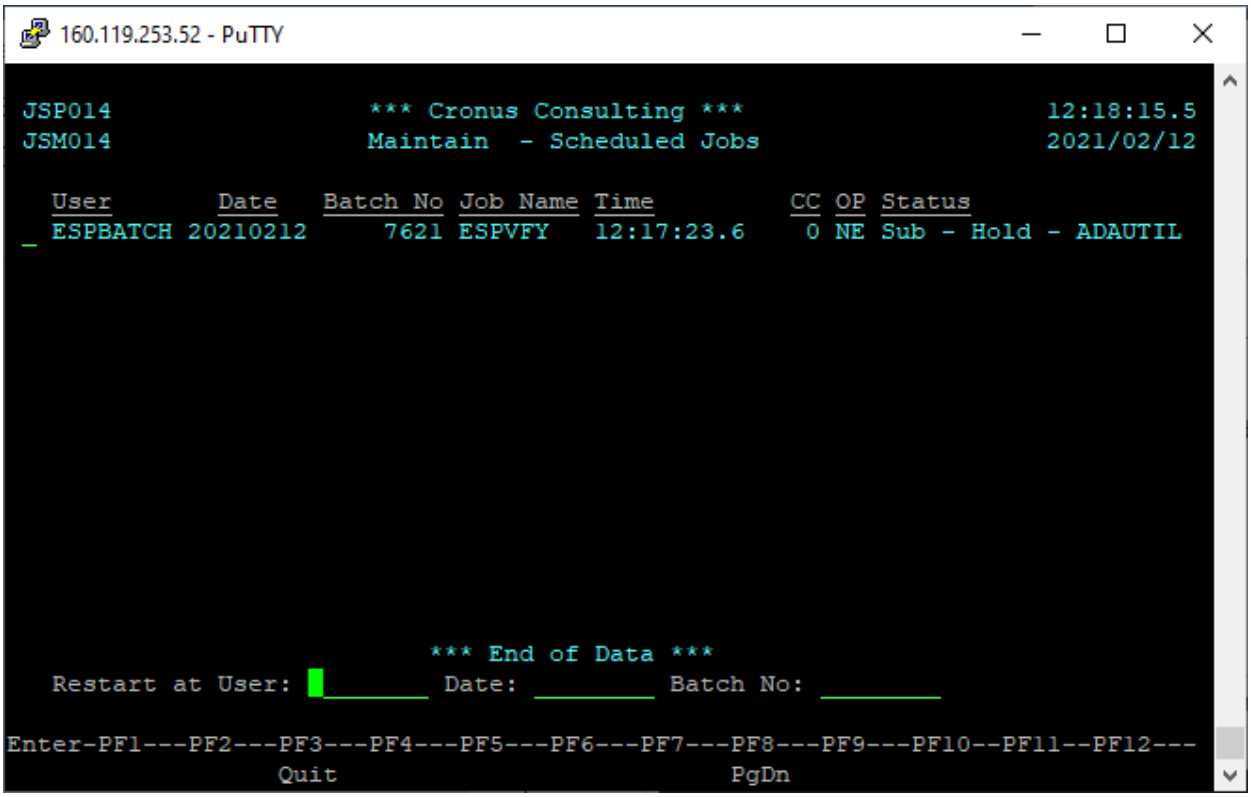


3.9 JS014 Maintain - Scheduled Jobs

All jobs that are currently in the Input Queue will be displayed. A job can now be selected to display all steps for that job.

Function Options:

S - Select Job



After a batch job has been selected a screen is displayed that lists all of the job steps.

The following actions can now be performed on a single/all job steps.

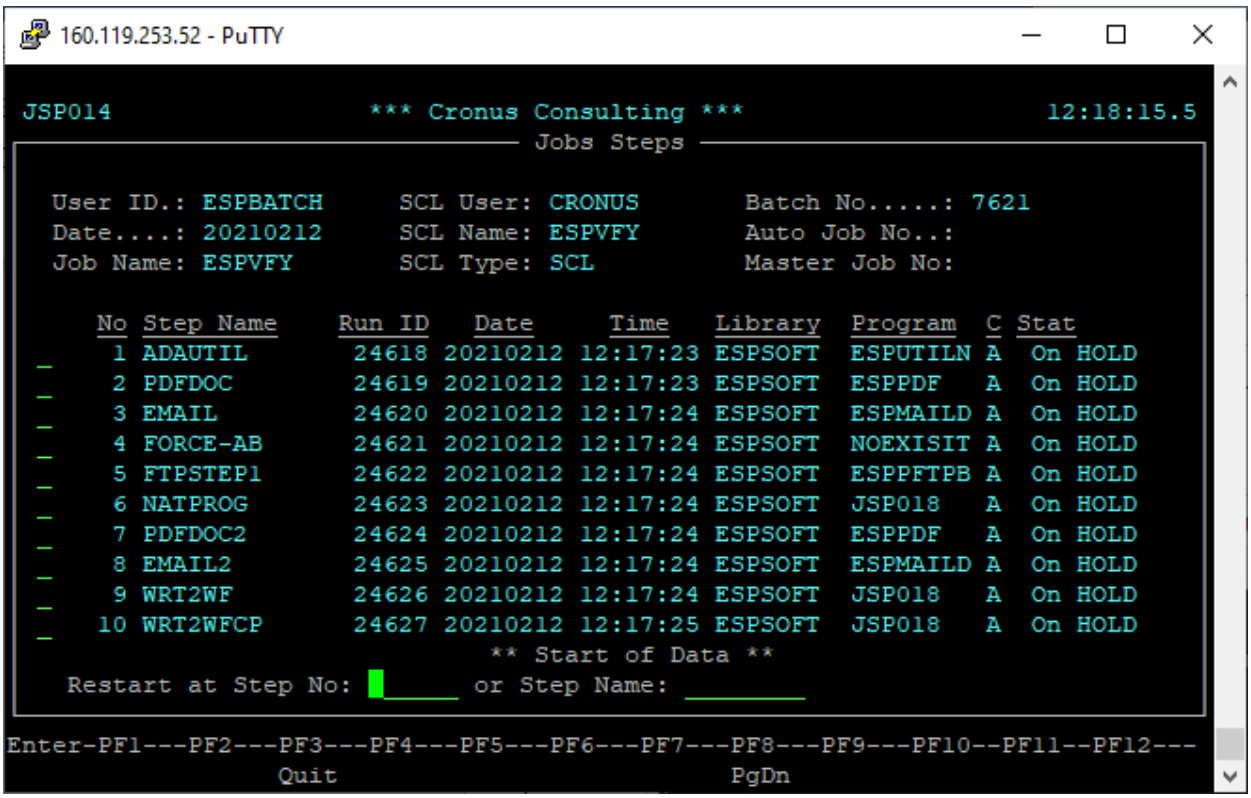
D - Delete Job Step(s)

H - Update HOLD status

S - Skip Job Step. Skipping a job step will allow the subsequent step to execute provided all execution requirements are met.

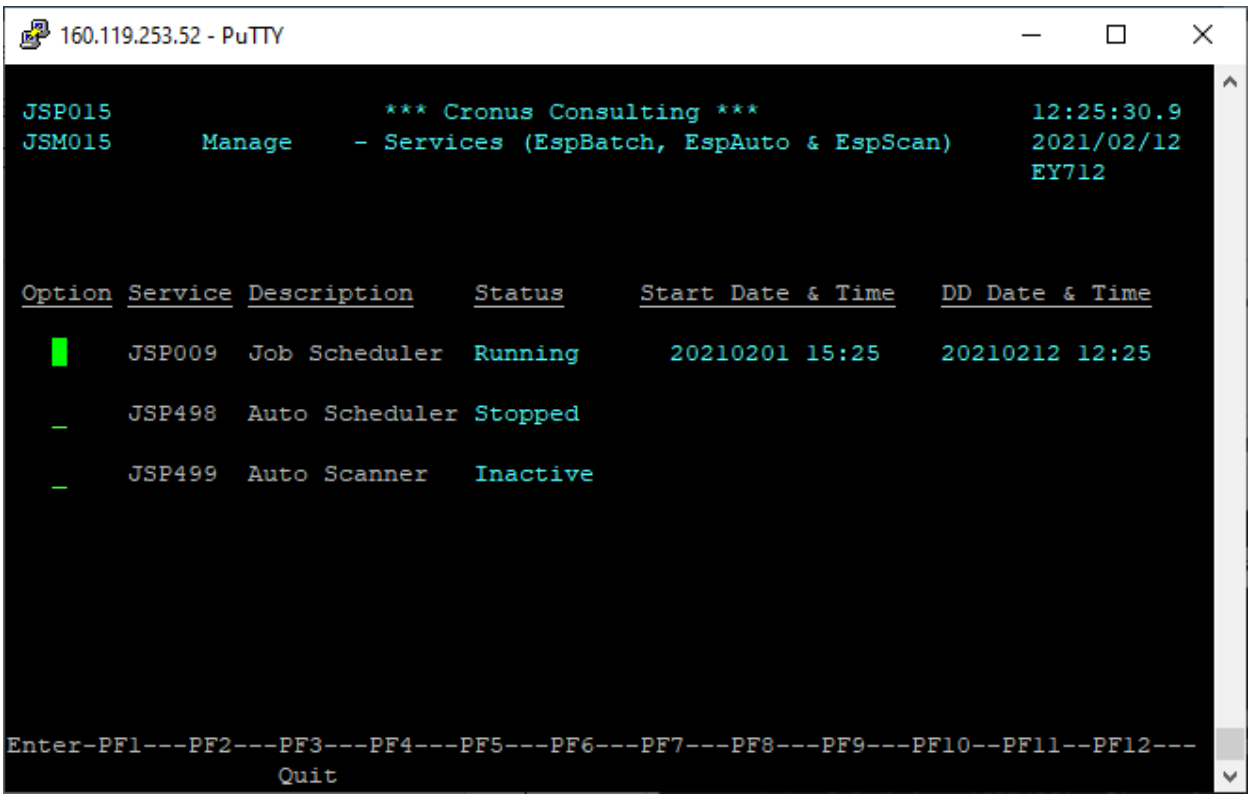
U - Reset UNIX submission Indicator for Job Step. (Option becomes available should job submission fail, this is indicated by a '1' under the "U" column on function **JS012 View - Scheduled and Executing Jobs**)

The above actions can only be performed on steps that have a status of submitted. All other job steps are protected. These actions can also be performed with function **JS050 Maintain - EspBatch Jobs**.



3.10 JS015 Manage - Services (EspBatch, EspAuto & EspScan)

The function is used to Start/Terminate the Batch/Auto Schedulers and Activate/Inactivate the EspAuto Scanner. Note that batch jobs can still be submitted if the scheduler have been terminated; they will however not start execution until the given scheduler has been restarted.



Service Name:

- JSP009 - EspBatch Scheduler
- JSP498 - EspAuto Scheduler
- JSP499 - EspAuto Scanner

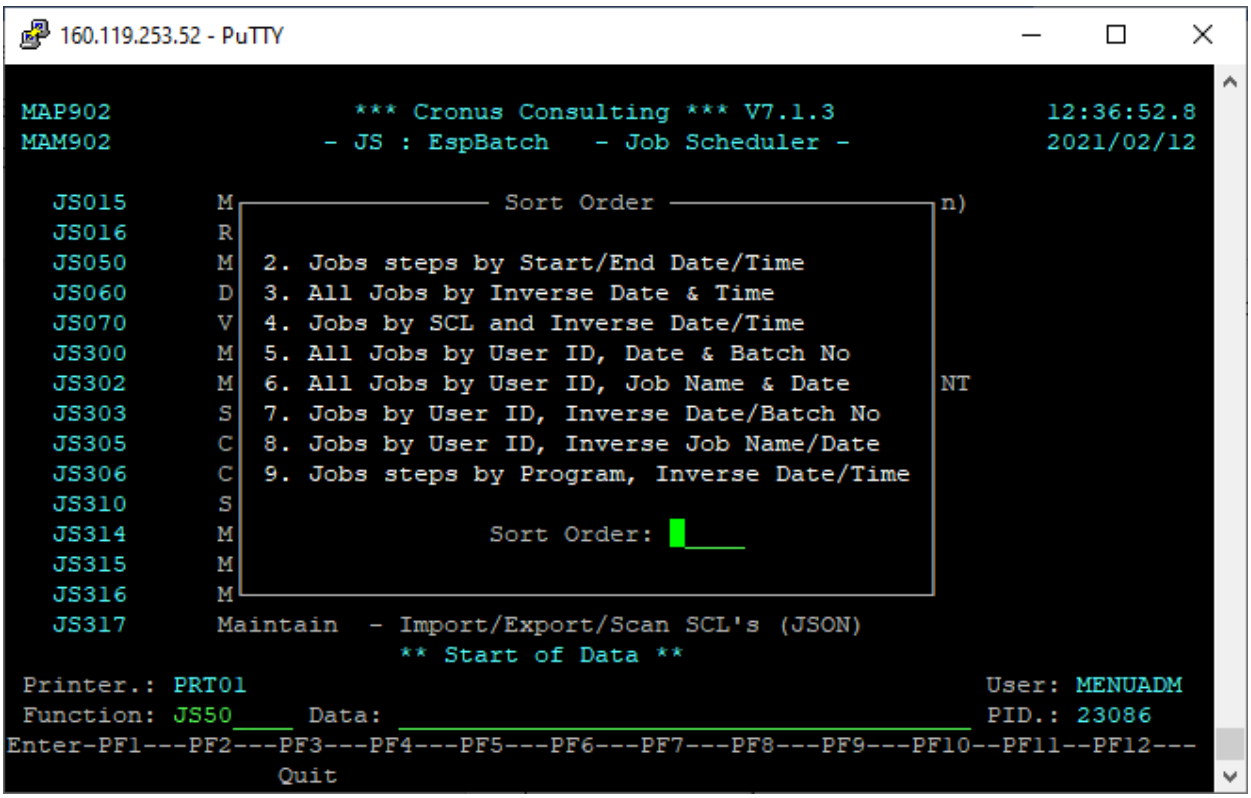
Action:

- S - Start Scheduler
- T - Terminate Scheduler
- L - View log file
- A - Activate Auto Scanner
- I - Inactivate Auto Scanner

A window is displayed after the action has been entered prompting for confirmation.

3.11 JS050 Maintain – EspBatch Jobs

After this function is selected a popup window is displayed upon which the desired batch job sort sequence can be selected.



Function Options:

C - Change Job Class

D - Delete Job

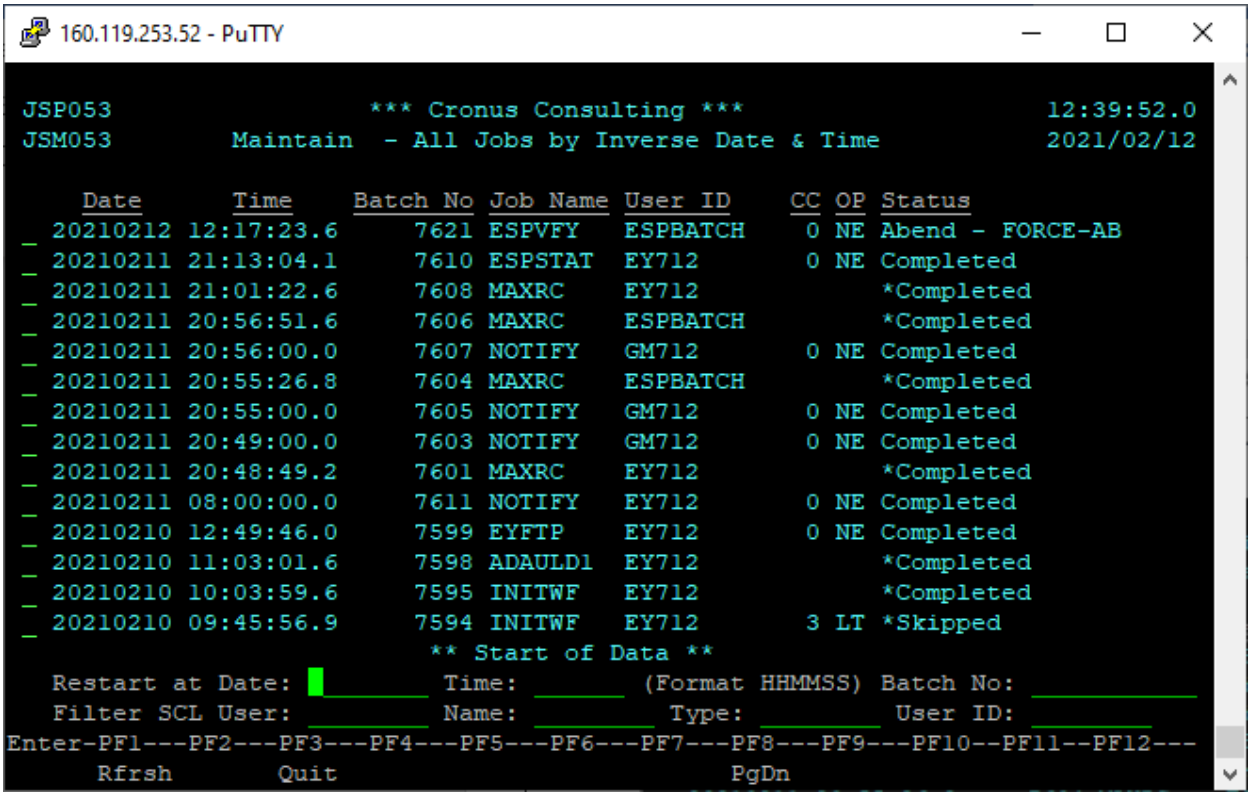
L - Display Log File

S - Select Job

T - Display Exec Times

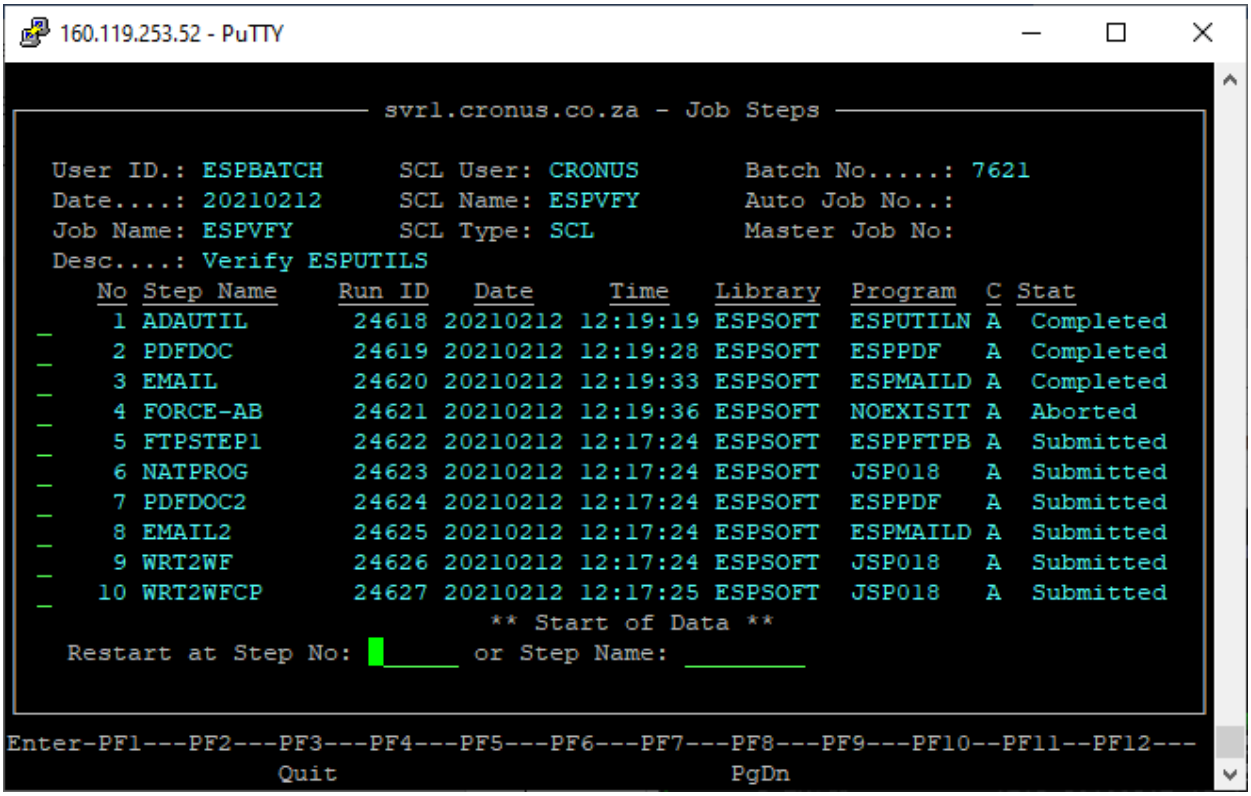
U - Update Scheduled date/Time

R - Select Reports for viewing



Option Selection:

- C A window is displayed that provides the functionality to change the Job Class. Only Jobs that are in a submitted status may be changed. Changes are applied to all job steps
- D A window is displayed requesting confirmation before deleting the Jobs Output.
- L A detailed log is displayed providing environment assignments and job parameters for each step that has executed.
- S All steps of the job will be displayed as shown in the following screen. If the status of a job step is submitted or aborted the parameters, work files and report set up information can be modified.
- T Displays total execution/elapsed time and execution times per job step.
- U A window is displayed that provides the functionality to change the scheduled date and scheduled time. Only Jobs that are in a submitted status may be changed. Changes are applied to all job steps
- R Displays a list of reports that were generated by the Batch Job. For each report, the following options are available:
  - V – View Report
  - D – Delete Report
  - R – Respool Report
  - P – Report parameters



The following options can be performed per Job Step:

- **C** Force complete an aborted job step. Status updated to "***Skipped**". Next job step will start execution.
- **E** Specify conditional logic
- **D** Delete job step – Job steps with a status of Submitted/Aborted will be Deleted.
- **H** Update HOLD status of job step(s).
- **I** Display the run-time parameters for the step such as the class, Batch ID, Run No, start/end date/time, duration, and completion code.
- **L** Display Log File for Job Step
- **O** Update Library/Program
- **P** Parameters (Input and Profile). The word **SPACE** as a parameter indicated that a blank was passed as parameter. After selecting the job step a popup window is displayed whereby the user is required to enter an "**I**" for input parameters or a "**P**" for profile parameters. Input parameters are the parameter(s) passed to the object being executed in the selected job step. Profile parameters contain the values assigned to the various dynamic natural parameters.
- **R** Report Info. Displays printer set up information per report consisting of the Printer Id, Report Name, No of copies, Report Class, Disposition and Form Type. In addition, the distribution information can be viewed by moving the cursor to the line containing the report and pressing PF10.

- **S** Skip Job Step. Skipping a job step will allow the subsequent step to execute provided all execution requirements are met.
- **U** Reset UNIX submit Indicator for Job Step. (Option becomes available should job submission fail this is indicated by a '1' under the "U" column on function **JS012 View - Scheduled and Executing Jobs**)
- **W** All defined work files for the step will be displayed, including the work file number, name, disposition, GDG information and sub system.
- **X** Resubmit Aborted Job Step, this option only applies to job steps that have an aborted status.

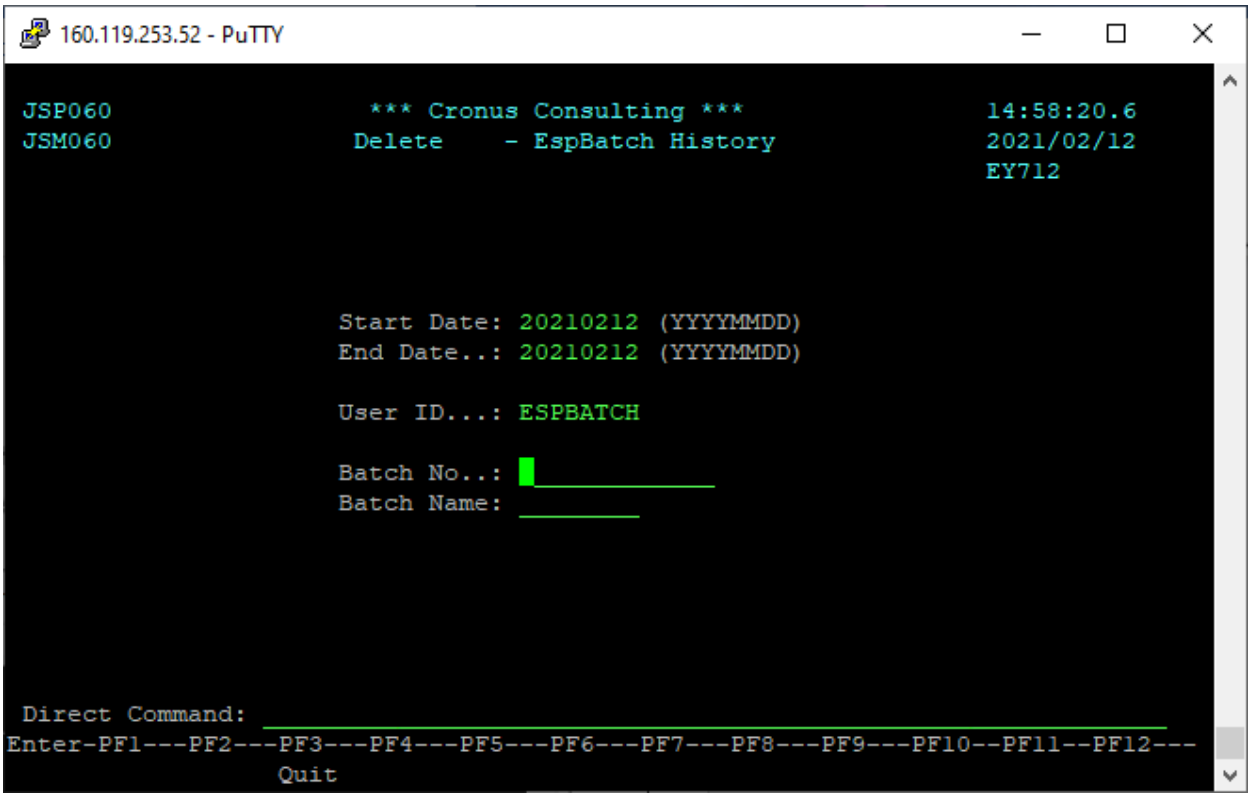
3.12 JS060 Delete - EspBatch History

This function is used to delete batch job history.

The following parameters can be given:

- Start Date
- End Date
- User Id
- Batch No
- Batch Name

Care should be taken when using this function as batch jobs that are currently in the input queue will also be deleted.

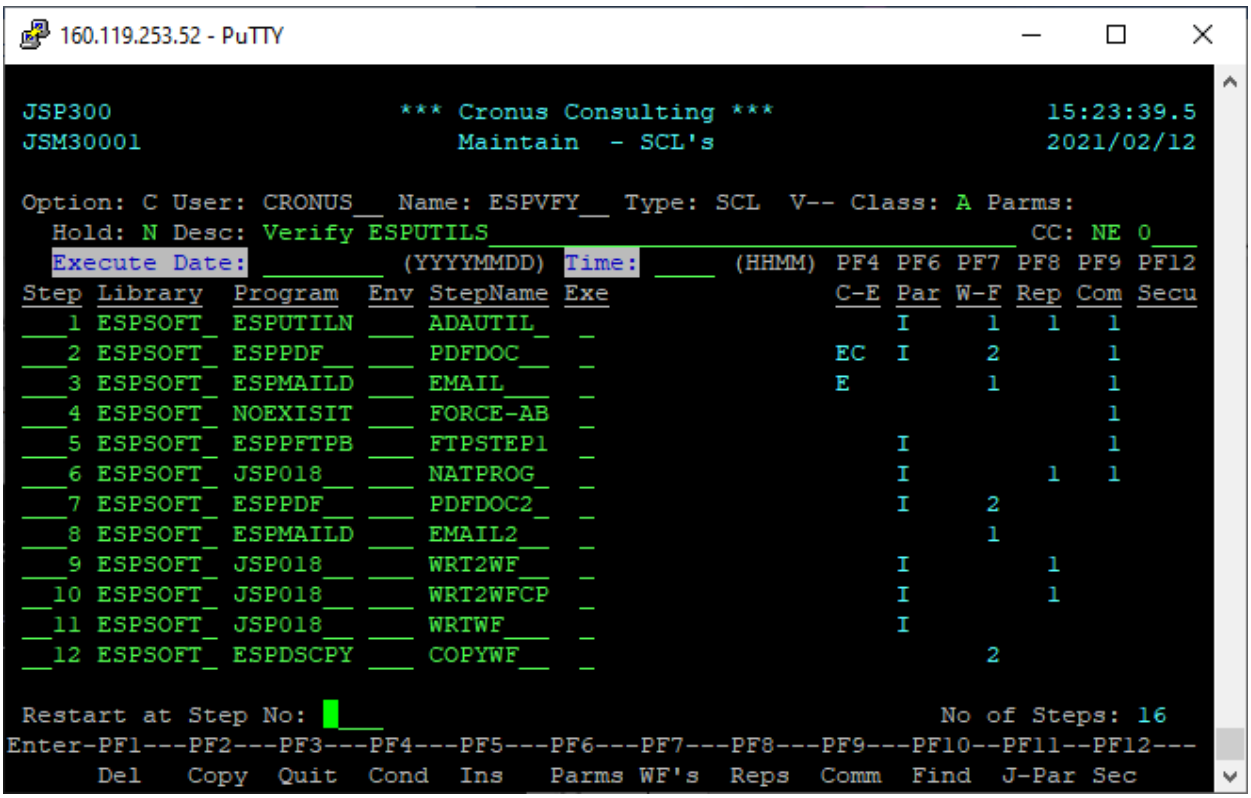


3.13 JS300 Maintain - SCL's

The function is used to maintain batch jobs. At run time each SCL step is converted into shell script that will do the physical submission of a job step.

Function Options:

A - Add SCL
C - Change SCL
D - Delete SCL
E - View SCL
P - Print SCL
R - Rename SCL
S - Submit SCL
U - Unlock SCL
V - Version SCL
X - Copy SCL



To uniquely identify a SCL, the user library, name and type must be specified. This allows developers to make copies of existing SCL for their own use and testing purposes. Take note that the SCL TYPE must be 5 characters or less to allow for versioning of SCL's

The 'V' option is used to create a backup version of the specified SCL. SCL versions are identified by **Vnn** (where **nn** is the version number) as characters 6-8 in the SCL TYPE field. SCL versions may not be changed or deleted. Version V01 will always be the most recent version of the SCL. Up to 10 SCL versions can be created.

A SCL may be left in a locked state, if this occurs the 'U' option must be used to unlock the SCL. Take care not to unlock a SCL that is currently being edited by another user.

The input screen consists of 2 sections. The top of the screen contains the batch jobs header information, while the lower part consists of the job steps.

Header information:

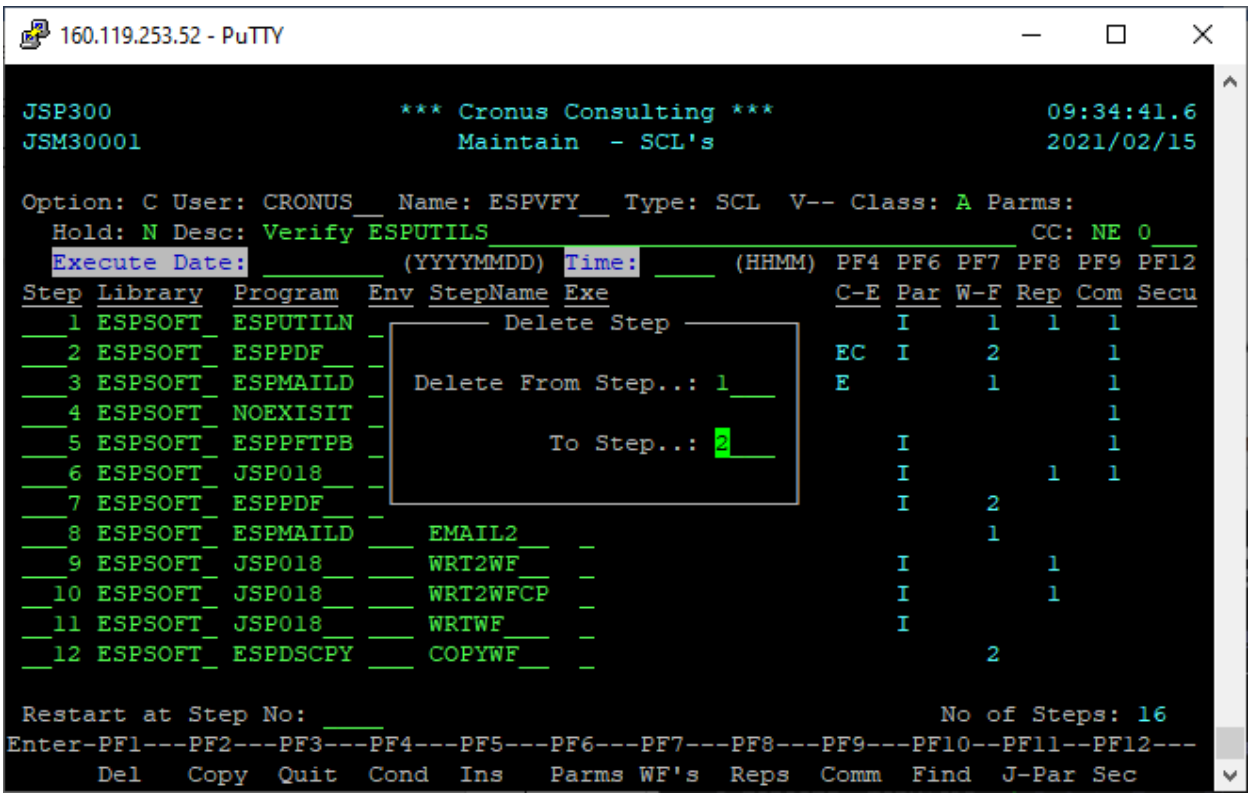
- Class - Class in which batch job will be submitted.
- Parms - Indicates if header parameters have been specified.
  J – Job Parameters using PF11
  P – Profile parameters using PF6 with a step no of 0
  D – Dynamic parameters using PF6 with a step no of 0
- Hold - If **Y**es is entered the job will be submitted in HOLD status and will not run until the HOLD is removed. A value of **N**o will ensure that the job starts running as soon as an initiator becomes available.
- Desc - A brief description of what the job is intended to do.
- CC - Condition Code operator and value. If a condition code is specified as part of the job header and any step completes with a return code value that does not correspond with the condition code specified, the job will abort at that step.
- Execute Date - Natural will be invoked with the date specified. Thus if 20201231 is specified in this field, the job will execute with a system date of 20201231. Note that the Execute Date only applies to Natural Date/Time system variables.
- Execute Time - Natural will be invoked with the time specified. Thus if 1300 is specified in this field, the job will execute with a system time of 13H00. Note that the Time only applies to Natural Time system variables.
  N.B: If only the execute date or time is specified, the system date/time will be substituted for the value that has been omitted.

Step Detail:

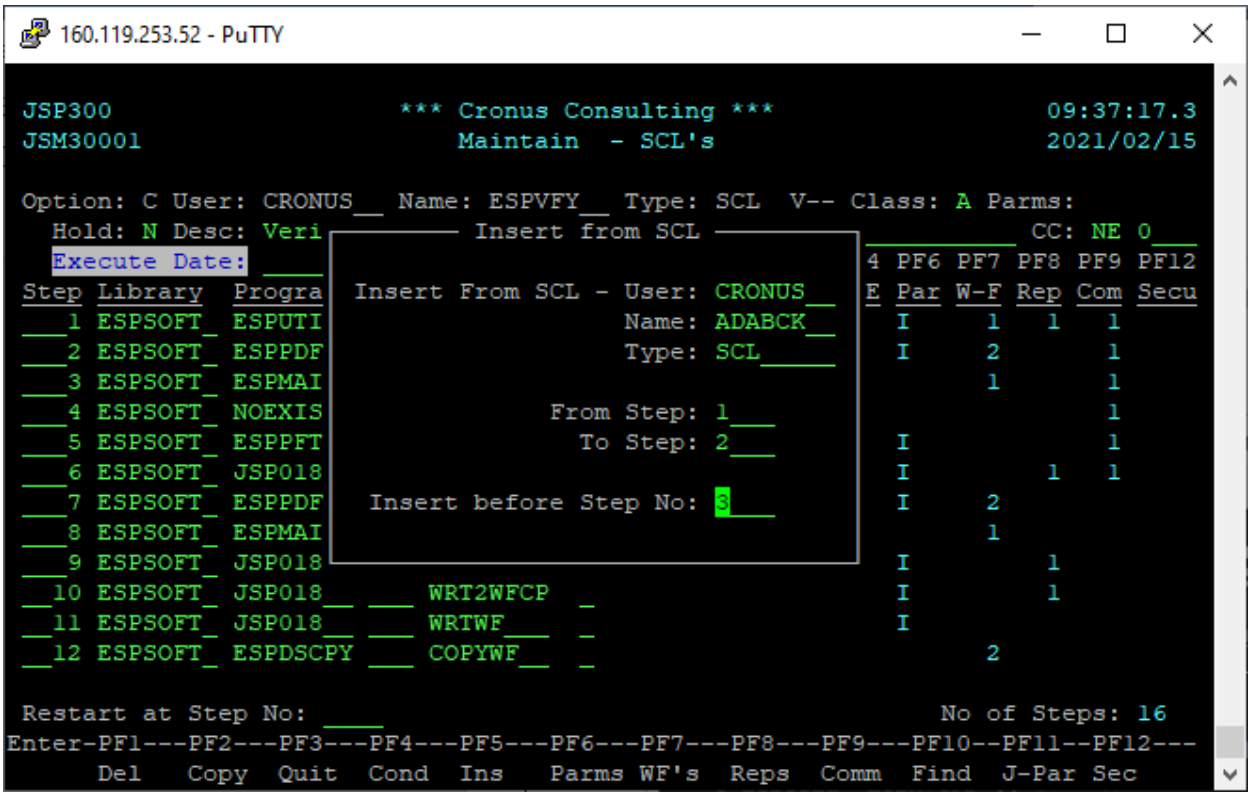
- Step - Step Number. The step numbers are automatically reordered into numeric sequence if step(s) are added or deleted.
- Library - The name of the library that contains the program/utility to be executed. When executing a PROC the library must contain the SCL user library of the PROC.
- Program - The name of the program/utility to be executed.
  When executing a PROC the program must contain the SCL name of the PROC.
- Env - The environment used to execute the job step. Environments are defined using function **JS002 Maintain - EspBatch Codes** with Code Type: **ENV**. If the environment is not specified, the default EspBatch environment is used.
- StepName - Name used to identify a job step. Must be unique for each SCL step.
- Exe - Valid values are (**Y**)es and (**N**)o. If (**Y**)es is specified, the job step is executed. If (**N**)o is specified, the job step is not executed. If no value is specified, the step will be executed.
- PROC - PROC is displayed between the "**Exe**" and "**C-E**" columns if the job step executes a proc.
- C-E Conditional execution. Allows for the specification of conditional and/or completion logic per job step.
- Par - Three types of parameters (**D**)ynamic, (**I**)nput and (**P)**rofile/Session) can be defined per job step. These are indicated by the letters **D**, **I** and **P** in the 'Par' column.
- W-F - Indicates the number of work files that have been defined for the job step. A blank is displayed if no work files have been defined.
- Rep - Indicates the number of reports that have been defined for the job step. A blank is displayed if no reports have been defined.
- Com - Indicates the number of comment lines that have been defined for the job step. A blank is displayed if no comments have been defined.
- Secu - If an **X** is displayed in the column it indicates that a Natural Security User ID and Password has been defined.

Function Keys:

- PF1 - Delete one or more job steps. If the **"To Step"** field is left blank, only the step specified in the **"Delete From Step"** will be deleted. In the example below steps 1 and 2 will be deleted.



- PF2 - Copy a step from the current or a different SCL and insert it before the specified step no. When copying a step from the current SCL both the **"Step No"** and **"Insert Before Step"** fields must be completed. When copying from a different SCL, the "**SCL User**", "**SCL Name**", "**SCL Type"**, "**From Step**", "**To Step"** and "**Insert before Step No"** fields must be completed. In the example below steps **1** and **2** will be copied from SCL "**CRONUS.ADABCK.SCL**" and inserted before step 3.

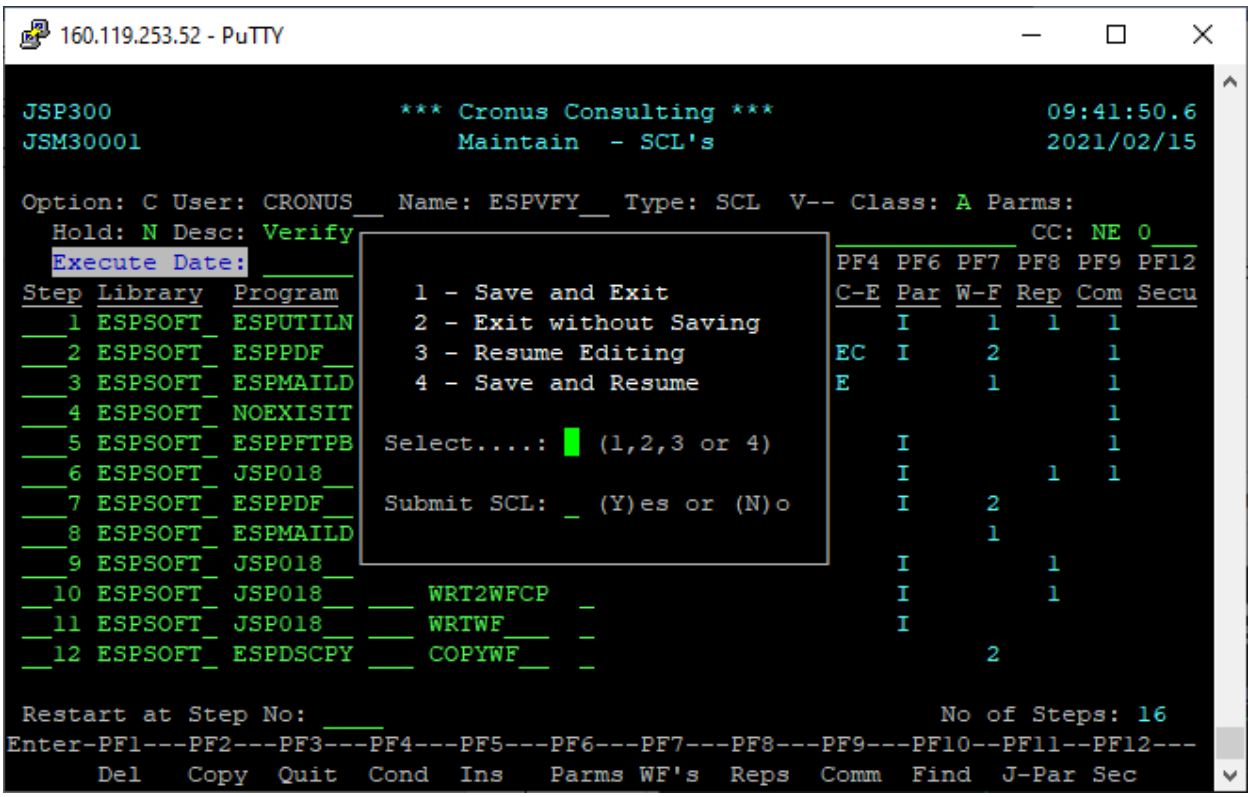


- PF3 - Return to SCL selection.

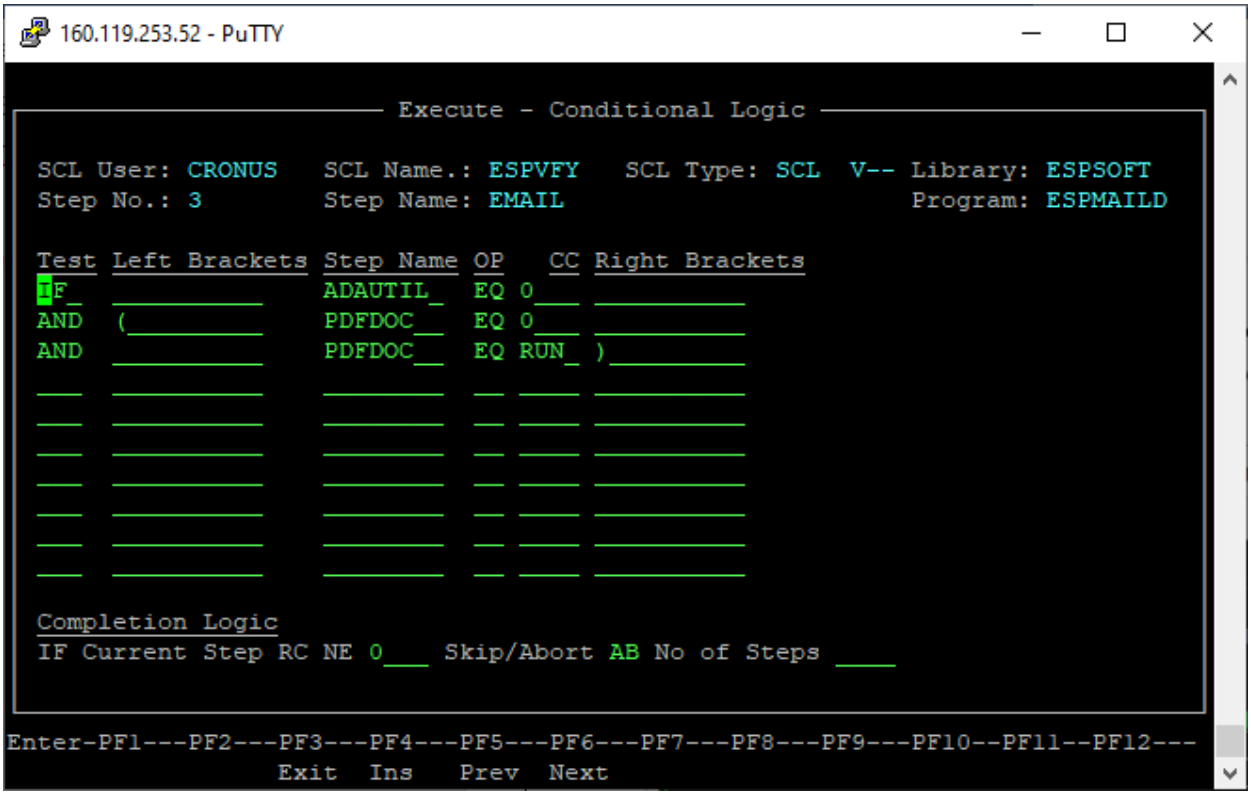
  If PF3 is pressed whilst adding or changing a SCL, the following window is displayed. This allows the user the following options:

  - Save SCL and exit to SCL selection
  - Exit to SCL without Adding/Changing SCL
  - Resume editing
  - Save SCL and resume editing

  In addition, the SCL can be submitted, thus providing the ability to submit an SCL without saving the changes or adding the SCL.

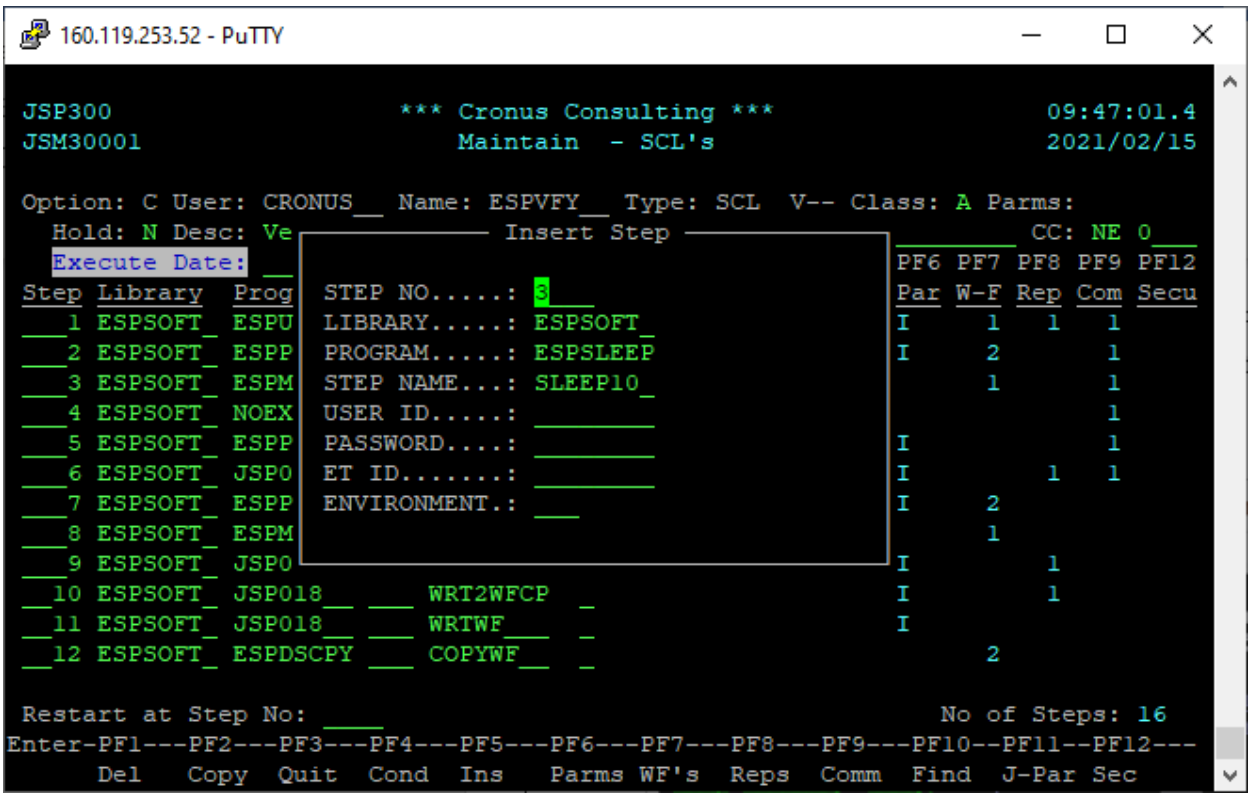


- PF4 - Specification of conditional and completion logic. Conditional logic is evaluated before the step is executed against the highest return code produced by previously executed steps. If the conditional logic does not evaluate to TRUE, the step is bypassed. Completion logic is evaluated on completion of a job step using the return code of the current job step.

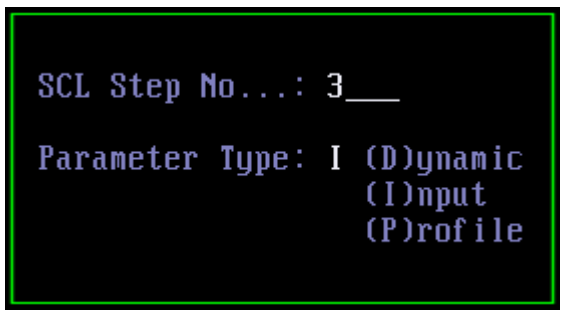


In the example above both conditional and completion logic have been specified. The conditional logic specified ensures that the current step (**EMAIL**) will only be executed if the return code of **ADAUTIL** is **0**, the return code of **PDFDOC** is **0** and **PDFDOC executed**. The completion logic will result in the job aborting if the **current step** does not complete with a return code of **0**.

- PF5 - Used to insert a new step between existing steps. Required fields are the Step No, Library, Program and Step Name. In the example below the new step will be inserted after step 2.



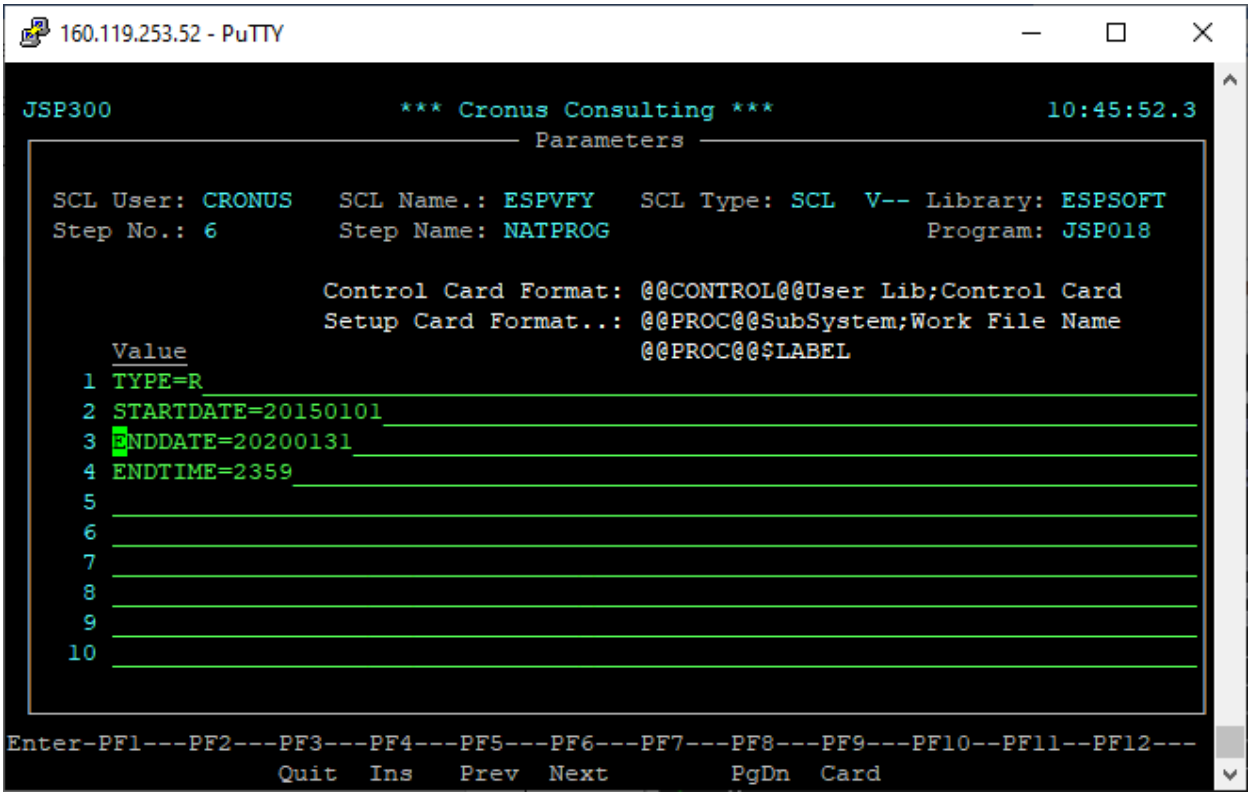
- PF6 - Activates the parameter specification screen. Dynamic, Input and Profile parameters can be specified. After PF6 is pressed a window is displayed requiring the step number and parameter type. By omitting the step number Dynamic and Profile parameters can be specified for the entire SCL.
  Dynamic parameters are used to define hardcoded values that can be used throughout the SCL for dynamic parameter substitution.
  Refer to Software AG's Natural documentation for a description of the allowable natural parameter module parameter values.



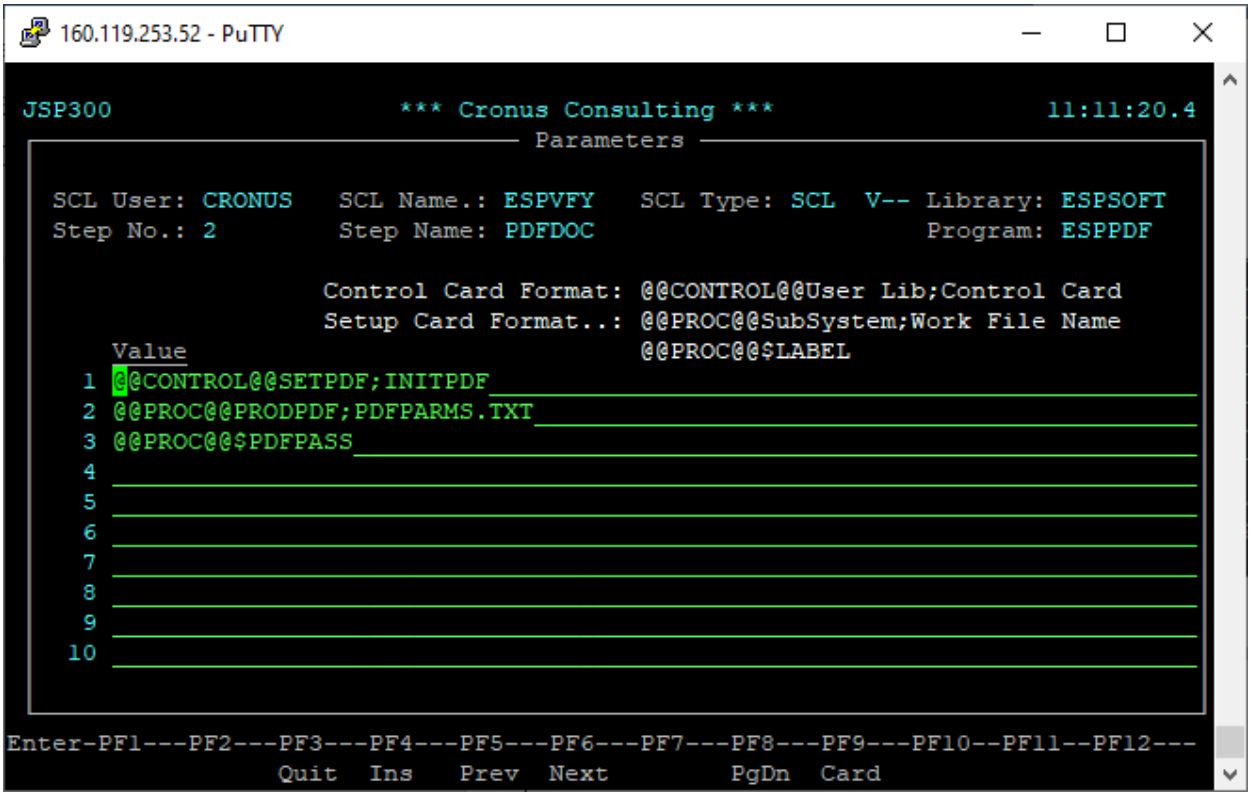
For input parameters a maximum of 180 parameters may be specified per job step. If a parameter with a blank "" value needs to be passed, the word **"SPACE"** must be entered and will be translated into a blank value at execution time. The **PF7** and **PF8** keys can be used to page backward and forward. If the parameters are contained within a setup/control card, the parameter line must be defined as follows:

"@@CONTROL@@User Lib;Control Card"
"@@PROC@@SubSystem;Work File Name"

Control Cards are expanded at submission time and Setup Cards at at execution time.
Control/Setup cards can be viewed by repositioning the cursor on the parameter line and pressing **PF9**.



Function key **PF4 (Ins)** can be used to insert parameter values. To insert an input parameter, position the cursor on the line at which you want to insert the parameter and press **PF4**.



In the example above the parameters passed to program **ESPPDF** are contained within a **control card** and 2 **setup cards.**

Control Card:

This is denoted by the @@**CONTROL**@@ followed by the sub system that contains the control card (**SETPDF**), a semi-colon (**;**) and the name of the control card (**INITPDF**).

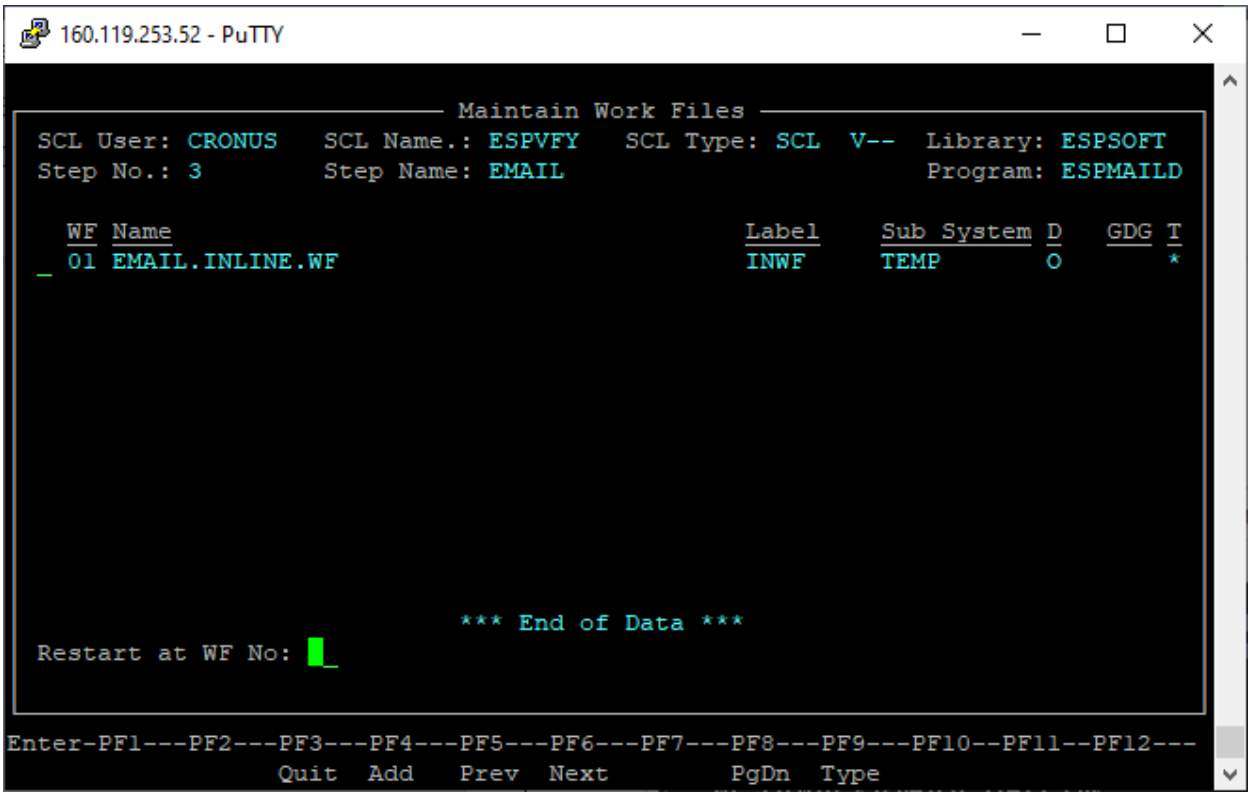
Setup Card 1:

This is denoted by the @@**PROC**@@ followed by the sub system that contains the setup card (**PRODPDF**), a semi-colon (**;**) and the name of the setup card (**PDFPARMS.TXT**).

Setup Card 2:

This is denoted by the @@**PROC**@@ followed by **$LABEL** (**MYPDFPARM**). The label corresponds with the label given to a work file that is defined within the current job step.

- PF7 - Activates the work file selection screen. A maximum of 96 work files may be specified per job step.
  Valid options are:
  - D - Delete work file
  - P - Modify work file parameters
  - V - View work file
  - PF4 - Add work file. When adding a new work file a window is displayed in which the work file number must be entered.



If the "**T**" column contains an '*****' it indicates that the work file specification contains inline text.

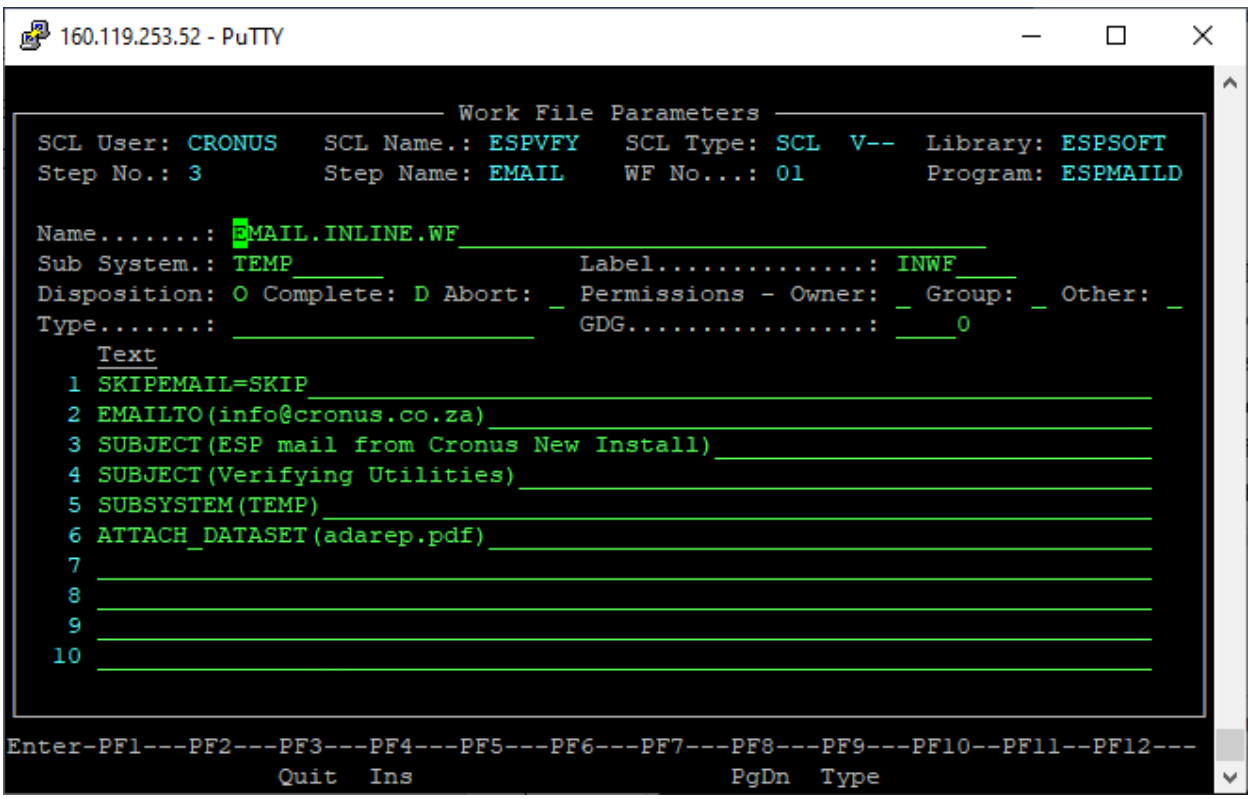
Parameters:

Work file parameters can be modified by specifying option (P)arameters. To add an additional work file press PF4.

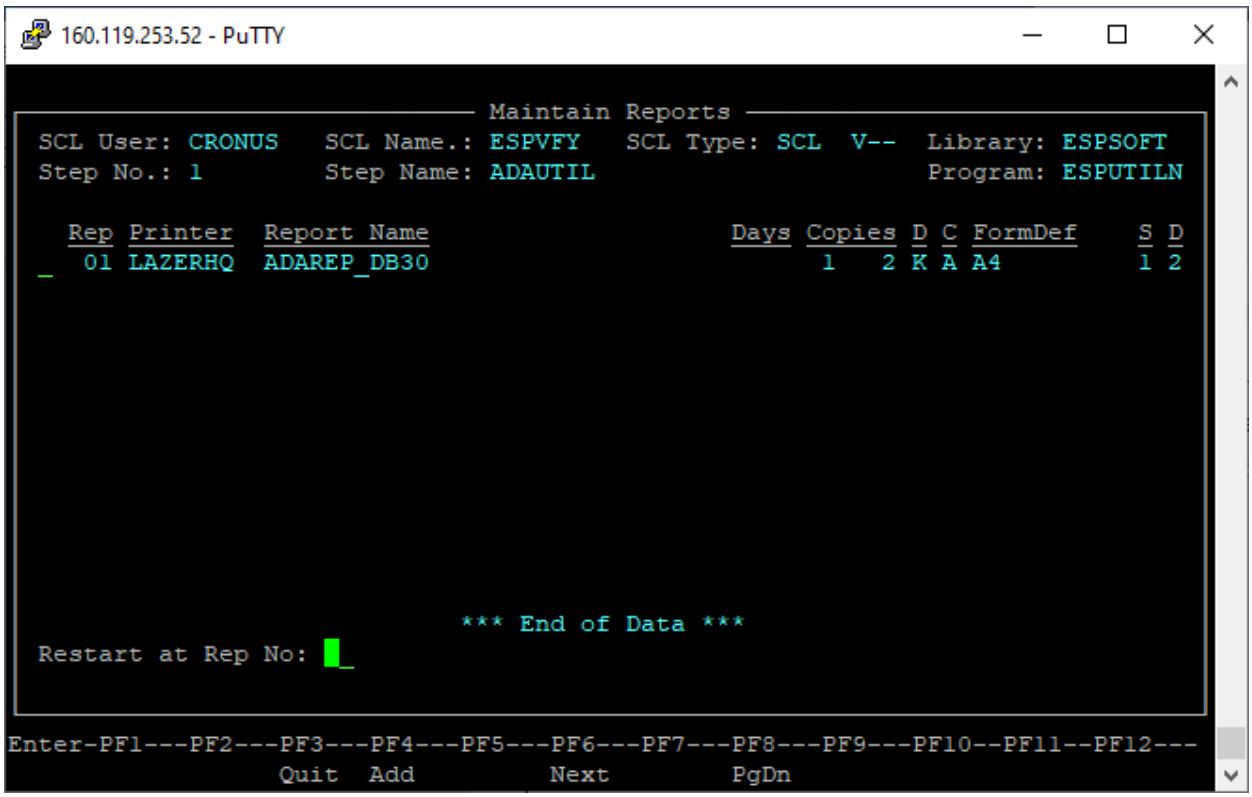
The following parameters may be specified:

- Name - File Name may not contain spaces. The file name can be specified as "/DEV/NULL" which will result in discarding any data being written to the work file.
- Sub System - File system path where work file is located. Sub systems are defined using function **JS002 Maintain – EspBatch Codes** with **Code Type** "SUBSYSTEM".
- Label - If a label is specified then an environment variable that contains the work file path and name is set within the script used to submit the job step. Labels are only required for use by certain of the ESP utilities - an explanation of the required labels can be found in the EspRoutines manual. By default, an environment variable consisting of NATWKnn is defined where nn is replaced by the specified work file number.
- Disp -
  - N – New: If a disposition of new is specified the work file content is deleted before execution of the program/utility specified in the job step. Even if the program/utility does not access the work file or only accesses it in READONLY mode the previous content will be lost.
  - M – Mod: Work file is appended to.
  - O – Old: If a disposition of old is specified and the program/utility writes to the work file the content will be overwritten.
  - P - Print: If the work file disposition is specified as print, the work file will be routed to the printer. The work file name will be assigned to the Report name.
- Complete - Action to be applied to the work file if the step completes successfully.
  - K – Keep: Work file is retained
  - D – Delete: Work file is deleted
- Abort - Action to be applied to the work file if the step aborts.
  - K – Keep: Work file is retained
  - D – Delete: Work file is deleted
- Permissions - Set the file permissions when a new work file is created. Refer to documentation of operating system command "chmod".
  - Owner: Allowable values are 6 & 7. If no value is specified, the default file creation mask (umask) is used.
  - Group: Allowable values are 0 thru 7. If no value is specified, the default file creation mask (umask) is used.
  - Other: Allowable values are 0 thru 7. If no value is specified, the default file creation mask (umask) is used.

- Type - Specifies the work file type. If no work file type is specified then the work file type is set to “ASCII” except if the work file name ends with a “.SAG” extension it is set to work file type “SAG”.
- GDG - Blank: No GDG specification
  - +1: Create next version of file. May not be specified if disposition is **O**ld.
  - -nnnn: Where n is the version that will be accessed. May not be specified if disposition is **N**ew.
- Text - Creates a work file that contains the text specified. The work file is created at execution time. The work file disposition must be specified as (**O)**ld to create an inline work file.
  To reference a specific GDG version, it must be defined within the work file text as **GDGVERSION=nnnn.** This is only allowed for work files with a disposition (**O**)ld.



- PF8 - Activates the report selection screen. Parameters for reports1 to 31 may be specified per job step.
  Valid options are:
  - C - Distribution copies
  - D - Delete report parameters
  - P - Modify report parameters
  - S - Split report
  - V - View report (Only available if disposition is “W”)



Distribution copies:

A single report can be routed to multiple destinations. The number of distribution copies defined for the report is displayed under the ‘D’ column.

Delete report:

If a report is deleted, all additional distribution copies and report split info is also deleted.

Parameters:

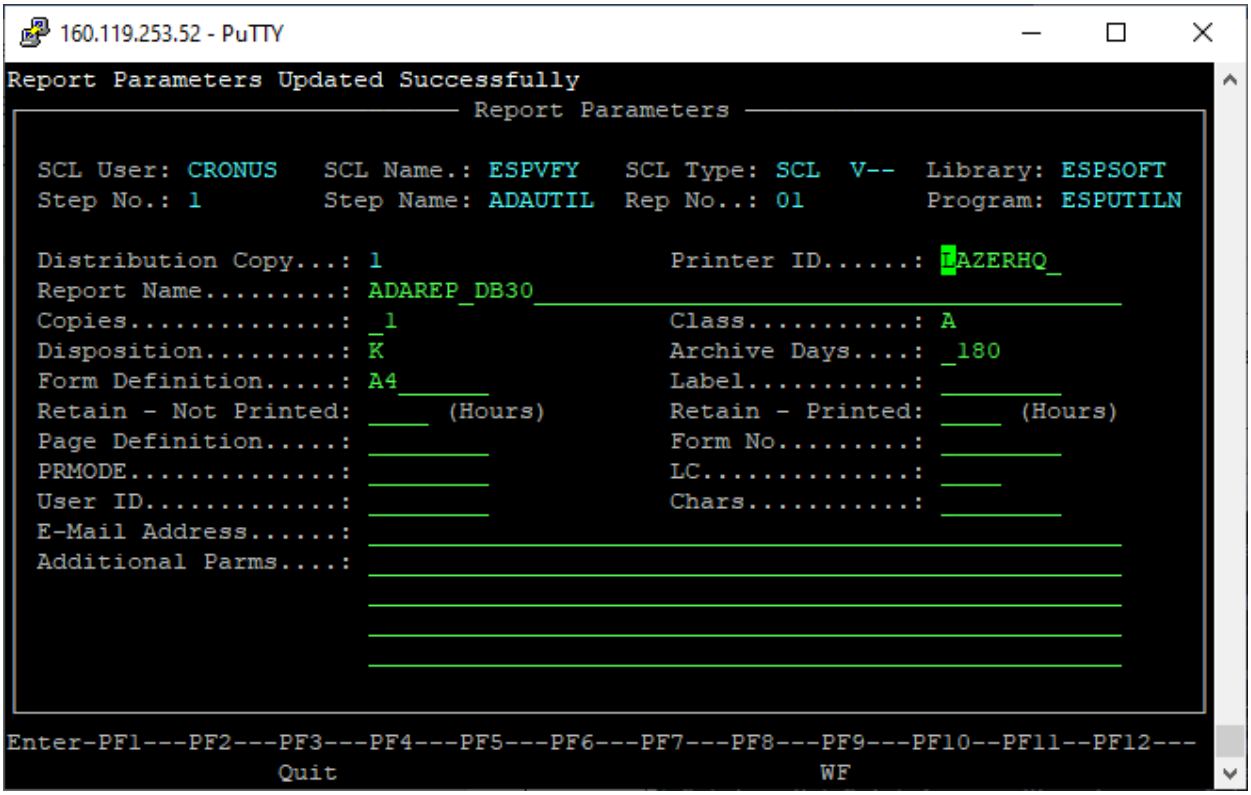
Report parameters can be modified by specifying option (P)arameters. To add additional report parameters, press PF4.

PF9 – Switch between standard reports and reports that are created as a work file. The parameters required differ for standard reports and reports created as work files.

The following parameters may be specified (Standard Reports):

- Printer ID - Name of physical printer as defined in operating system. If the Printer ID specified is defined on function **JS004 – Maintain EspBatch Printers**, it is replaced with the Physical Printer ID & Form Type defined for the given logical printer.
- Report Name - Name of report in spool queue.
- Copies - Number of copies to be printed.
- Class - The class that the report must be routed to.
- Disposition - Report disposition may be defined as follows:
  - **H** – Hold: Report submitted in Hold status. Requires manual release before report will print. Once the report has printed it will be deleted from the print queue.
  - **K** – Keep: Retain report after printing.
  - **L** – Hold & Keep: Report submitted in Hold status. Requires manual release before report will print. Once the report has printed it will be retained in the print queue.
  - **D** – Delete: Delete report after printing.
- Archive Days - The number of days to retain the archived report before it is deleted. If a value of 9999 is specified, the report is not archived.
- Form Definition - The form definition to be used for printing.
- Label - Reserved for creation of COBOL reports.
- Retain – Not Printed The number of hours the report will be retained in the print queue if it has not printed.
- Retain – Printed - The number of hours the report will be retained in the print queue if it has printed.
- Page Definition - The page definition to be used for printing.
- Form No - The form number to be used for printing.
- PRMODE - The PRMODE to be used for printing.
- LC - The line count parameter to be used for printing.
- User ID - User that report will be spooled as.
- Chars - The Chars parameter to be used for printing.
- E-Mail Address - Email address.

- Additional Parms - Free format text for specification of additional parameters.



The following parameters may be specified (Work Files):

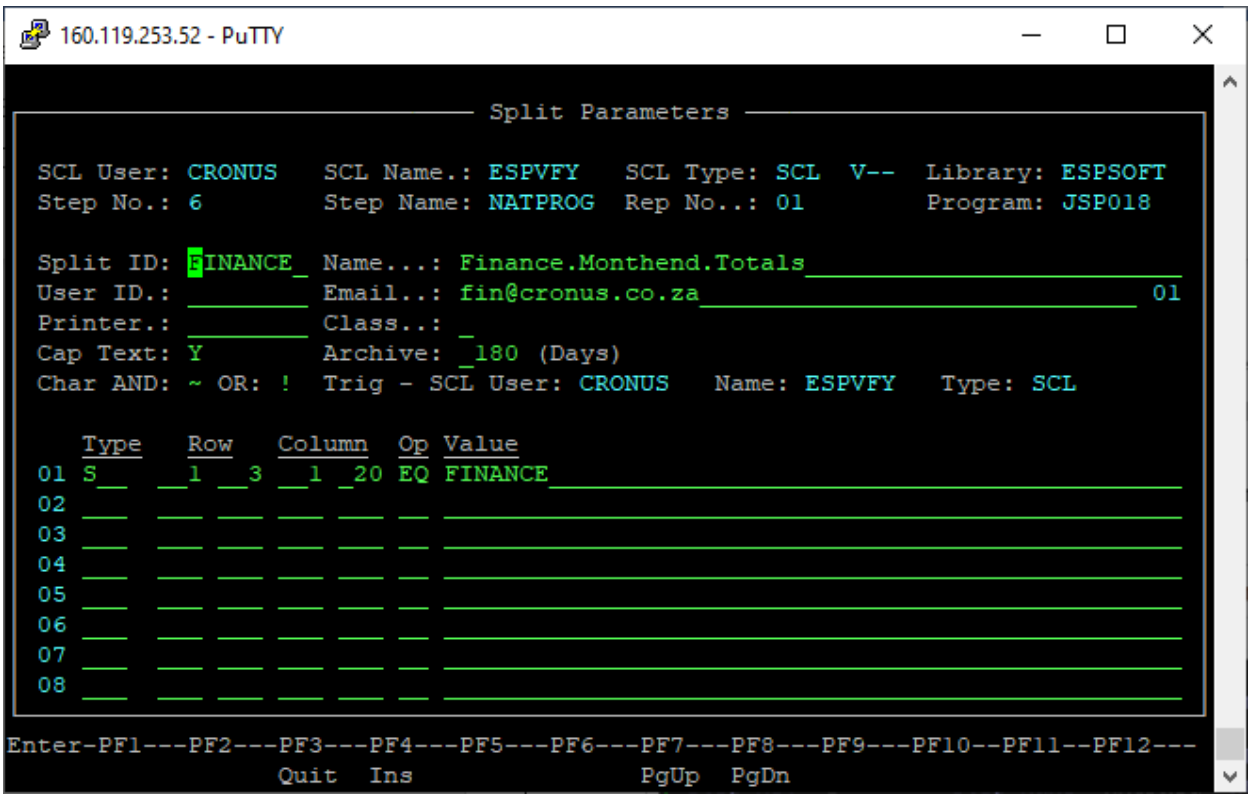
- WF Name - Name of work file.
- Sub System - Name of the sub system where the work file will be created.
- Disposition - W- New
  M - Mod
- Complete - Action to be applied to the work file if the step completes successfully.
  K – Keep Work file is retained
  D – Delete Work file is deleted
- Abort - Action to be applied to the work file if the step aborts.
  K – Keep Work file is retained
  D – Delete Work file is deleted
- GDG - Valid values are 0 and 1. If a value of **1** is specified the work file is created as a GDG.

If any of the report parameters are omitted the default values specified on function **JS002 Maintain - EspBatch Codes** with Code Type: **REPORT** and Code Value: **DEFAULTS** are used.

Splitting a report:

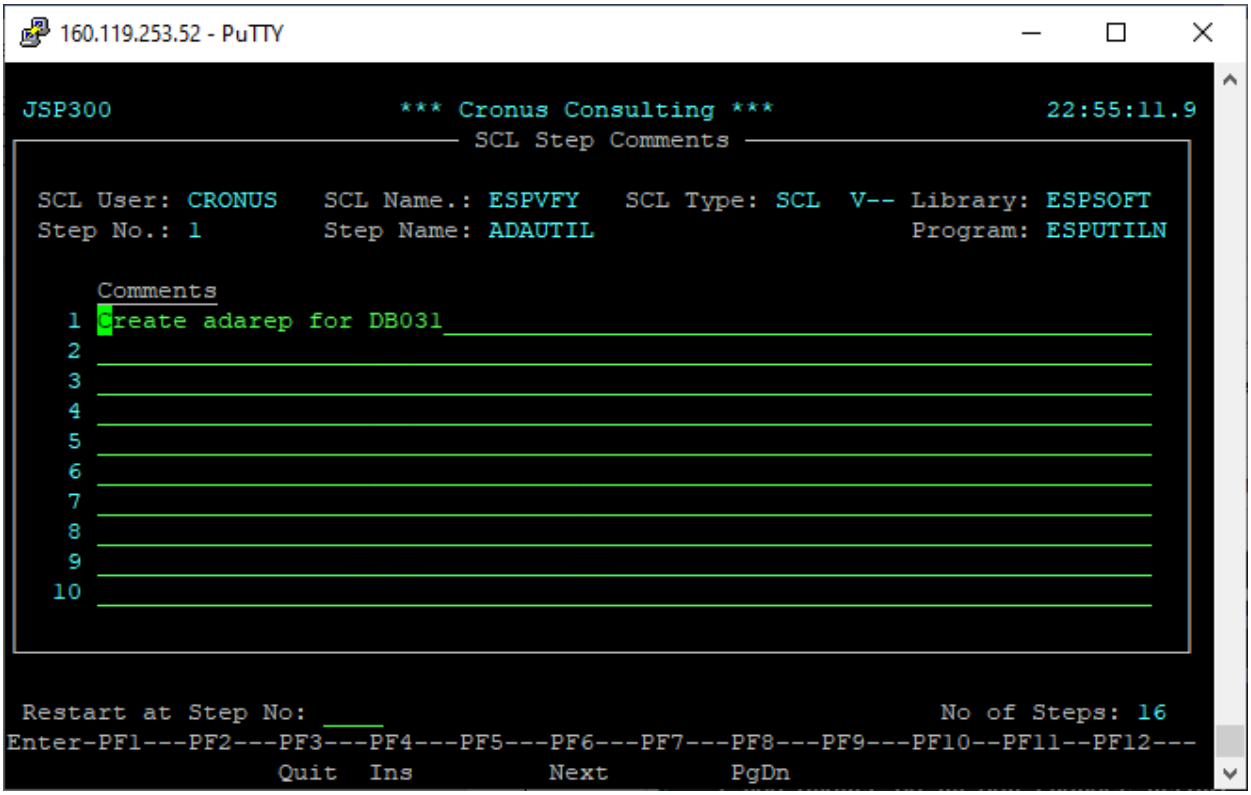
A single report can be split in multiple reports be defining split parameters. The number of split parameters that have been defined for the report is displayed under the 'S' column.

Reports will only be split is the **Printer ID** of the master report is defined as '**espsplit**'.

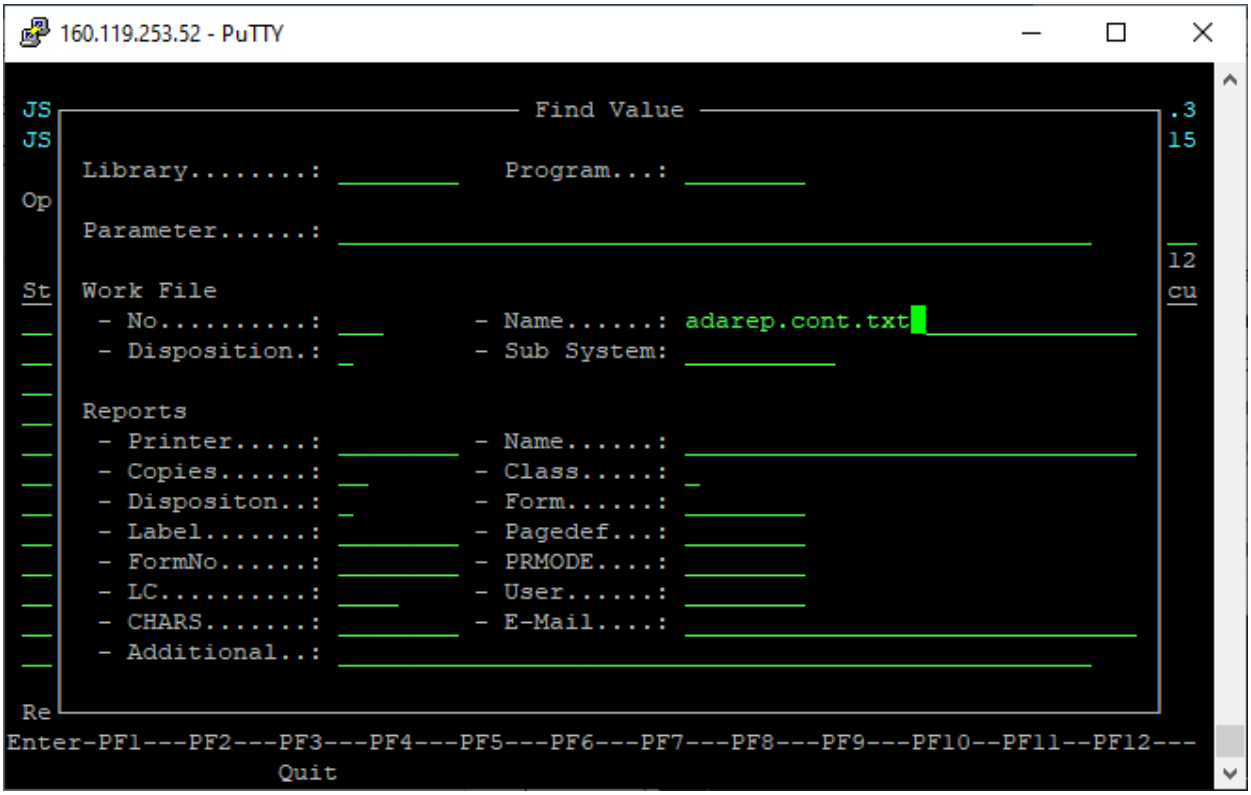


In the example above and additional report "Finance.Monthend.Totals" is created if lines 1-3 columns 1-20 contain the value "FINANCE".

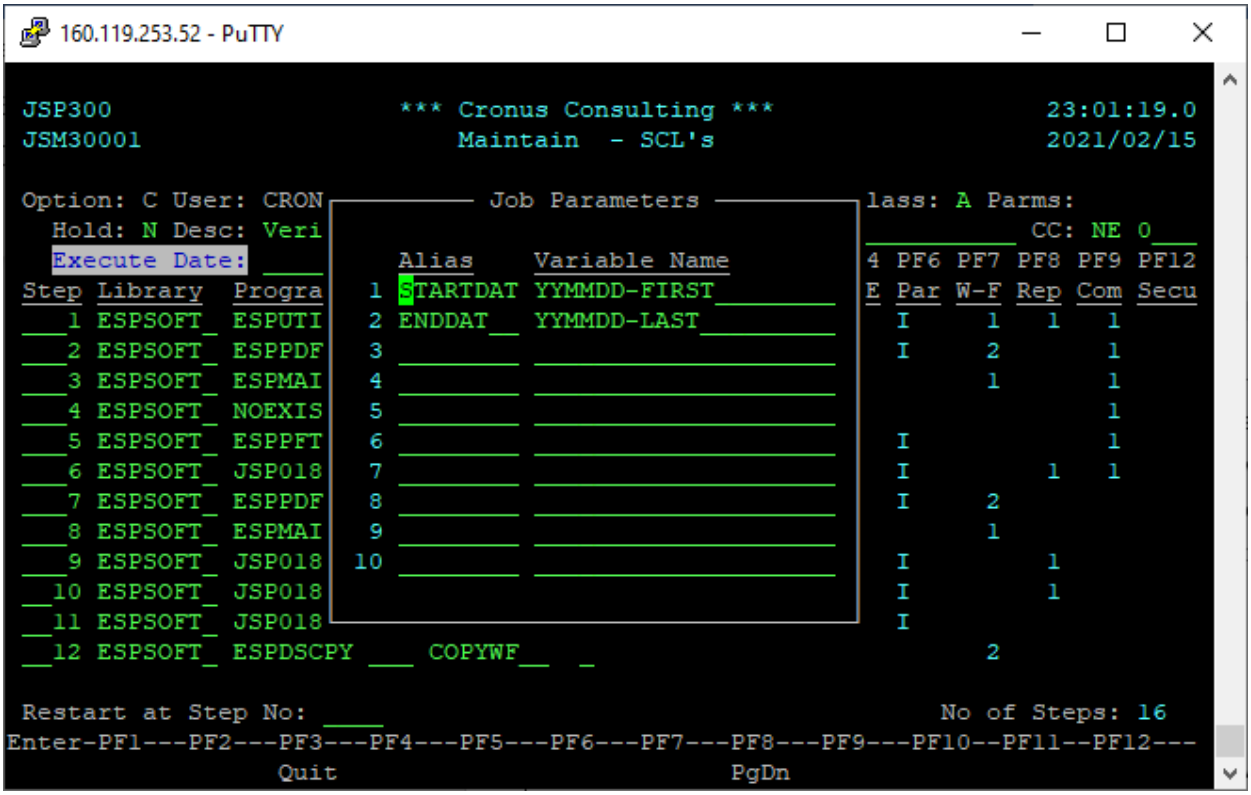
- PF9 - Activates the comment specification screen. Allows for 100 comment lines per job step. Header level comments can be specified by omitting the SCL Step No. Use PF7 and PF8 to page backwards and forwards.



- PF10 - Activates the Find Value specification screen. Any single field may be specified and the complete SCL will be scanned for the value entered. If the value is found a screen will be displayed showing all steps containing the value entered. Function **JS003 Scan - SCL's** can be used to scan multiple SCL's.

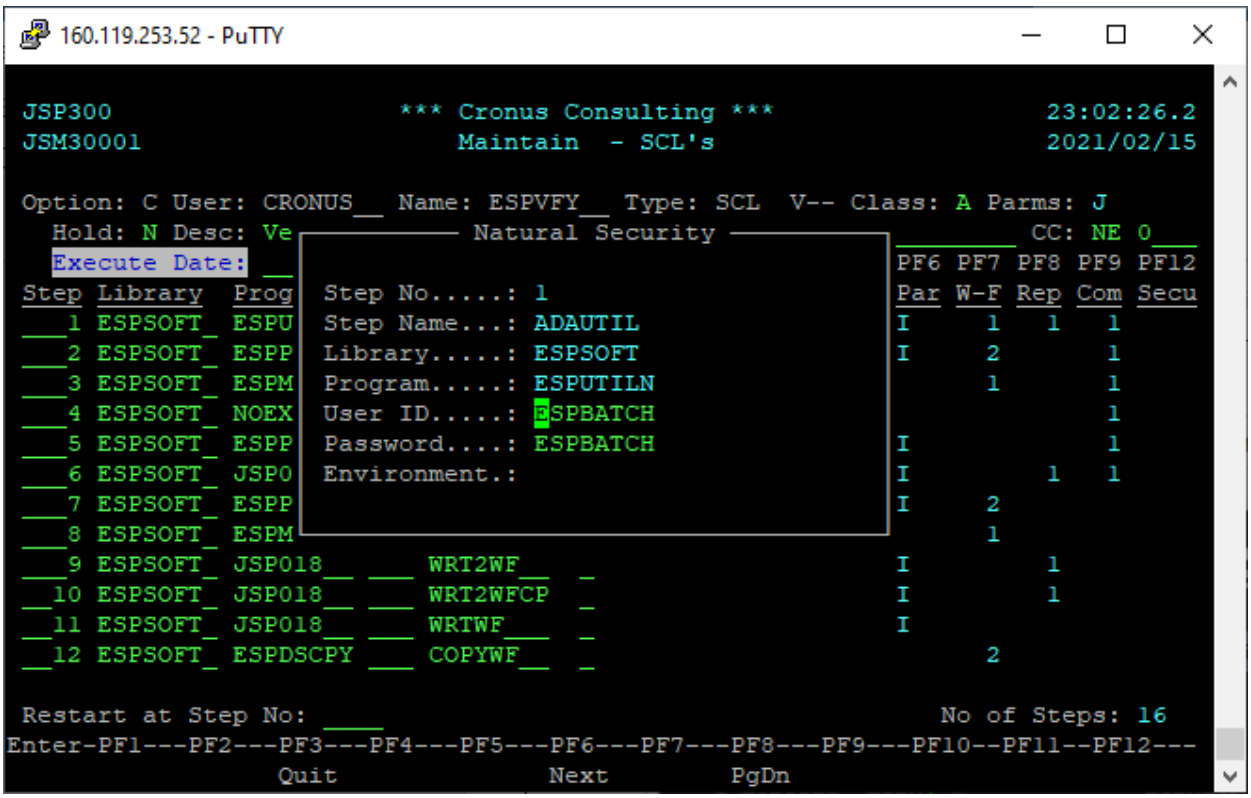


- PF11 - Activates the Job Parameter specification screen. This allows the user to create aliases for previously defined variables. Refer to section: **6 – Dynamic variable substitution** for a list of predefined dynamic variables and an explanation on how to define your own dynamic variables.



Alias “STARTDAT” and “ENDDAT” can now be used within the SCL and will be substituted at submission time will the relevant values. Note that dynamic variables must be prefixed by a ‘&’.

PF12 - Activates the Natural Security specification screen where the User-Id and Password can be specified for submission via Natural Security.



Job steps will only be executed using Natural Security if the Natural Security Indicator is set for the run environment specified. The Natural Security indicator is set using function **JS002 Maintain - EspBatch Codes**. Code Type: **ENV**, Code Value: <**env-id**> - where env-id denotes the run environment specified for the job step. If no environment is specified for the job step, the default EspBatch environment is used.

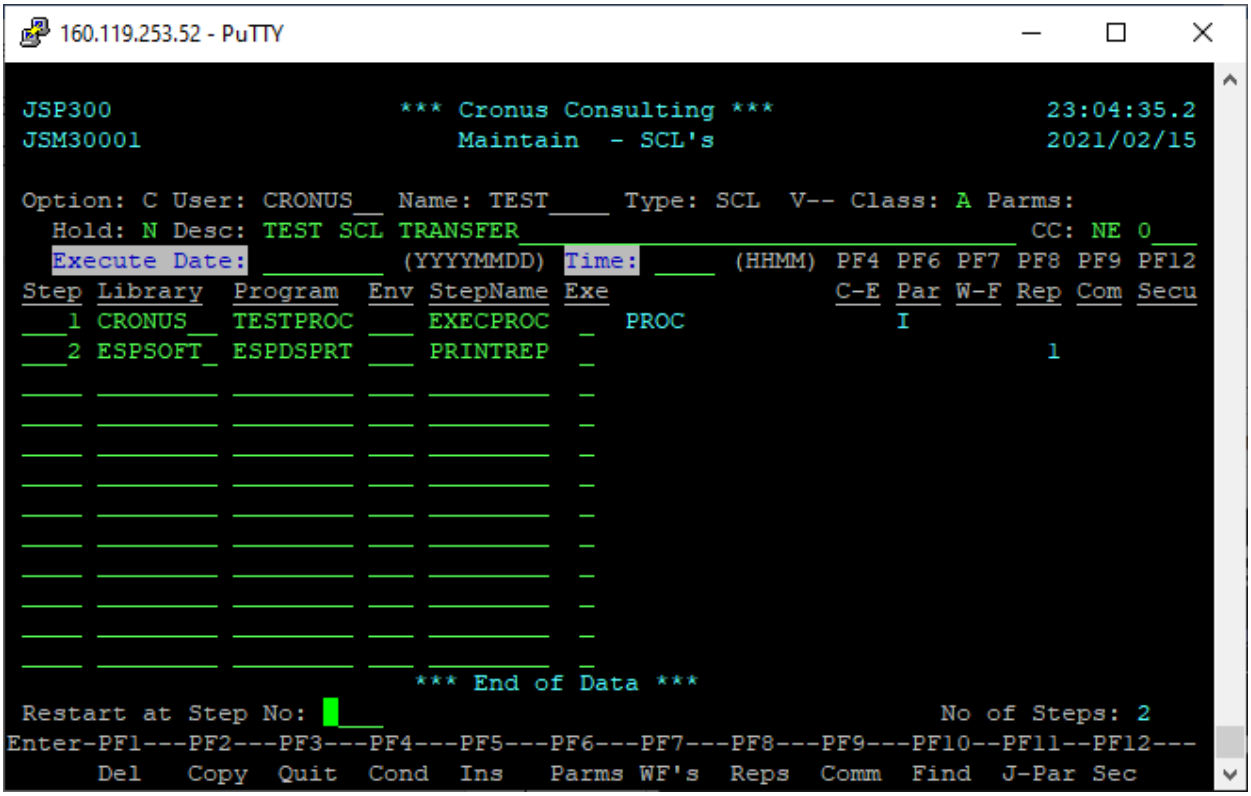
Submission of SCL Procedures:

A SCL procedure is defined in the same way as a normal SCL, the only exception is that the SCL Type must be specified as **PROC**. A SCL procedure may only contain 1 step.

To execute a SCL procedure from within a SCL step

- Library must contain the procedures SCL User Library
- Program must contain the procedures SCL Name

When a SCL step is submitted - EspBatch first tests whether a procedure exists using the **Library** as SCL User Library, the **Program** as SCL User with a SCL Type of '**PROC**'. If no procedure is found the step executes the program/utility within the specified library.



In the example above SCL **CRONUS.TEST.SCL** is executing procedure **CRONUS.TESTPROC.PROC**.

When a procedure is executed from within a SCL, the SCL header information (Class, Hold, etc.) of the procedure is ignored – the header information of the SCL executing the procedure is used.

Any values defined within the SCL step executing the procedure will override values specified within the SCL procedure.

To override the Library and Program/Utility specified within the procedure the following variables may be used. These variables must be defined under the Input parameters of the step executing the procedure.

Note that the # variables defined are only used for parameter substitution and will not be passed as input parameter(s) to the calling program/utility.

#LIBRARY=Library Name
#PROGRAM=Program/Utility Name

If #LIBRARY and/or #PROGRAM is specified as a parameter within the SCL step executing the procedure the Library Name and/or Program/Utility Name specified within the procedure will be overwritten with the value assigned to #LIBRARY and #PROGRAM.

```
                                  Parameters

 SCL User: CRONUS    SCL Name.: TEST      SCL Type: SCL  V-- Library: CRONUS
 Step No.: 1         Step Name: EXECPROC                    Program: TESTPROC

                   Control Card Format: @@CONTROL@@User Lib;Control Card
                   Setup Card Format..: @@PROC@@SubSystem;Work File Name
     Value                              @@PROC@@$LABEL
   1 #LIBRARY=ESPSOFT
   2 #PROGRAM=ESPSLEEP
   3 5
```

Procedure "**CRONUS.TESTPROC.PROC**" called with above parameters.

```
160.119.253.52 - PuTTY

JSP300                    *** Cronus Consulting ***                 23:12:43.0
JSM30001                      Maintain  - SCL's                     2021/02/15

Option: C User: CRONUS__ Name: TESTPROC Type: PROC V-- Class: A Parms:
  Hold: N Desc: TEST PROC                                          CC: __ ____
  Execute Date: ________ (YYYYMMDD) Time: ____ (HHMM)  PF4 PF6 PF7 PF8 PF9 PF12
Step Library  Program  Env StepName Exe              C-E Par W-F Rep Com Secu
___1 EXECLIB_ EXECPROG ___ SLEEP10_  _

                          *** End of Data ***
 Restart at Step No: ___                                    No of Steps: 1
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                 Quit  Cond        Parms WF's  Reps  Comm  Find  J-Par Sec
```

In the example above the values EXECLIB & EXECUTIL will be replaced with the #LIBRARY and #PROGRAM values defined within the Input parameters of the SCL step executing the procedure.

Note that a procedure may not execute another procedure.

3.14 JS302 Maintain - Control Cards

The function is used to maintain control cards. All control cards used within a SCL must be defined using this function. Control cards are expanded at submission time.

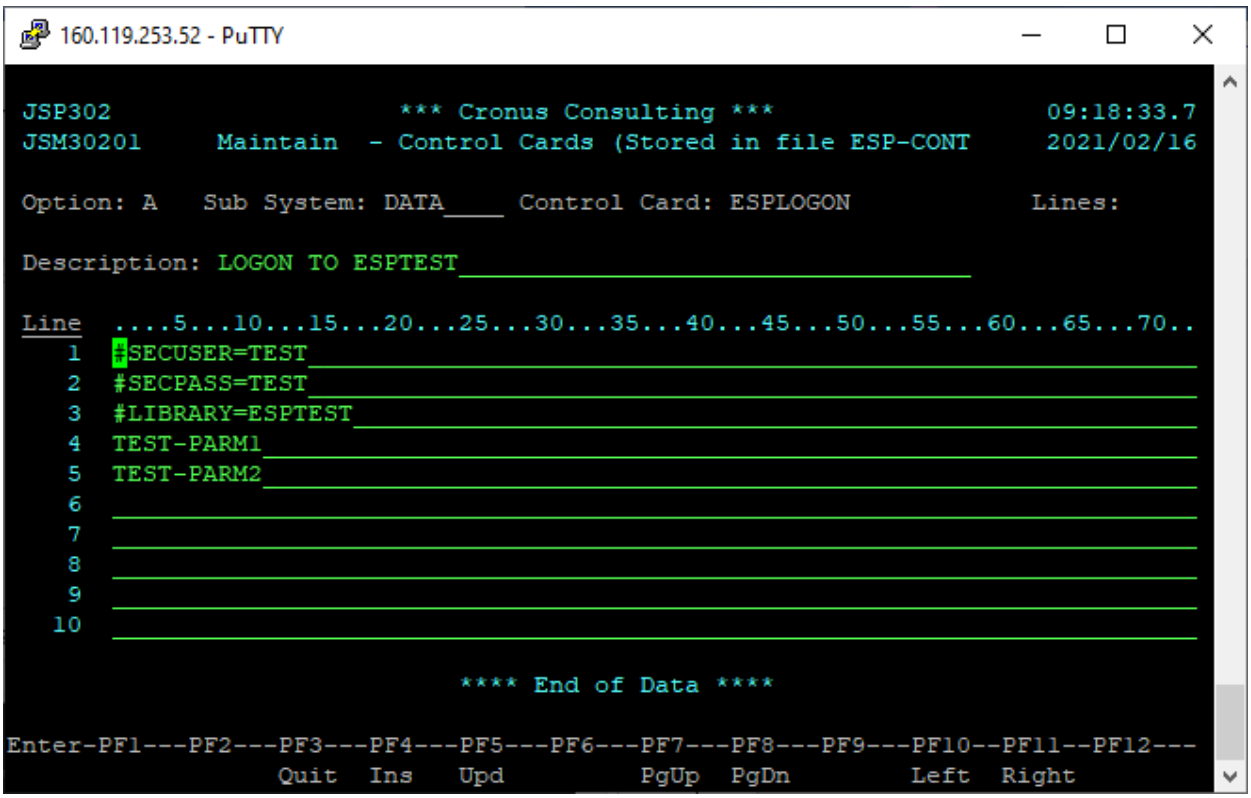
Function Options:

A - Add Control Card

C - Change Control Card

D - Delete Control Card

E - View Control Card



PF4 can be used to insert a blank line.

Control cards are specified within a SCL using the following syntax:
@@CONTROL@@User Lib;Control Card

In the example above “User Lib” is **DATA** and the “Control Card” is **ESPLOGON**.

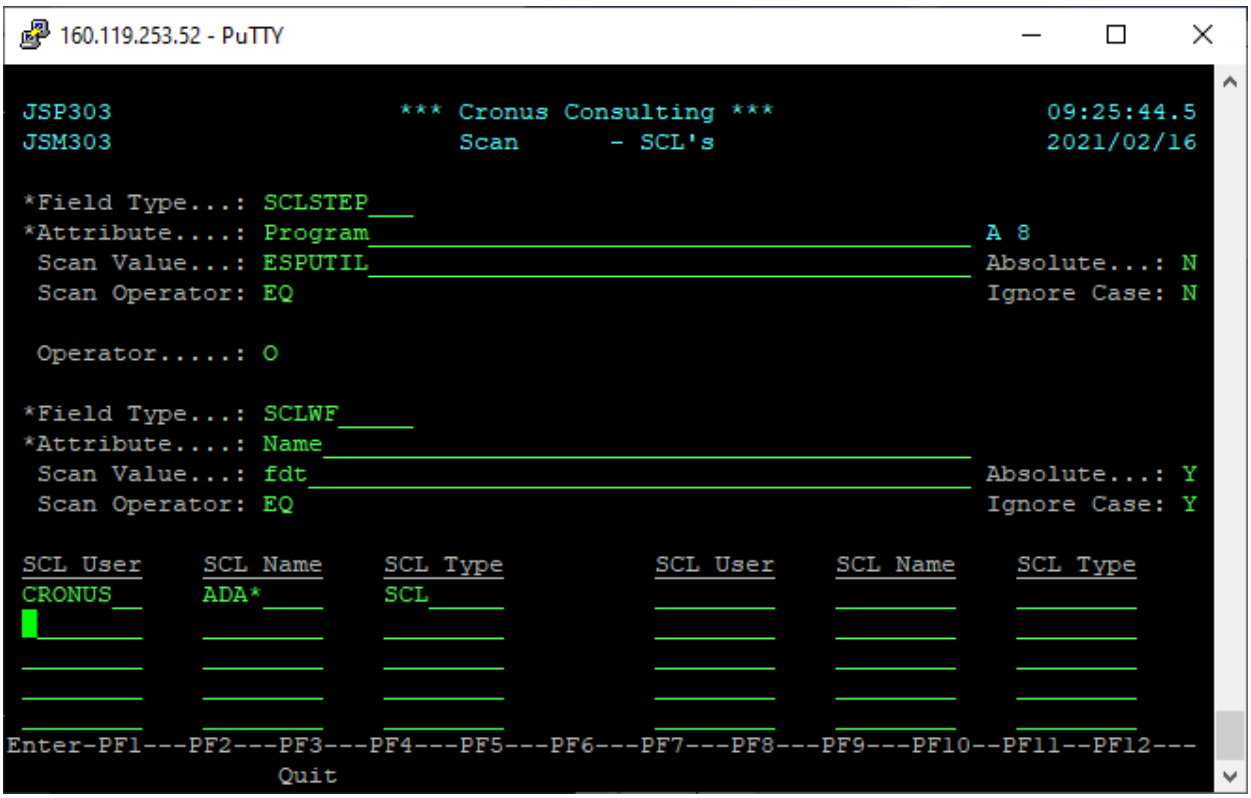
Substitution variables can be defined by using the #Variable Name=Variable Value syntax. These variables are not passed as input parameters to the program/utility but are used to replace certain SCL values.

The following are allowable substitution variables:
#SECUSER=Natural Security User ID
#SECPASS=Natural Security Password
#LIBRARY=Library Name
#PROGRAM=Program/Utility to be execute

The example above contains 3 substitution variables (#SECUSER, #SECPASS & #LIBRARY) and 2 input variables (TEST-PARM1 & TEST-PARM2).

3.15 JS303 Scan - SCL's

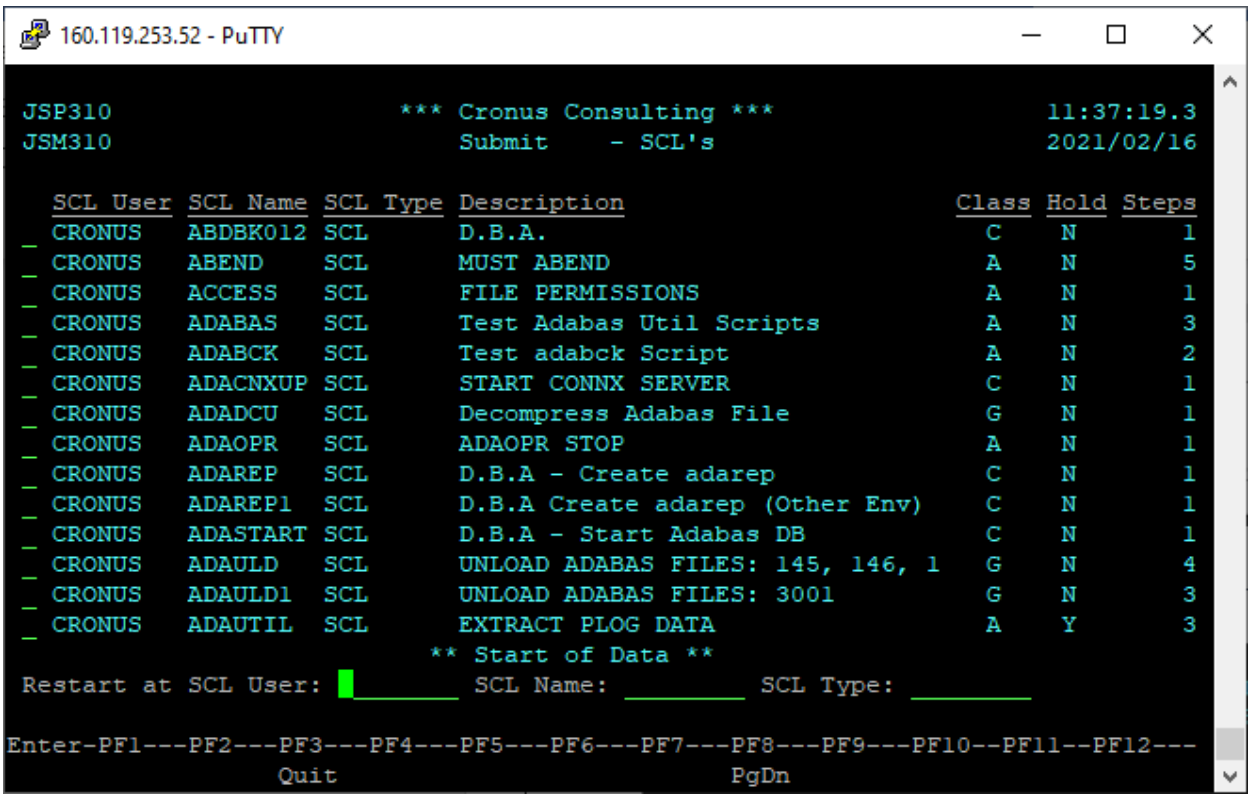
The function is used to scan SCL's for specific values. A report is produced containing all SCL's that contain the scanned values.

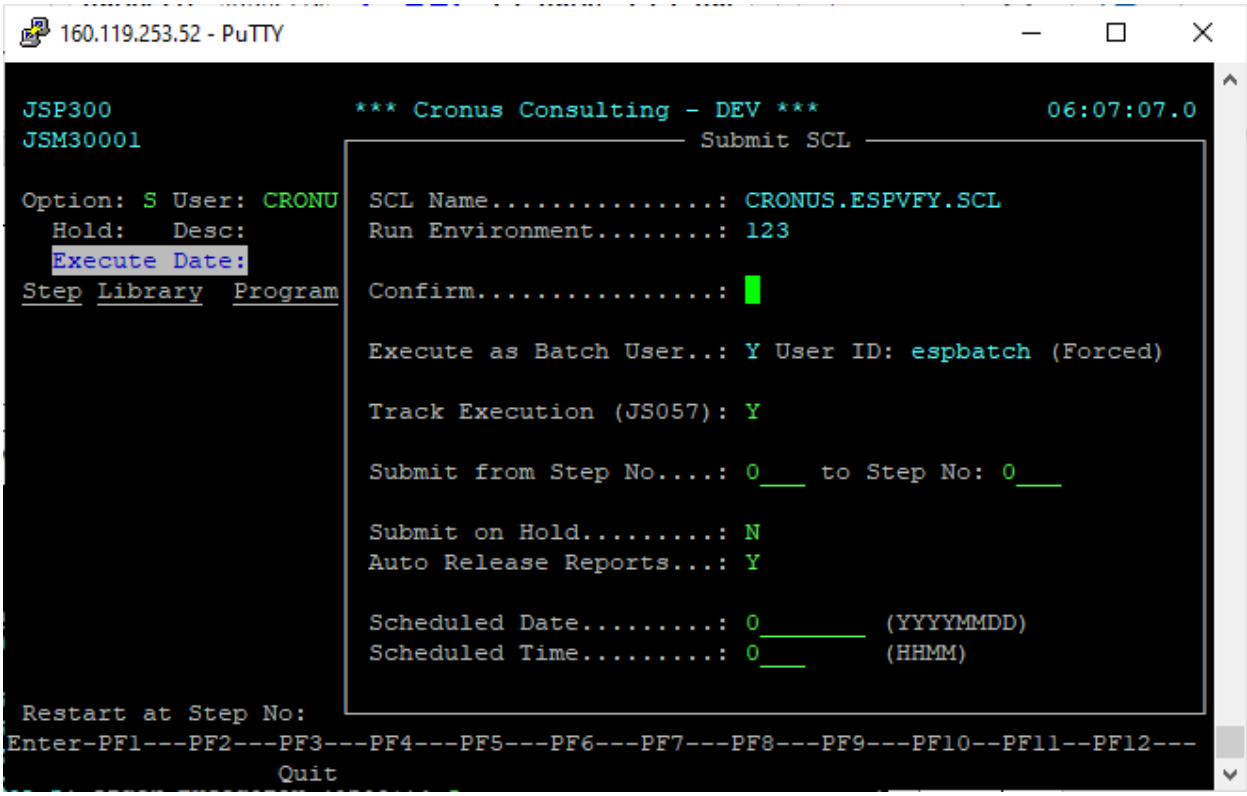


Field Type	-	Each field type has specific attributes that can be scanned.
Attribute	-	The attribute field must contain one of the relevant values for the field type specified.
Scan Value	-	The value that must be scanned for.
Absolute	-	If (**N**)o is specified, the full attribute value must match the scan value entered.
Ignore Case	-	If (**Y**)es is specified the case of the scan value is ignored for searching.
Operator	-	Valid values are: (**A**)nd and (**O**)r.
SCL User	-	The SCL User library that must be scanned, If the SCL User contains an '*' – all SCL User libraries that contain the value specified will be scanned.
SCL Name	-	The SCL Name that must be scanned, If the SCL Name contains an '*' – all SCL Names that contain the value specified will be scanned.
SCL Type	-	The SCL Type that must be scanned, If the SCL Type contains an '*' – all SCL Types that contain the value specified will be scanned.

3.16 JS310 Submit - SCL's

This function is used to submit an SCL by entering an **S** next to the desired SCL. Once the SCL has been selected, a window is displayed providing various submission options.





- Confirm: Valid values are (**Y**)es and (**N**)o. The job will only be submitted if (**Y**)es is entered.
- Execute as Batch User: Valid values are (**Y**)es and (**N**)o. If **Y** is specified the job will be submitted using the User Id defined on function **JS002** with Code Type: **ESPBATCH**, Code Value: **SETUP**. The default batch User Id is specified in the **Batch Submit User** field. If (**N)**o is specified, the job will be submitted using the user id of the person that requested the submission.
- Track Execution (JS057): Valid values are (**Y**)es and (**N**)o. If (**Y**)es is specified function **JS057** is invoked to track the execution of the job.
- Submit from Step No: Submit job starting at the specified step
- To Step No: Submit job ending at the specified step
- Submit on Hold: Valid values are (**Y**)es and (**N**)o. If (**Y**)es is specified, the job will be submitted in Hold.
- Auto Release Reports Valid values are (**Y**)es and (**N**)o. If (**N**)o is specified all reports will be placed on hold. A value of (**Y**)es uses the disposition specified per report.
- Scheduled Date: Job will be scheduled at the specified date. The date must be in YYYYMMDD format and may not be in the past.

- Scheduled Time: Job will be scheduled at the specified time. The time must be in HHMM (Hour and Minute) format and may not be in the past.

3.17 JS315 Maintain - Import/Export/Scan SCL's

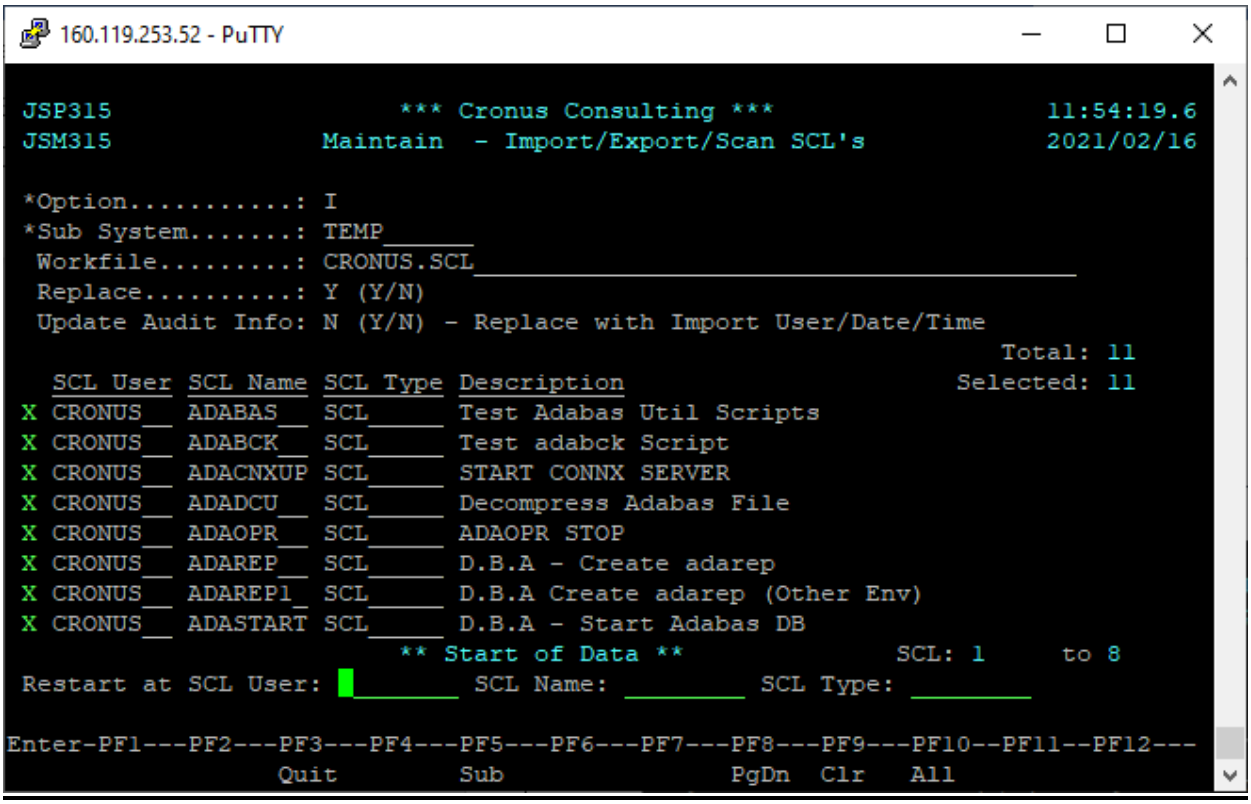
This function is used to export, import and scan SCL's. The SCL's are exported, imported, and scanned to/from the sub system and work file specified.

Function Options:

E - Export SCL

I - Import SCL

S - Scan SCL



With the export option wildcard characters '?' and '*' may be specified in the SCL User, SCL Name or SCL Type fields. The '?' character is used to identify specific positions that can contain any character.

When importing SCL(s) the "Replace" and "Update Audit Info" option becomes available.

Replace - If (**Y**)es is specified existing SCL's will be replaced if they exist on the SCL file being imported. If (**N**)o is specified only SCL(s) that do not exist will be imported.

Update Audit Info - If (**Y**)es is specified the Audit User ID is updated with the User ID of the person that imported the SCL(s) and the Audit Date/Time is set to the

current system date/time. If (N)o is specified, the Audit info contained within the SCL(s) being imported is retained.

User exists:

The following user exits allow customer sites to modify SCL values when importing SCL(s).

ESPUX008: Modify SCL values contained on natural view ESP-SCL
ESPUX009: Modify SCL values contained on natural view ESP-SCL-DET
ESPUX010: Modify SCL values contained on natural view ESP-SCL-DET-TEXT
ESPUX013: Modify SCL values contained on natural view ESP-SCL-REP-SPLIT

If you add code to the user exists, first make sure that you have a clear understanding of the underlying SCL structure as no validation is done of coding in the user exists.

- PF5 - Used to submit the requested option selected and produce a report of all SCL's affected.
- PF7 - Page backward.
- PF8 - Page forward.
- PF9 - Clear all selected SCL's for option (I)mport. Clear all specified SCL's for option (E)xport.
- PF10 - Select all SCL's for option (I)mport.

Take note that the SCL record structure may change between EspBatch versions.

The function must not be used to import SCL's exported in a previous version of EspBatch.

3.18 JS320 Maintain - Security for SCL User Libraries

This function is used to maintain SCL User Library security per User Group and/or User-Id. Users will only have access to SCL User libraries defined per function and be allowed to perform the options specified. A Wildcard character ("*") may be specified in the **SCL User**, **Function** or **Options fields**. By specifying a wildcard character in all fields, the User Group/User-Id, is granted unrestricted access to all SCL User libraries, Functions and Options.

Function Options:

C - Change Security Group

E - View Security Group

In the example below, unrestricted access has been granted to User Group **SYSTEM** by specifying a wildcard character in all fields.

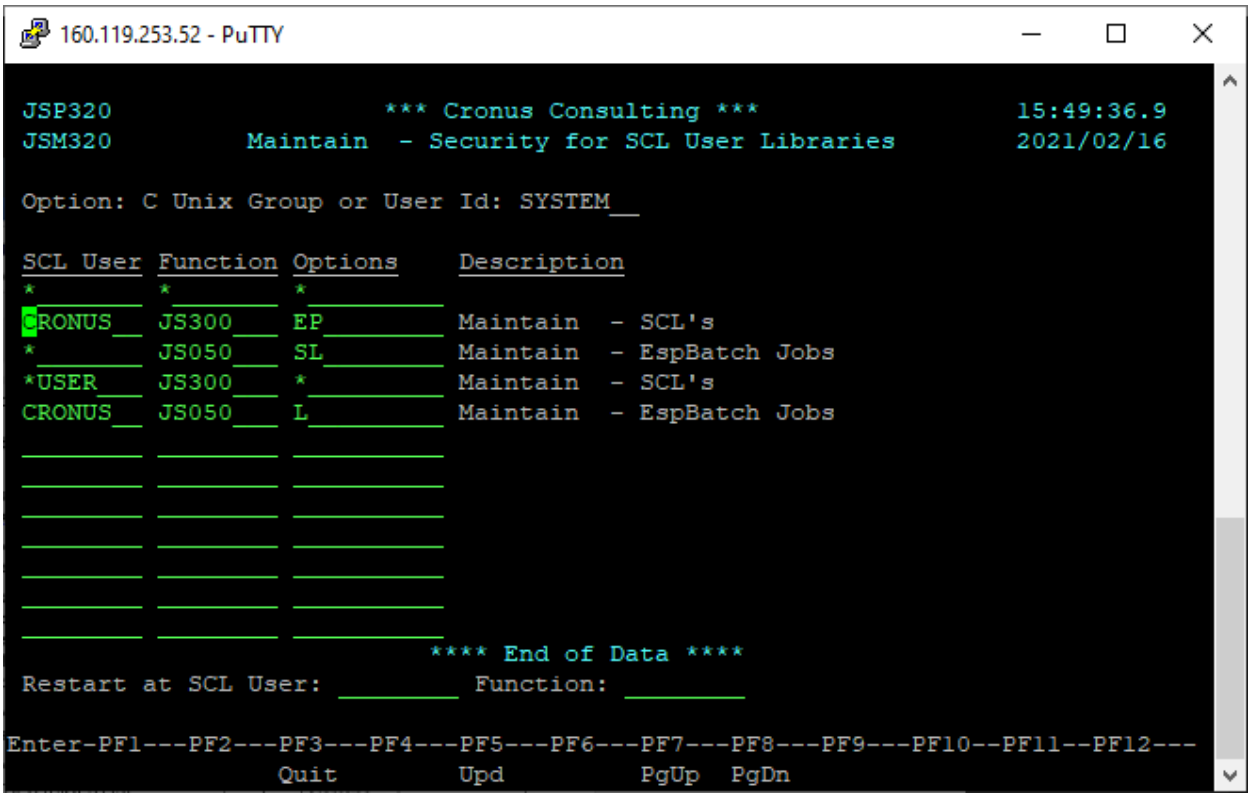
The unrestricted access has been overwritten for the following functions:

JS300 restricted to options "**EP**" for SCL User library "**CRONUS**".

JS050 restricted to options "**SL**" for all SCL User libraries.

JS050 restricted to option "**L**" for SCL User library "**CRONUS**".

JS300 unrestricted access



Note: "SCL User" ***USER** is replaced with the Natural System variable ***USER** when assigning access.

SCL User Library access is granted by combining the access defined on function JS320 for the user and all O/S groups that the user belongs to.

Example: User ID "**ESPBATCH**" executes Function "**JS300**" with SCL User Library "**CRONUS**"

Access is evaluated in the following sequence for the user and all O/S groups that the user belongs to:

SCL User Library	**Function**	
*USER	JS300	Access defined for SCL User Library "**ESPBATCH**" (***USER** replaced by Natural System Variable ***USER**) and Function **JS300**
CRONUS	JS300	Access defined for SCL User Library "**CRONUS**" and Function **JS300**
*USER	*	Access defined for SCL User Library "**ESPBATCH**" (***USER** replaced by Natural System Variable ***USER**) and Function **JS300**
*	JS300	Access defined for **ALL** SCL User Libraries for Function **JS300**
CRONUS	*	Access defined for SCL User Library "**CRONUS**" and **ALL** Functions
*	*	Access defined for **ALL** SCL User Libraries and **ALL** Functions

4. JOB DEFINITION & MONITORING

This section is included as a step-by-step reference on how to define both Online-to-Batch (RJE) and SCL (Mainframe JCL) for the submission of batch jobs. Once a job has been submitted the same functions are used for monitoring a job.

4.1 Online to Batch Submission

The process below describes how a mainframe job submitted via NATRJE must be defined and monitored using EspBatch.

```
************************************************************************
* NAME          : JSP017                                               *
* AUTHOR        : Copyright: Cronus Consulting                         *
* DESCRIPTION   : Online to Batch submission - Example                 *
************************************************************************
DEFINE DATA
LOCAL USING JSLPARM                       /* SUBMIT
LOCAL USING MLCNTRL                       /* GENERAL
LOCAL USING MLGENVAR                      /* GENERAL
LOCAL
1 #PARAMETERS           (A16)
1 REDEFINE #PARAMETERS
  2 #START-DATE         (N8)
  2 #END-DATE           (N8)
*
1 #MESSAGE              (A78)
1 #NO-REQUIRED          (N3) INIT <2> /* Number of steps in SCL
END-DEFINE
*
PERFORM GET-INPUT
************************************************************************
DEFINE SUBROUTINE GET-INPUT /* User Input for Online to Batch Job
************************************************************************
INCLUDE MCSETUP /* SETUP PF-KEYS AND INITIAL VALUES
*
E01. REPEAT
  #START-DATE := *DATN
  #END-DATE   := *DATN
*
```

```
  INPUT WITH TEXT #MESSAGE USING MAP 'JSM017'
*
  INCLUDE MCCHKPF /* PF KEY ACTIONS
*
  IF #START-DATE NE MASK(19-20YYMMDD)
    REINPUT 'Invalid Start Date'
      MARK *#START-DATE
  END-IF
*
  IF #END-DATE NE MASK(19-20YYMMDD)
    REINPUT 'Invalid End Date'
      MARK *#END-DATE
  END-IF
*
  PERFORM SUBMIT-JOB
END-REPEAT
END-SUBROUTINE /*(GET-INPUT)
************************************************************************
DEFINE SUBROUTINE SUBMIT-JOB /* Get Batch No & Run Id for each step
************************************************************************
CALLNAT 'JZN050' #JS-RUN-ID-ARRAY(1:500) #NO-REQUIRED #JS-BATCH-NO
*
PERFORM SUBMIT-STEP-1
PERFORM SUBMIT-STEP-2
END-SUBROUTINE /*(SUBMIT-JOB)
************************************************************************
DEFINE SUBROUTINE SUBMIT-STEP-1
************************************************************************
ADD 1 TO #JS-RUN-ID-POS   /* Increment Step Pointer
*
#JS-APPLICATION      := 'ESPSOFT'    /* Library containing Program
#JS-JOB-NAME         := 'JSR017'     /* Program Name
#JS-JOB-CLASS        := 'A'          /* Job Class to run in
#JS-BATCH-NAME       := 'ESPSTATS'   /* Name of Batch Job
#JS-PROF-PARMS(1)    := 'ETID=JSR017'/* End Transaction ID for Step
#JS-HOLD-FLAG        := 'N'          /* Submit on hold (Y)es or (N)o
*
#JS-PARMS(1)         := #PARAMETERS  /* Parameters passed to program
#JS-PARMS(2)         := 'PARM2'
```

```
#JS-RESERVE          := 'BATCH'     /* BATCH for all but last step
*
#JS-SEC-LIBRARY      := 'ESPSOFT'   /* Library Name - Natural Security
#JS-SEC-USER         := 'DBA1234'   /* User ID      - Natural Security
#JS-SEC-PASSWD       := 'MYPASS'    /* Password     - Natural Security
*
* Report setup: Report 1, Distribution 1
#JS-PRINTER(1,1)     := 'OPTPRT'    /* Printer ID
#JS-REPORT-NAME(1,1) := 'ESPREPORT' /* Report Name in Spool Queue
#JS-COPIES(1,1)      := 2           /* Number of Copies
#JS-REP-CLASS(1,1)   := 'A'         /* Spool Class
#JS-REP-DISP(1,1)    := 'K'         /* Disposition
*
* Report setup: Report 3, Distribution 1 & 2
#JS-PRINTER(3,1)     := 'P117'      /* Printer ID
#JS-COPIES(3,1)      := 1           /* Copies
#JS-REP-CLASS(3,1)   := 'A'         /* Spool Class
#JS-REP-DISP(3,1)    := 'K'         /* Distribution
*
#JS-PRINTER(3,2)     := 'P118'      /* Printer ID
#JS-COPIES(3,2)      := 2           /* Copies
#JS-REP-CLASS(3,2)   := 'A'         /* Spool Class
#JS-REP-DISP(3,2)    := 'K'         /* Distribution
*
* Work File setup: Workfile 1
#JS-WORKFILE(1)      := 'WF1.TXT'   /* Workfile Name
#JS-SUB-SYSTEM(1)    := 'TEMP'      /* Subsystem (Setup on JS002)
#JS-DISPOSITION(1)   := 'M'         /* Disposition
                                    /* N - New
                                    /* M - Mod (Append)
                                    /* O - Old
                                    /* P - Print
* Work File setup: Workfile 4
#JS-WORKFILE(4)      := 'WF4.TXT'   /* Workfile Name
#JS-SUB-SYSTEM(4)    := 'PROD'      /* Subsystem (Setup on JS002)
#JS-DISPOSITION(4)   := 'N'         /* Disposition
*
INCLUDE JSC102 /* Submit Job Step
END-SUBROUTINE /*(SUBMIT-STEP-1)
```

```
************************************************************************
DEFINE SUBROUTINE SUBMIT-STEP-2
************************************************************************
ADD 1 TO #JS-RUN-ID-POS   /* Increment Step Pointer
*
#JS-APPLICATION      := 'ESPSOFT'   /* Library containing Program
#JS-JOB-NAME         := 'JSR018'    /* Program Name
#JS-JOB-CLASS        := 'A'         /* Job Class to run in
#JS-BATCH-NAME       := 'ESPSTATS'  /* Name of Batch Job
#JS-HOLD-FLAG        := 'N'         /* Submit on hold (Y)es or (N)o
*
#JS-RESERVE          := 'LAST'      /* BATCH for all but last step
*
#JS-SEC-LIBRARY      := 'ESPSOFT'   /* Library Name - Natural Security
#JS-SEC-USER         := 'DBA1234'   /* User ID      - Natural Security
#JS-SEC-PASSWD       := 'MYPASS'    /* Password     - Natural Security
*
* Report setup: Report 1, Distribution 1
#JS-PRINTER(1,1)     := 'OPTPRT'    /* Printer ID
#JS-REPORT-NAME(1,1) := 'EMP_DET'   /* Report Name in Spool Queue
#JS-COPIES(1,1)      := 2           /* Number of Copies
#JS-REP-CLASS(1,1)   := 'A'         /* Spool Class
#JS-REP-DISP(1,1)    := 'K'         /* Disposition
*
INCLUDE JSC102 /* Submit Job Step
END-SUBROUTINE /*(SUBMIT-STEP-2)
END
```

4.2 Online to Batch Parameters

Parameter Name	Format	Length	Occurrences	Description
#JS-BATCH-NAME	Alpha	8		Batch Job Name. If the Batch Name defined is defined as a valid UNIX/Linux user-id, then the job will be submitted Batch Name User-Id.
#JS-DESCRIPTION	Alpha	50		Description of batch job.
#JS-JOB-CLASS	Alpha	1		Job Class that job will be submitted in.
#JS-SCHED-DATE	Numeric	8		Schedule Job at specified Date. Format is YYYYMMDD.
#JS-SCHED-TIME	Numeric	4		Schedule Job at specified Time. Format is HHMM (Hour & Minute).
#JS-EXEC-DATE	Numeric	8		Execute Job with specified system Date. Format is YYYYMMDD
#JS-EXEC-TIME	Numeric	4		Execute Job with specified system Time. Format HHMM (Hour & Minute).
#JS-COND-CODE	Numeric	4		Job Header Condition Code value.
#JS-COND-CODE-TEST	Alpha	2		Job Header Condition Code test. E.G. **NE, EQ, LT, LE, GT or GE.**
#JS-HOLD-FLAG	Alpha	1		Submit Job on hold: **Y**es or **N**o
#JS-RESERVE	Alpha	80		Valid values are: '**SCREEN**', '**BATCH**' and '**LAST**'. These values are only evaluated for Online to Batch submission. **SCREEN** – A window is displayed showing Job Step information. User must press <ENTER> to proceed. **BATCH** – A window is displayed showing Job Step information. **LAST** – A window is displayed showing Job information. User must press <ENTER> to proceed. **SUBMIT** – Wait flag is removed for all steps that have already been submitted. Normally wait flag is only removed when last step is submitted.

#JS-ERROR	Logical			Set to TRUE if job step could not be submitted.
Logon Information for Natural Security				
#JS-SEC-LIBRARY	Alpha	8		Natural Security Library ID.
#JS-SEC-USER	Alpha	8		Natural Security User-ID.
#JS-SEC-PASSWD	Alpha	8		Natural Security Password.
Report Specification (Report 1 to 31, Distribution 1 to 5)				
#JS-PRINTER	Alpha	8	1:31,1:5	Name of printer as defined in spooler. If the disposition is defined as **W**orkfile, this field must contain the name of the sub-system where the work file must be created.
#JS-REPORT-NAME	Alpha	50	1:31,1:5	Name that report must be routed as. If the disposition is defined as **W**orkfile, this field must contain the name of the work file that must be created.
#JS-COPIES	Numeric	2	1:31,1:5	Number of copies to print. If the disposition is defined as **W**orkfile and the copies field contains a 1 the next GDG version will be created.
#JS-REP-CLASS	Alpha	1	1:31,1:5	Report Class. Valid report classes are: A thru P and a thru p.
#JS-REP-DISP	Alpha	2	1:31,1:5	Report disposition: **H** – Submit in Hold status and delete after printing. **K** – Print and Retain. **L** – Submit in Hold status and retain after printing. **D** – Print and delete. **W** – Spool to disk.
#JS-FORM-TYPE	Alpha	8	1:31,1:5	Form Definition used to print report.
#JS-PRINTER-PAGE	Alpha	8	1:31,1:5	Page Definition used to print report.
#JS-PRINTER-FCB	Alpha	8	1:31,1:5	Form No used to print report.
#JS-PRINTER-LABEL	Alpha	8	1:31,1:5	Reserved for COBOL.
#JS-PRINTER-PRMODE	Alpha	8	1:31,1:5	PRMODE used to print report.
#JS-PRINTER-LC	Numeric	4	1:31,1:5	LC Parameter used to print report.
#JS-EMAIL-ADDRESS	Alpha	50	1:31,1:5	E-mail Address to route report to.

#JS-USER-ID	Alpha	8	1:31,1:5	User-Id that report must be routed with. This User-Id is case sensitive.
#JS-PRINTER-CHARS	Alpha	8	1:31,1:5	CHARS Parameter used to print report.
#JS-PRINTER-UDEF	Alpha	200	1:31,1:5	Free format field for specification of additional print parameters.
#JS-ARCHIVE-DAYS	Numeric	4	1:31,1:5	The number of days to retain the archived report before it is deleted. If a value of 9999 is specified the report is not archived.
#JS-RETAIN-NO-PRINT	Numeric	4	1:31,1:5	The number of hours the report will be retained in the print queue if it has not printed.
#JS-RETAIN-PRINT	Numeric	4	1:31,1:5	The number of hours the report will be retained in the print queue if it has printed.
Split Specification (Report 1 to 31, Split parameters 1 to 100)				
#JS-SPLIT-REP-NO	Numeric	2	1:31	Report number to split
#JS-SPLIT-ID	Alpha	8	1:31,1:100	Split ID
#JS-SPLIT-NAME	Alpha	50	1:31,1:100	Name that report must be routed as.
#JS-SPLIT-USER-ID	Alpha	8	1:31,1:100	User-Id that report must be routed with. Must be in lowercase.
#JS-SPLIT-PRINTER-ID	Alpha	8	1:31,1:100	Name of printer as defined in spooler.
#JS-SPLIT-PRINTER-CLASS	Alpha	1	1:31,1:100	Report Class. Valid report classes are: A thru P and a thru p.
#JS-SPLIT-ARCHIVE-DAYS	Numeric	4	1:31,1:100	Number of days that report must be retained in archive.
#JS-SPLIT-SCL-USER	Alpha	8	1:31,1:100	Reserved for future use.
#JS-SPLIT-SCL-NAME	Alpha	8	1:31,1:100	Reserved for future use.
#JS-SPLIT-SCL-TYPE	Alpha	8	1:31,1:100	Reserved for future use.
#JS-SPLIT-CAP-TEXT	Alpha	1	1:31,1:100	Capitalise report text before evaluating.
#JS-SPLIT-CHAR-AND	Alpha	1	1:31,1:100	“AND” character used in split text search value.
#JS-SPLIT-CHAR-OR	Alpha	1	1:31,1:100	“OR” character used in split text search value.
#JS-SPLIT-EMAIL-ADDRESS	Alpha	50	1:31,1:100,1:20	Email address that report must be

				sent to. Occurrences 2:20 reserved for future use.
#JS-SPLIT-TYPE	Alpha	3	1:31,1:100,1:50	Valid value are: AND, OR, E, EN, EP, P, S, Sn and Snn: **S** - Start output and continue until end of report or until end criteria is found. **Snn** - Start output nn pages previous to page on which criteria is found and continue until end of report or until end criteria is found. **P** - Start output and continue until a new PAGE is found.
#JS-SPLIT-ROW-START	Numeric	3	1:31,1:100,1:50	Starting row to search for split text.
#JS-SPLIT-ROW-RANGE	Numeric	3	1:31,1:20,1:50	Ending row to search for split text.
#JS-SPLIT-COL-START	Numeric	3	1:31,1:100,1:50	Starting column to search for split text.
#JS-SPLIT-COL-RANGE	Numeric	3	1:31,1:20,1:50	Ending column to search for split text.
#JS-SPLIT-OPER	Alpha	2	1:31,1:20,1:50	Valid values are: EQ, NE, GT, GE, LT and LE.
#JS-SPLIT-TEXT	Alpha	50	1:31,1:20,1:50	Text to search for
Work File Specification (Work File 1 to 96)				
#JS-WORKFILE	Alpha	50	1:96	Work File Name
#JS-WF-LABEL	Alpha	8	1:96	Label used to identify work file
#JS-SUB-SYSTEM	Alpha	10	1:96	Sub system that the work file must be created in. Sub system must previously be defined using function **JS002** with Code Type **SUBSYSTEM**
#JS-GDG	Numeric	4	1:96	May be blank, +1 (Create next GDG version, only if disposition is **N**ew), -n where “n” refers to the previous GDG version required (Only if disposition is **O**ld)
#JS-DISPOSITION	Alpha	3	1:96	Work File disposition: **N**ew, **M**od, **O**ld or **P**rint.

#JS-WF-DISP-COMP	Alpha	1	1:96	Action to be applied to the work file if the step completes successfully. K – Keep: Work file is retained. D – Delete: Work file is deleted.
#JS-WF-DISP-ABORT	Alpha	1	1:96	Action to be applied to the work file if the step aborts. K – Keep: Work file is retained. D – Delete: Work file is deleted.
#JS-WF-TYPE	Alpha	20	1:96	Specifies the work file type. If no work file type is specified then the work file type is set to “ASCII” except if the work file name ends with a “.SAG” extension it is set to work file type “SAG”.
#JS-WF-MODE-OWNER	Alpha	1	1:96	Set the file permissions when a new work file is created. Refer to documentation of operating system command “chmod”. Allowable values are 6 & 7. If no value is specified the default file creation mask (umask) is used.
#JS-WF-MODE-GROUP	Alpha	1	1:96	Set the file permissions when a new work file is created. Refer to documentation of operating system command “chmod”. Allowable values are 0 thru 7. If no value is specified the default file creation mask (umask) is used.
#JS-WF-MODE-OTHER	Alpha	1	1:96	Set the file permissions when a new work file is created. Refer to documentation of operating system command “chmod”. Allowable values are 0 thru 7. If no value is specified the default file creation mask (umask) is used.
#JS-WF-TEXT	Alpha	70	1:96,1:100	Specification of inline work files. #JS-DISPOSITION must be **O**ld when defining inline text.

Step specific Parameters				
#JS-APPLICATION	Alpha	8		Object Library
#JS-JOB-NAME	Alpha	8		Object Name
#JS-DBID	Numeric	3		ID of Database/Environment to execute Job
#JS-STEP-NAME	Alpha	10		Step Name
#JS-PARMS	Alpha	80	1:180	Input parameters passed to object. To pass a blank parameter enter "**SPACE**" which will be translated at submit time to a blank.
#JS-PROF-PARMS	Alpha	30	1:20	Profile parameters: • Record Hold Processing (WH) • ADABAS User Identification (ETID) • Storage for Sort Programs (SORTSZE) • User Database ID (UDB) • Override Default Report Number (MAINPR) • Maximum Number of DBMS Calls (MADIO) • Day Differential (DD) • Time Differential (TD) • Zero Division (ZD) • Update ADABAS (UPDATE-DB) • Input Type (INPUT-TYPE) The profile parameters must be specified using the following syntax: PARMNAME=PARMVALUE e.g. ZD=OFF
#JS-STEP-DESC	Alpha	78	1:100	Step Description
Execution logic – Step level				
#JS-EXEC-TEST	Alpha	3	1:10	Execution test – values are: **IF**, **AND** & **OR**. 1'st occurrence must contain **IF**.
#JS-EXEC-LEFT-BRACKET	Alpha	10	1:10	Used to specify left parenthesis for

				grouping of conditional logic.
#JS-EXEC-STEP-NAME	Alpha	8	1:10	Name of step that must be evaluated in conditional test. If blank the return code of the last step that executed is evaluated.
#JS-EXEC-OPERATOR	Alpha	2	1:10	Valid operators are: **EQ**, **NE**, **LE**, **LT**, **GE**, **GT**
#JS-EXEC-COND-CODE	Alpha	4	1:10	Valid return codes are: **RUN** and **0** thru **9999**. If **RUN** is specified, the condition will evaluate to **true** if the given step executed.
#JS-EXEC-RIGHT-BRACKET	Alpha	10	1:10	Used to specify right parenthesis for grouping of conditional logic.
Completion logic – Step level				
#JS-SCL-COND-CODE-STEP	Numeric	4		No of steps to skip. Only allowed if completion test is **SS**.
#JS-SCL-COND-CODE-TEST	Alpha	2		Completion Test: **AB** – Abort & **SS** – Skip Step
#JS-SCL-COND-CODE	Numeric	4		Job Step Completion Code
Internal Fields				
#JS-BATCH-NO	Numeric	13		Batch Job Number.
#JS-RUN-ID	Packed	13		Run ID of Job Step
#JS-SEQUENCE	Numeric	6		Sequence number of Job Step.
#JS-RUN-ID-POS	Numeric	3		Must be incremented in online-to-batch SCL definition programs, where #JS-RUN-ID-POS will be 1 for the 1’st step, 2 for the 2’nd, etc.
#JS-RUN-ID-ARRAY	Packed	13	1:500	Contains Run-Id’s assigned to each Job Step. System generated values obtained by calling JZN050.
#JS-SUCCESSOR-RUN-ID	Packed	13		Contains the Run-Id of the next Job Step.
#JS-SCL-USER	Alpha	8		Contains SCL User as defined on JS300. Value is set to *INIT-USER for jobs submitted from online-to-batch.
#JS-SCL-NAME	Alpha	8		Contains SCL Name as defined on JS300. Value is set to #JS-BATCH-

				NAME for jobs submitted from online-to-batch.
#JS-SCL-TYPE	Alpha	8		Contains SCL Type as defined on JS300. Contains ”**OTB**” for jobs submitted from online-to-batch.
#JS-SCL-STEP-NO	Numeric	4		Contains SCL step No as defined on JS300. Value is blank for jobs submitted from online-to-batch.
#JS-AUTO-JOB-NO	Packed	13		Auto Scheduler Job No. Only contains a value if the job was submitted via the Auto Scheduler.
#JS-LOGNAME	Alpha	20		Contains User Id used for job submission (Max - 8 characters).
#JS-API-JOB-NO	Aplha	13		Contains Job No of external scheduler. Only populated if EspBatch job is submitted via API call from external scheduler.
#JS-API-JOB-NAME	Alpha	30		Contains Job Name of external scheduler. Only populated if EspBatch jobs are submitted via API call from external scheduler.
#JS-ERROR	Logical	L		Used for error trapping.

4.3 Batch Submission

Batch jobs (SCL) are defined using function **JS300 - Maintain SCL** by entering job parameters on several input screens. Once a job has been defined it can be submitted using functions **JS300** – **Maintain SCL** or **JS310 -** S**ubmit SCL**.

4.4 Monitoring

Once a job has been submitted (Online-to-Batch or Batch) it can be viewed using function **JS12 View - Scheduled and Executing Jobs**. Once all prerequisites have been satisfied the job will start execution.

Prerequisites:

- Job Scheduler status must be “**Running”** – Displayed in top right-hand corner of **JS12**.
- Job must not be in **HOLD** status – Indicated by a ‘**Y**’ under the **H** column on the far-right hand side of the job entry on **JS12**.
- The scheduled Date/Time must not be in the future – Displayed on the far-left hand side of the job entry on **JS12**.

- The job was submitted in a single stream class and another job is currently executing in that job class.
- The job was submitted in a class that is currently closed for execution. The number of jobs awaiting execution in closed job classes is displayed next to **Jobs in Closed Job Classes:** text.

4.5 Aborted Jobs

A Batch job will only abort if a job step completes with a return code that does not satisfy the condition code specified on the job header or the 'AB' operator is specified at step level.

Once a job aborts function **JS010 Maintain - Aborted Job Steps** can be used to either resubmit or force complete the aborted step.

5. REPORT ARCHIVING

EspBatch provides a facility whereby reports are archived for a given period. By archiving reports the user can view or re-spool a report once it has been removed from the spool queue.

The number of days that a report is retained in the archive is defined by the "**Archive Days**" field on the SCL report parameter specification screen. When submitting online to batch jobs the **#JS-ARCHIVE-DAYS** field must contain the number of days. If "**Archive Days**" is not specified, the default value specified on function **JS002 Maintain - EspBatch Codes** with Code Type: **REPORT** Code Value: **DEFAULTS** is used. If a report must not be archived a value of **9999** must be specified.

5.1 Archive Process

All reports are spooled to the cronus spool directory (**$PRINTTMP**) and remain there until the archive routine (**ESPRARCH**) is executed. **ESPRARCH** must be submitted via an SCL that is defined within the scheduler being used.

The archive process updates the archive inventory and transfers all reports to sub system **ARCHIVE** which must be defined via function **JS002 Maintain - EspBatch Codes** with Code Type: **SUBSYSTEM** Code Value: **ARCHIVE.** Reports remain in the archive sub system until the archive clean-up routine (**JSP551**) is executed which removes reports based on the "**Archive Days**" specified. **JSP551** must be submitted via an SCL that is defined within the scheduler being used.

5.2 View/Re-Spool Archived Reports

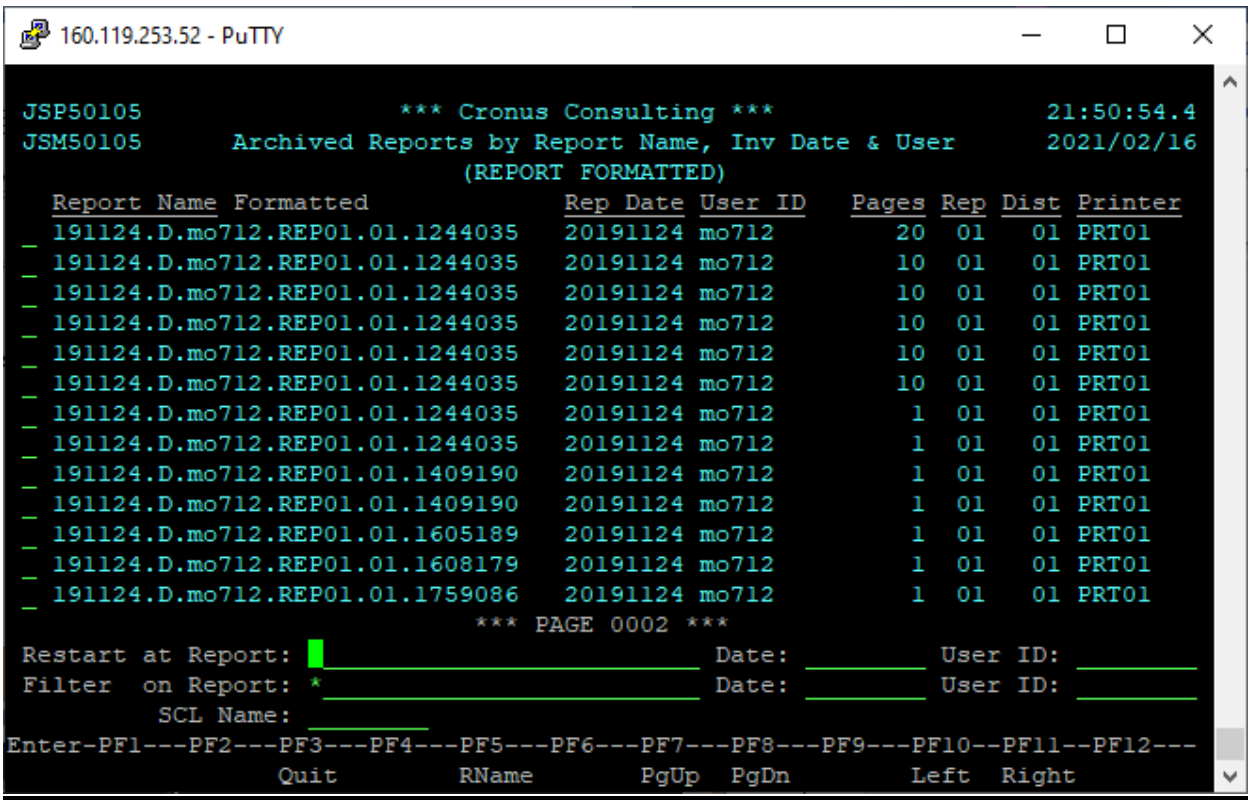
Reports can be viewed and re-spooled using function **JS501 Maintain - Archived Reports**.

Function Options:

P - View Archive Parameters

R - Respool Report

V - View Report



Viewing of reports is dependent on the operating system command “less”. If the command is not available contact your system administrator.

If a report is re-spooled it will not be archived.

6. PRINTING

6.1 Online Printing

The EspBatch routine ESPSETUP in library ESPSOFT must be invoked so that the default report parameters defined on function **JS002 Maintain - EspBatch Codes** with Code Type: **REPORT** and Code Value: **DEFAULTS** can be setup.

If the user has defined printers via function key **PF9** which is executed from the EspMenu then the defined printers will override the default values specified on function JS002.

Should you wish to override the default printer settings from within a natural module then EspBatch routine ESPPRTSN in library ESPSOFT can be invoked.

6.2 Batch Printing

For each batch job that is executed (including online to batch) the parameters defined for the given job will override the default report parameters defined on function **JS002 Maintain - EspBatch Codes** with Code Type: **REPORT** and Code Value: **DEFAULTS.**

Should you wish to override the default printer settings from within a natural module then EspBatch routine ESPPRTSN in library ESPSOFT can be invoked.

6.3 ESPPRTSN

ESPPRTSN provided the ability to setup report parameters from within a natural module. ESPPRTSA contains a list of the parameters that can be passed to ESPPRTSN.

Only 2 of the parameters contained within **ESPPRTSA** are required, the remaining parameters will be populated with the values defined on function **JS002 Maintain - EspBatch Codes** with Code Type: **REPORT** and Code Value: **DEFAULTS**.

When invoking **ESPPRTSN** parameter data area (PDA) **ESPPRTSA** should be used to ensure forward compatibility.

Example: **CALLNAT 'ESPPRTSN' ##ESPPRTSA**

Required Parameters:

- ##REPORT-NUMBER (Alphanumeric 2) - Contains the report no
- ##DIST-NO (Numeric 2) - Contains the distribution copy

The report name that is visible within the print spooler is generated according to the parameters and lengths defined on function **JS002 Maintain - EspBatch Codes** with Code Type: **REPORT** and Code Value: **FORMAT-BATCH** or **FORMAT-ONLINE**.

Should you wish to override generation of the report name, then variable **##REPORT-NAME-FORMATTED** must be set to "**USE-REPORT-NAME**". If this value is set, the report name is set to the value contained in variable **##REPORT-NAME**.

7. USER EXISTS

7.1 ESPUX004

This exit is invoked when a batch step terminates abnormally and can be used for in-house error logging or notification. The exit receives the run id of the step that terminated. Job step info and parameters can be retrieved by accessing the relevant EspBatch files using the run id.

EspBatch Files:

SCH-SCHEDULED-JOBS: Jobs step info including input parameters.
SCH-JOB-WORKFILE: Work file definitions.
SCH-REPORT-DISTRIBUTION: Report parameters.

This user exit or any modules that it calls should not contain any **STOP** or **TERMINATE** statements as control must be passed back to EspBatch after its execution.

7.2 SCLPARMU

This exit allows for the setup of user defined variables that can be defined within an SCL. Refer to section describing dynamic variable substitution,

8. BATCH ROUTINES

8.1 JSP018 - EspBatch Statistics

The routine provides statistics of jobs that have run via EspBatch and can be executed via a script, SCL or online to batch job.

The statistics can be extracted to a file comma delimited file, written to a report or both.

Parameters: Syntax PARMNAME=VALUE

- TYPE=A/E/R (A=Extract & Report, E=Extract or R=Report) Required
- STARTDATE=YYYYMMDD Required
- STARTTIME=HHII Optional
- ENDDATE=YYYYMMDD Required
- ENDTIME=HHII Required
- SUBSYSTEM=Sub System (As per definition on JS002) Required if TYPE is ‘A’ or ‘E’
- FILENAME=File Name Required if TYPE is ‘A’ or ‘E’
- USER=User Name (If blank all User ID's are extracted) Optional
- CLASS=Job Class (If blank all Job Classes are extracted) Optional

If TYPE is set to ‘A’ or ‘E’ a report is produced that only contains the run parameters and totals.

The following fields are extracted to the comma delimited file:

- User ID: User ID that submitted the job
- SCL User Library: SCL User or User ID for online to batch jobs
- SCL Name: SCL/Job name
- SCL Type: SCL Type or “OTB” for online to batch jobs
- Start Date: Start Date in format YYYYMMDD
- Start Time: Start Time in format HH:II:SS:T
- End Date: End Date in format YYYYMMDD
- End Time: End Time in format HH:II:SS:T
- Execution Days: Number of days that step executed
- Execution Time: Execution time in format HH:II:SS:T
- Status: Status of job step
- Job Class: Job class that job step executed in
- Batch No: Batch No of job step
- Run ID: Run ID of job step
- Step Name: Name of job step
- Library: Library that job step executed from
- Program: Program/Utility that job step executed
- Process ID: Process ID assigned to job step

If the routine is invoked from a script the parameters must be exported as environment variables.

8.2 JSP061 – Delete EspBatch history

This routine is used to delete EspBatch history and can be executed via a SCL or online to batch job.

Parameters: Syntax PARMNAME=VALUE

- DAYS=NNN Optional
  - No of days of history to retain
  - Min value: 002
  - Max value: 999
- WORKDAYS=X Optional
  - Where X = (Y)es or (N)o
  - If (Y)es then EspBatch history is retained for the number of workdays (Mon-Fri) specified

If no parameters are specified, the following default values are used:

- DAYS=2
- WORKDAYS=Y

8.3 JSP315B – Batch - Import/Export/Scan SCL's

This routine is used to export, import or scan SCL's to/from a file.

Parameters: Syntax PARMNAME=VALUE

- REPLACE-SCL=(Y)es or (N)o Optional
  - Only applicable for OPTION (I)mport
  - Default is (N)o
- UPDATE-AUDIT=(Y)es or (N)o Optional
  - Only applicable for OPTION (I)mport
  - Default is (N)o

- OPTION=(I)mport, (E)xport, (S)can Required
- SCL=SCL_USER.SCL_NAME,SCL_TYPE Required
  - SCL User, Name & Type separated by dots "."
  - E.G. USERLIB.SCLNAME.SCLTYPE
  - Multiple SCL's can be specified
  - Wildcard character '*' and '?' allowed
    - "?" - Does not check the specified position
    - "*" - Examines for pattern E.G. ESP* (Returns all SCL's that contains 'ESP')

Input-Outfile file must be specified as 1'st work file in SCL

SCL versions are not exported.

9. CONDITION CODES AND CONDITIONAL LOGIC

Condition codes can be specified on the SCL header and/or on individual job steps.

9.1 Condition Codes

If a condition code is specified on the SCL header and any job step completes with a return code that does not satisfy the specified condition code, the job will abort.

SCL header Condition Codes	**RC (Return Code of previous steps)**	
Condition Code	**Execute Current Step**	**Bypass Current Step**
COND=(code,GT)	RC GE code	RC LT code
COND=(code,GE)	RC GT code	RC LE code
COND=(code,EQ)	RC NE code	RC EQ code
COND=(code,LT)	RC LE code	RC GT code
COND=(code,LE)	RC LT code	RC GE code
COND=(code,NE)	RC EQ code	RC NE code

Example:

Condition code of **GT 4** is specified:

- Job will continue execution if the return code of the last step is **4 or greater**.
- Job will abend if the return code of the last step is **0 to 3**.

Condition code of **NE 0** is specified:

- Job will continue execution if the return code of the last step is **0**.
- Job will abend if the return code of the last step is **not 0**.

9.2 Conditional Logic – Step Level

Conditional logic is specified at step level and consists of the following:

Execution Logic:

If specified and the execution logic evaluates to true, the step is executed.

The execution logic is evaluated against the highest return code of all previous job steps that have executed unless reference is made to a specific job step.

Completion Logic:

Completion logic is evaluated after a step has executed and can be used to:

- Abort the execution of a batch job
- Bypass/skip subsequent job steps

10. DYNAMIC VARIABLE SUBSTITUTION

Dynamic variables are coded within an SCL and must be preceded by an '&' which indicates to EspBatch that the variable must be expanded at submission time. There are 2 types of variables:

- Predefined EspBatch/EspAuto variables: These variables are available as a standard feature of EspBatch/EspAuto. Note that the EspAuto variables are only available when the EspBatch job is running under control of EspAuto. EspAuto variables defined within a SCL with produce a SCL submission error if the job is not running under control of EspAuto. Refer to the EspAuto manual for a complete list of available variables.
- User defined variables: These variables can be defined using 1 of 2 methods.
    - Variables defined on JS002 – Code Maintenance: These variables are defined using **Code Type:** SCLVAR and **Code Value:** Variable Name.
    - Variables defined within the provided user exit **SCLPARMU** – The provided user exit is a natural sub-program, thus allowing for creation of user defined variables writing natural code. When upgrading EspBatch **SCLPARMU** must be replaced by the site specified version if used.

When a SCL is submitted all dynamic variables are expanded to reflect the variable value. If the expanded value exceeds the allowable field size an error message is generated and the SCL is not submitted. Secondly the SCL is checked for undefined dynamic variables (all SCL fields are checked for the '&' character), if any exist an error message is generated and the SCL is not generated. There is one exception to the rule – once all dynamic variables have been expanded the input parameters may still contain the '&' character.

The table below contains a list of the predefined dynamic variables available within EspBatch.

Variable Name	Format Returned	Description
YYMMDD-FIRST	YYMMDD	1'st day of current month
YYMMDD-LAST	YYMMDD	Last day of current month
YYMMDD-7	YYMMDD	Today minus 7 days
YYMMDD-6	YYMMDD	Today minus 6 days
YYMMDD-5	YYMMDD	Today minus 5 days
YYMMDD-4	YYMMDD	Today minus 4 days
YYMMDD-3	YYMMDD	Today minus 3 days
YYMMDD-2	YYMMDD	Today minus 2 days
YYMMDD-1	YYMMDD	Today minus 1 days
YYMMDD-FWDOM	YYMMDD	1'st work day of current month
YYMMDD-LWDOM	YYMMDD	Last work day of current month

YYMMDD	YYMMDD	Today
YYJJJ-FIRST	YYJJJ	1'st day of current month
YYJJJ-LAST	YYJJJ	Last day of current month
YYJJJ-7	YYJJJ	Today minus 7 days
YYJJJ-6	YYJJJ	Today minus 6 days
YYJJJ-5	YYJJJ	Today minus 5 days
YYJJJ-4	YYJJJ	Today minus 4 days
YYJJJ-3	YYJJJ	Today minus 3 days
YYJJJ-2	YYJJJ	Today minus 2 days
YYJJJ-1	YYJJJ	Today minus 1 days
YYJJJ-FWDOM	YYJJJ	1'st work day of current month
YYJJJ-LWDOM	YYJJJ	Last work day of current month
YYJJJ	YYJJJ	Today
MMDDYY-FIRST	MMDDYY	1'st day of current month
MMDDYY-LAST	MMDDYY	Last day of current month
MMDDYY-7	MMDDYY	Today minus 7 days
MMDDYY-6	MMDDYY	Today minus 6 days
MMDDYY-5	MMDDYY	Today minus 5 days
MMDDYY-4	MMDDYY	Today minus 4 days
MMDDYY-3	MMDDYY	Today minus 3 days
MMDDYY-2	MMDDYY	Today minus 2 days
MMDDYY-1	MMDDYY	Today minus 1 days
MMDDYY-FWDOM	MMDDYY	1'st work day of current month
MMDDYY-LWDOM	MMDDYY	Last work day of current month
MMDDYY	MMDDYY	Today
DDMMYY-FIRST	DDMMYY	1'st day of current month
DDMMYY-LAST	DDMMYY	Last day of current month
DDMMYY-7	DDMMYY	Today minus 7 days
DDMMYY-6	DDMMYY	Today minus 6 days
DDMMYY-5	DDMMYY	Today minus 5 days
DDMMYY-4	DDMMYY	Today minus 4 days
DDMMYY-3	DDMMYY	Today minus 3 days
DDMMYY-2	DDMMYY	Today minus 2 days
DDMMYY-1	DDMMYY	Today minus 1 days
DDMMYY-FWDOM	DDMMYY	1'st work day of current month
DDMMYY-LWDOM	DDMMYY	Last work day of current month
DDMMYY	DDMMYY	Today
YYYYMMDD-FIRST	YYYYMMDD	1'st day of current month

YYYYMMDD-LAST	YYYYMMDD	Last day of current month
YYYYMMDD-7	YYYYMMDD	Today minus 7 days
YYYYMMDD-6	YYYYMMDD	Today minus 6 days
YYYYMMDD-5	YYYYMMDD	Today minus 5 days
YYYYMMDD-4	YYYYMMDD	Today minus 4 days
YYYYMMDD-3	YYYYMMDD	Today minus 3 days
YYYYMMDD-2	YYYYMMDD	Today minus 2 days
YYYYMMDD-1	YYYYMMDD	Today minus 1 days
YYYYMMDD-FWDOM	YYYYMMDD	1'st work day of current month
YYYYMMDD-LWDOM	YYYYMMDD	Last work day of current month
YYYYMMDD	YYYYMMDD	Today
YYYYJJJ-FIRST	YYYYJJJ	1'st day of current month
YYYYJJJ-LAST	YYYYJJJ	Last day of current month
YYYYJJJ-7	YYYYJJJ	Today minus 7 days
YYYYJJJ-6	YYYYJJJ	Today minus 6 days
YYYYJJJ-5	YYYYJJJ	Today minus 5 days
YYYYJJJ-4	YYYYJJJ	Today minus 4 days
YYYYJJJ-3	YYYYJJJ	Today minus 3 days
YYYYJJJ-2	YYYYJJJ	Today minus 2 days
YYYYJJJ-1	YYYYJJJ	Today minus 1 days
YYYYJJJ-FWDOM	YYYYJJJ	1'st work day of current month
YYYYJJJ-LWDOM	YYYYJJJ	Last work day of current month
YYYYJJJ	YYYYJJJ	Today
MMDDYYYY-FIRST	MMDDYYYY	1'st day of current month
MMDDYYYY-LAST	MMDDYYYY	Last day of current month
MMDDYYYY-7	MMDDYYYY	Today minus 7 days
MMDDYYYY-6	MMDDYYYY	Today minus 6 days
MMDDYYYY-5	MMDDYYYY	Today minus 5 days
MMDDYYYY-4	MMDDYYYY	Today minus 4 days
MMDDYYYY-3	MMDDYYYY	Today minus 3 days
MMDDYYYY-2	MMDDYYYY	Today minus 2 days
MMDDYYYY-1	MMDDYYYY	Today minus 1 days
MMDDYYYY-FWDOM	MMDDYYYY	1'st work day of current month
MMDDYYYY-LWDOM	MMDDYYYY	Last work day of current month
MMDDYYYY	MMDDYYYY	Today
DDMMYYYY-FIRST	DDMMYYYY	1'st day of current month
DDMMYYYY-LAST	DDMMYYYY	Last day of current month
DDMMYYYY-7	DDMMYYYY	Today minus 7 days

DDMMYYYY-6	DDMMYYYY	Today minus 6 days
DDMMYYYY-5	DDMMYYYY	Today minus 5 days
DDMMYYYY-4	DDMMYYYY	Today minus 4 days
DDMMYYYY-3	DDMMYYYY	Today minus 3 days
DDMMYYYY-2	DDMMYYYY	Today minus 2 days
DDMMYYYY-1	DDMMYYYY	Today minus 1 days
DDMMYYYY-FWDOM	DDMMYYYY	1'st work day of current month
DDMMYYYY-LWDOM	DDMMYYYY	Last work day of current month
DDMMYYYY	DDMMYYYY	Today
YYMM-PREV	YYMM	Current Month – 1
YYMM-NEXT	YYMM	Current Month + 1
YYMM	YYMM	Current Month
YYYYMM-PREV	YYYYMM	Current Month – 1
YYYYMM-NEXT	YYYYMM	Current Month + 1
YYYYMM	YYYYMM	Current Month
YY-PREV	YY	Current Year – 1
YY-NEXT	YY	Current Year + 1
YY	YY	Current Year
YYYY-PREV	YYYY	Current Year – 1
YYYY-NEXT	YYYY	Current Year + 1
YYYY	YYYY	Current Year
MM	MM	Current Month
DD	DD	Current Day
DOW-9	Alpha 9	Day of Week
DOW-3	Alpha 3	Day of Week
DOW-2	Alpha 2	Day of Week
HHIISST	HHMMSST	Current Time
HHIISS	HHMMSS	Current Time
HHII	HHMM	Current Time

Below is an extract of a user defined function coded in **SCLPARMU** that sets variable **USERDEF** according to the user that submitted the batch job.

```
000270 PERFORM SET-USERDEF
000280*************************************************************************
000290 DEFINE SUBROUTINE SET-USERDEF
000300*************************************************************************
```

```
000310 ADD 1 TO #START
000320 *
000330 #SCL-PARM-NAME(#START) := '&USERDEF'
000340 *
000350 IF *INIT-USER = 'SAG'
000360   #SCL-PARM-VALUE(#START) := 'SAG'
000370 ELSE
000380   #SCL-PARM-VALUE(#START) := 'USER'
000390 END-IF
000400 END-SUBROUTINE /*(SET-USERDEF)
```