Maintaining *I/A Series save all's* with the `save_all.sh` script

Implementation guide for "save_all.sh" version 2.2.1

invensys

FOXBORO®

Revision 1.1 / June 28, 2003

FOXBORO

FOXBORO

Maintaining *I/A Series save all's* with the
`save_all.sh` script
Implementation guide for "save_all.sh" version 2.2.1

M. de Waal (mwaal@foxboro.com) / R.Deen (rdeen@foxboro.com)
Invensys Process Systems
Field Service Installation Group

Revision 1.1 / June 28, 2003

FOXBORO

Revision 1.1 / June 28, 2003

FOXBORO

Table of Contents:

1. Introduction.

This is the implementation guide for the **`save_all.sh`** script. This script was created to allow interactive and unattended *save_alls* for Foxboro Control databases as found on the I/A Series systems. **Please note that this is NOT an Official Foxboro product. For support you should contact the sales representative or, since you probably did not get it from sales, I guess that you could contact the authors.**

The `save_all.sh` script is not to be mistaken for the Foxboro standard supplied utility named `save_all` (`save_all.ksh` for 70 Series NT) which is located in the `/usr/fox/ciocfg/api` directory on AP and AW station types. This program is however intensively used by the `save_all.sh` script which this manual is about.

Why on earth did we choose the name "`save_all.sh`" for the script and not something completely different? I guess it was poor marketing on our behalf. It is however the name it is known by today, so we have decided to keep it that way.

1.1 Disclaimer.

Due to the inherently complex nature of computer software, Invensys does not warrant that the software described in this document or this documentation is completely error free, will operate without interruption, is compatible with all equipment and software configurations, or will otherwise meet your needs. Accordingly, this documentation is provided as-is, and you assume all risks associated with its use. Invensys makes no warranties expressed or implied, with respect to this document. In no event will Invensys be liable for indirect, incidental or consequential damages, including, without limitation, loss of income, use, or information.

The `save_all.sh` script documented in this manual is provided under the GNU license.
This is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This software is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of merchantability or fitness for a particular purpose. See the GNU General Public License (http://www.gnu.org/licenses/licenses.html#GPL) more details.

To obtain the GNU General Public License, write to:

Free Software Foundation, Inc
59 Temple Place - Suite 330
Boston, MA 02111-1307
USA

1.2 Conventions.

The conventions used in this document are listed below.

When you see this:	It means this:
[Filename]	This typeface indicates a filename of which the contents are printed starting on the next line. This line is NOT part of the file contents.
`STATION#` **`some command<cr>`**	Text printed like this is ASCII text as it appears in a file or on screen. User data that has to be entered is printed in **bold**. Also used to display a pathname or filename in normal text. When used to indicate a command, type the bold printed command until <CR> which means to hit the ENTER key.
Use this data	This typeface is used to show a list of data to be entered in a location indicated in the text. Also used to describe a procedure.
`<Alt_F4>`	Text displayed like this means: press the keys mentioned between brackets <u>simultaneously</u>. In this case press the "ALT" key on your keyboard <u>together</u> with the function-key "F4" Can also be a variable that replaces the word between < >
Start/Run…	Indicates a menu sequence. Here it reads Press the **Start** button and on the next menu select **Run**…

1.3 Revision history (this manual).

Revision number:	Description:
Revision 1.0 December 18, 2002	• Initial release.
Revision 1.1 June 16, 2003	• Many fixes and typos. • Changed the title. • Reorganised much text to make sure you can't find anything anymore. • Added "Save_all script error and status messages" section. • Updated the FSF GNU License link. Old one had expired. • It is as good as new again...

1.4 Revision history (the save_all.sh script).

Version number:	Description:
Version beta February 23, 2000	• Initial release.
Version beta February 24, 2000	• Added usage information. • Added backup procedure for older save_alls. • A lot of testing, seems OK. • Added log-file: saveall.log. • Major clean-up of original script.
Version beta February 25, 2000	• A lot of problems solved, it seems to work now! • Compressing the backup tar files to save space. • All tests pas%&&*!@~~!@#$!____!!@$$#%%$^&^^& core dumped.
Version 1.0 February 28, 2000	• Fixed: Scripts did not work in crontab, added ". /" • Added check for control station and remote hosts.
Version 1.1 February 29,2000	• Added 20 Series host support. • Cleaned up the script. Tested remote stations handling. • Improved reporting to screen and log file.
Version 1.2 March 01,2000	• Added Windows NT Support, Added archive management. • Added free space check.
Version 1.3 March 02,2000	• Enhanced NT Support. Improved reporting. • Bug fix: script would fail with no SAVEALL directory (UNIX only).
Version 1.3.2 March 10,2000	• Bug fixed: Deletion of old backups was done BEFORE making new backup, causing one backup more then specified. • Timestamp for backups now *saveall*-time instead of backup-time, including hour:minutes, so more then 1 backup per day is possible
Version 2.0 August 22,2000	• Added -u option to include upload before save_all (SBD) • Added the -h HOST option to make save_alls for HOST (RD) • Fixed: restore the "before last" save_all when last save_all failed (RD) • Added recursive search for include files in sequences (MDW), Include files are stored in "$SA_DIR/include"
November 09, 2000	• Time of backup fixed (only occurred on NT-platform
February,March 2001	• Remote host includes added • Includes saved with "relative" paths • Cleaned up script and tested on NT platform • Added external config file support (filename: `save_all.cfg`)
Version 2.1 Dec 12, 2001	• Bug fix. Timeout was set to 0 after first CP with surplus backups, resulting in no prompt/timeout for all others. • Bug fix. No validation done on user CPFILE file. • Enhancement: show if external "save_all.cfg" used at startup. • Enlarged STATKB from 2000 to 5000 kb.(estimated reqd diskspace per station) • Bug fix. Remote locked stations would not be identified as such. • Bug fix. Diskspace check always counts all controlstations even if only a few where selected. • Renamed all tmp files, added "sa_" prefix • Removed "schrink" command from "upload" function, reason was possible workfile corruption. • Added include search for SFC files
Verson 2.2 Dec 2002	• Version new variable "PF" for temp files (prefix). • Added save_all of volumes in this version. (by request) • Modified create save_all dir using **mkdir -p** command. • Entering invalid **save_all.sh -b** option, no longer stops save_all,instead continues with default value in script or config file. • Added few cleanup statements when script would exit. • Logfiles are now saved with *save_all* date and old logs are in "log" directory. • Some minor fixes to logfile.

Version number:	Description:
Version 2.2.1 June 12, 2003	• Bug fix. Sequence include search could not handle **#include <filename>** syntax. Usually the syntax is **#include "filename"** • Modified: Include files were always searched on "remote station", even if the host was the local station. • Bug fix. Temp files where not cleared after exit. Could result in multiple *save_alls*. • Added feature. Script is now IACC aware. Will not start if system is prepared for the new IACC platform. • Bug fix. Changed the crontab example because the save_all.cfg file would not be found in the script home directory. • Added simple interrupt handling. Cleanup files after this happens. • Added "local" commandline switch to allow locally hosted stations and volumes only.

2. Making "save alls"

Making a *save_all* is protecting your engineering efforts. Traditionally the *save_all* procedure saves your Control database which usually resides in a Control Processor, Gateway, Integrator or Micro I/A (referred to as Control Stations after this) to a floppy disk from where it can be stored in a safe place. Since I/A Series version 4.3, the Foxboro system provides an **ICC API** which allows creating scripts, that can optionally run unattended, to interact between the Control Station and a user application. I.e.: creating blocks and compounds etc. can now be done while creating these from a database to generate all the control blocks on the fly but also other more administrative tasks come to mind.

These *ICC Driver Task scripts* can also be used to get data out of the Control Stations. An example Saving the Control loops from the Control Station to a floppy is what is known as a *save_all*. A *save_all* creates a readable (For the ICC anyway) format database on this floppy.

However, floppies are very slow and sometimes (un)reliable, the database will not always fit on one floppy and requires user intervention. It is also to say the least, a very time consuming task and it would be a great thing if this could be automated.

The Foxboro I/A 50 and 70 Series stations provide a tool that is located in the `/opt/fox/ciocfg/api` directory named "save_all". This tool, which makes use of the **ICC API** mentioned earlier, will allow making a *save_all* to a location on the local diskdrive of an I/A Series AP or AW.

This basic tool with the name `save_all` (`save_all.ksh` for I/A 70 Series) takes at least two parameters:

```
1AWB11# save_all<cr>
Usage: save_all [-d] <station> <path>
       -d        debug mode: do not remove command input file to
                 driver task
       <station> is the Control Station to be saved
       <path>    is the directory (or device) to which the data
                 is to be saved
1AWB11#
```

The `save_all.sh` script was built around this basic tool and was intended to make life a lot easier when making and maintaining *save_alls*!

This implementation guide was written based on **version 2.2.x** of the `save_all.sh` script. The `save_all.sh` script is platform independent so it runs all I/A 50 and I/A 70 hosts without modifications. The script depends very much on the standard utilities provided on the standard Foxboro I/A 50 and 70 Series. All examples in this document are assuming you use the default settings for the different file location etc. and are not using the external configuration file unless stated otherwise.

2.1 A simple example

With the tools provided in the Foxboro standard software a very simple *save_all* strategy can be achieved. The Foxboro supplied `save_all` utility can be found in `/opt/fox/ciocfg/api` directory on any AP or AW running I/A Series software (On Drive `D:` for I/A 70 Series hosts).

When the `/opt/fox/ciocfg/api/save_all` script is used, an automatic *save_all* procedure can be very simple if we follow these simple steps:
- Make sure a directory for every control station in the system exists.
- Note that all configured Control Stations are listed in a system file: `/etc/cplns`.
- For all stations found in `/etc/cplns` run the `save_all` program with the parameters
  `save_all <station> <path>`
  If the path exists you will get a proper `save_all` for all the stations in `/etc/cplns`.

So a very simple `save_all.sh` script could be something like this:

```
for STATION in `cat /etc/cplns`
do
mkdir /opt/SAVEALLS/$STATION >/dev/null 2>&1
cd /opt/fox/ciocfg/api
save_all $STATION /opt/SAVEALLS/$STATION
done
```

This script will do the trick, however there are some drawbacks to this approach:
- Every time the script is run, you overwrite the previous *save_all*.
- When a compound is removed, it remains in the *save_all* directory if it was not cleared prior to the new *save_all*.
- There is no backup mechanism.
- If you delete the directory contents before making the *save_all* and if the *save_all* fails for any reason, you end up with nothing!
- There is no check of used diskspace.
- There is no log of the *save_all* progress.

The `save_all.sh` script was in fact written based on the *barebone* example as shown above.
In order to have a tool that is a little more useful these problems (and many others) should be addressed.

2.2 The `save_all.sh` script features

As mentioned before, there are some things left to be desired with the earlier example. The actual `save_all.sh` script discussed here has all these features:

- The `save_all.sh` script will maintain *save_alls* for generally anything you can get into with the *Integrated Control Configurator*.
- The `save_all.sh` script, will maintain *save_alls* for all station types hosted by I/A 50 Series, I/A 70 Series, AP20's and PW's.
- The Directory administration is automatically maintained. If hosts, stations (or volumes) are added to the configuration, everything is taken care of the next session.
- You have full control over the selection of hosts, control stations, volumes you want to make *save_alls* for and full control of the number of backups.
- You can perform an upload for all or any subset of control stations.
- If a previous *save_all* exists, it will be backed up (compressed and timestamped).
  The directory will be emptied before a new *save_all* takes place to allow a fresh start.
- The script will maintain 10 backups for every station and volume found on the system.
- The available diskspace is determined prior to the *save_all* action to preserve at least some critical filesystem space required for I/A Series Software to operate and will inform the user if the available space will not accommodate the requested *save_all* operation.
- If the *save_all* fails you will be notified of this and the latest backup will be restored to the default *save_all* location for that particular station or volume.
- Extensive logging is part of the `save_all.sh` script to screen and log file.
- Entering non-existing station names and/or volumes will be checked by the script and the user will be informed of this. I.e.: all user data entered at the command-line is validated by the `save_all.sh` script during operation.

In addition to this:
- The Control block sequences found in I/A Series can make use of the `#include` mechanism to allow for easier sequence code generation. These *sequence include* files are never part of a standard `save_all` but they are required when editing and compiling Sequence code.
  The `save_all.sh` script goes through all of the source code files on the harddisk to locate these include files and stores them in a separate location.
- Availability of hosts and stations/volumes is reported by the script so the operator will be able to take actions.
- The script always attempts to make as many *save_alls* as possible. This means that the script will always try to recover from errors it encounters and proceed with the next station or volume.
- An optional configuration file may exist that overrides some of the default settings in the script.

All of the features here are available without ANY user configuration. All is done by entering one simple command: **`./save_all.sh all<cr>`**.

3. Installation

Although the process is not too complex, some notes about preparing your I/A Series system for the save_all.sh script.

3.1 I/A Series requirements

The requirements for the **save_all.sh** script are:
- Your I/A System is NOT configured for the new IACC.
- The script must be installed and run on an I/A 50 or 70 Series AW or AP. It will not run on a WP, PW-any or AP20! (You can run the script on a 50 Series AP or AW and maintain *save_alls* for stations hosted by an AP20 or PW).
- The running I/A Series software version must be 4.3 or higher.
- In an all I/A 50 Series (UNIX) environment, installing and running the save_all.sh script on one (1) host is sufficient.
- In a mixed I/A 50/70 Series or I/A 70 Series only environment you should install and run the save_all.sh script on every I/A 70 Series host AW and on at least one (if your systems contains one that hosts Control Stations) I/A 50 Series host.
- Your I/A Series System reflects the current **System Definition** configuration. This is very important because the script relies on the system files containing configured stations, volumes and hosts to be correct. These system files are distributed through the Committed Install process.
- The I/A 70 Series platform can be based on Windows NT 4.0 or Windows XP Professional.
- The I/A 50 Series platform can be based on SUN Solaris 2.5.1 or Solaris 8.0.

I/A 70 Series hosts can only maintain *save_alls* for stations and volumes hosted locally.
Station and volumes hosted by I/A 70 Series CANNOT be accessed from other hosts.

3.2 Available files

The save_all.sh script is made available in several ways. Due to the "open" nature of the tool we have no intent to control this. The most common way it through a ZIP archive usually named **save_all.zip**.
When the **save_all.zip** file was distributed it contains at least these files:
1. save_all.sh The actual script which is the only one required when the defaults are acceptable.
2. save_all.rel which is the optional configuration file. This must be configured and renamed to **save_all.cfg** to be used by the save_all.sh script.
3. **save_allxxx.pdf** which is the manual for the save_all.sh and save_all.cfg files where **xxx** is the manual version and NOT an indication of the content.
But there may be other files...

The save_all.sh script can be copied to a location of your personal choice. This manual assumes that the script will be located in "/opt/tools" (I/A 50 Series) or "d:\opt\tools" (I/A 70 Series), however any location is fine. Beware that the script MUST reside on the D: drive in an I/A 70 Series environment.

For I/A 50 Series hosts you may find it better to unzip the archive on a DOS/Windows platform since the unzip utility may not be available on your host.

The script assumes that approximately 2Mb of discspace is required per control database and the script maintains 10 backups of the *save_alls* for all these stations. As a result, assuming a 5x compression, we get a diskspace requirement of approximately 2Mb + (10 x 400k) = 6Mb per station.

3.2.1 Installing on I/A 50 Series (UNIX based) hosts:

1. Unpack the ZIP archive.
2. Choose one or more I/A 50 Series AW/AP station(s) (In a "all UNIX" system: one will do, more is an option).
3. Start a vt100 session on the AW or AP.
4. Create or find a home directory for the tool. (for example /opt/tools).

5. Copy the `save_all.sh` and `save_all.rel` files to this location.
6. Make `save_all.sh` executable (**chmod 700 save_all.sh**).
7. Execute "**save_all.sh<cr>**" without any parameters: If you get usage info the script is correctly installed.

3.2.2 Installing on I/A 70 Series (NT/XP based) hosts:

1. Locate all I/A 70 Series (NT) host AW/AP stations.
2. Open a Windows Command box and make drive D: the active drive.
3. Create or find a home directory for the tool on drive D: (for example `D:\opt\tools`).
4. Copy the `save_all.sh` and `save_all.rel` files to this location.
5. Start a Korn shell (**sh<cr>**) and in the `/opt/tools` directory, execute "**./save_all.sh<cr>**" without any parameters: If you get usage info the script is correctly installed.

FOXBORO

4. Usage and Configuration

After putting everything in place, you are ready to run the script. Note that, by default, no changes are made to your system files and control databases (i.e.: no upload is done). Uploads are not performed by default but must be forced via a commandline option or in the configuration file.

The save_all.sh script is self-explanatory and provides usage info when invoked without any parameters:

```
1AWB11#./save_all.sh<cr>
Version 2.2.1 / June 28, 2003

Usage: save_all.sh  -s string [[string] [...]] [-h string [[string] [...]] [-u][-b]
  save_all.sh all|local [-b backups] [-h string1 string2] [-u]
  Use regular expression similar to grep ( no * ):
  Where: -u        = force upload first
          -s string = [part of] station or volume letterbug
          -b number = number of backup's to be maintained
          -h string = [part of] host letterbug
Example:  save_all.sh all
   Make a save_all for ALL stations by ALL hosts in
   /opt/SAVEALLS, maintaining 10 backups.
Example:  save_all.sh local
   Make a save_all for ALL stations by this hosts in
   /opt/SAVEALLS, maintaining 10 backups.
Example:  save_all.sh -s P30 -b 4 -u
   Do an upload and make a save_all for stations:
   CP3011, MGCP30, ACP30A, ..., in /opt/SAVEALLS, maintaining 4 backups:
Example:  save_all.sh -h 1AWB11 all
   will make a save_all for all stations hosted by 1AWB11
   in /opt/SAVEALLS maintain 10 backups. (the "all" parameter is optional)

1AWB11#
```

If the response is like shown, you have installed the save_all.sh script correctly.

4.1 Available command line switches

The save_all.sh script takes at least one command line switch. What they mean and what they do is explained here:

- **all**
  Creates a *save_all* for all stations and volumes on the system. By default the script will maintain 10 backups of all stations and volumes.
  Example: save_all.sh **all**<cr>.

- **local**
  Creates a *save_all* for all stations and volumes for this host only. By default the script will maintain 10 backups. On I/A 70 Series this switch gets you the same results as the **all** switch above.
  Example: save_all.sh **local**<cr>.

- **-u**
  Perform an upload of the Control Database. This is the same function as found in the Control Configurator when doing an upload from the *Maint* menu pick. This parameter requires at least one other parameter selecting Control Stations or Hosts.
  Example: save_all.sh **-u** all<cr>.

- **-s string [string]**
  Make a *save_all* for all stations and volumes matching **string**. This can be a letterbug but also just a part of a letterbug which the desired stations have in common. You can put more than one string on the command line.
  Example: save_all.sh **-s ACU CTL CP60**<cr>
  will select all stations that contain ACU, CTL or CP60 in their letterbug/name **for all hosts**.
  A little trick can be to do "-s `cat /opt/mystations`" to use a list of stations to process.

- **-h string [string]**
  The same function as the -s option but this time it is for hosts selection.
  Example: save_all.sh **-h AW5101 AP00**<cr>
  will make a *save_all* for **all stations and volumes hosted by AW5101 and all other hosts matching AP00**.

- **-b number**
  This switch will make the script to maintain "number" backups. This parameter requires at least one other parameter selecting Control Stations or Hosts.
  Example: save_all.sh all **-b 45**<cr>
  will make the script maintain 45 backups for every station and volume on the system.

Combinations of these parameters are supported for example:

    1AWB11# **save_all.sh -s 30 vol -h AW5011 -u -b 50<cr>**

To make a *save_all* for all stations containing "*the 30* and *vol*" on the *host AW5011*", with an "*upload*" and "*maintain 50 backups*":

Note for I/A 70 Series based hosts:
The Foxboro I/A 70 Series platform will not allow remote *save_alls* and as a result you must (when having more that one I/A 70 Series station hosting Control stations) install and run the save_all.sh script on every I/A 70 Series host.

4.2 Changing the script defaults

Sometimes the default save_all.sh script does not behave as desired for instance:
- You have a special location for your system backups and it is NOT /opt/SAVEALLS!
- Sometimes you just want to maintain 100 backups without specifying this on the commandline every time.
- Your databases are very large and the default 2Mb reservation does not cover the diskspace usage.
- Maybe you are in a specific situation where you want to *save_all* only a subset of the stations and/or hosts in your system.
- You ALWAYS want to upload the database for all stations.
- You want to run *save_alls* for the local host only.
- You do not want any volumes to be saved.
- The log file should go somewhere else than the current directory.

These kind of problems can of course be solved by editing the save_all.sh script, but this is not a task many like to perform if it were only for the time consuming task to find out where to make the edits without breaking something else.
Instead we came up with the external configuration file for this job. This configuration file is always consulted by the script at execution time and if it exists, it will be validated and override the script defaults where applicable. An example of the configuration file is part of the **save_all.zip** file and is named save_all.rel.

4.2.1 The Configuration file requirements

The requirements for the configuration file are:
- It must be (re)named to save_all.cfg! You can use the example save_all.rel file as a starting point.
- It must be in the directory where you execute the save_all.sh script.
- The files that the save_all.cfg refers to (like the CPFILE and LBFILE) must be valid. These files contain one letterbug on each line.
- Empty variables are set to the save_all.sh default.
- Variables not mentioned in the save_all.rel example are not supported.

The file layout is like this:

```
[save_all.rel]
# Example save_all.cfg file
##########################################################################
# START of file
# Here make you own choices for the standard parameters
# DO NOT use the "root" directory for anything at all!!
# This file will not be overwritten by new version save_all.sh script
# Format:
```

```
# VARIABLE=VALUE
#***********************************************************************
# This file name MUST be save_all.cfg to work with save_all.sh version 2.x
# and MUST be located in the directory where you execute save_all.sh
#***********************************************************************
SA_DIR=/opt/SAVEALLS
CPFILE=/etc/cplns
LBFILE=/etc/lbls
APFILE=/etc/aplns
ARCHNR=15
UPLOAD=FALSE
STATKB=2000
```

4.2.2 The settable defaults

There are some settings we thought we would make available to the outside world. The supported user variables and what they mean:

- **`SA_DIR=path`**
  This is where the *save_alls* will be stored and it is the base of the `save_all.sh` administration. This directory will be created if it does not exsist when invoking the `save_all.sh` script with one or more valid options. The path can not be "/", the default `/opt/SAVEALLS` will be used in that case.
  Example: `SA_DIR=/opt/customer/application/control`
  Default: `SA_DIR=/opt/SAVEALLS`

- **`CPFILE=file`**
  This variable points to a file where `save_all.sh` can find the control stations to maintain *save_alls* for. This overrides the ability for the script to find any stations not listed in **`file`**! It is recommended NOT to alter this entry. When **`file`** is empty, NO Control Stations will be saved! By default this variable points to a file that is maintained by the **System Definition** program.
  Example: `CPFILE=/opt/customer/mystations.txt`
  Default: `CPFILE=/etc/cplns`

- **`APFILE=file`**
  This variable points to a file that contains the available hosts in the system. This overrides the ability for the script to find any hosts not listed in **`file`**! It is recommended NOT to alter this entry! When **`file`** is empty, NO Hosts will be found and thus no *save_all* for ANY station or volume! By default this variable points to a file that is maintained by the **System Definition** program.
  Example: `APFILE=/opt/customer/myhosts.txt`
  Default: `APFILE=/etc/aplns`

- **`LBFILE=file`**
  This variable points to a file that contains the available volumes in the system. This overrides the ability for the script to find any volumes not listed in **`file`**! It is recommended NOT to alter this entry. When **`file`** is empty, NO Volumes will be saved! By default this variable points to a file that is maintained by the **System Definition** program.
  Example: `LBFILE=/opt/customer/myvolumes.txt`
  Default: `LBFILE=/etc/lblns`

- **`ARCHNR=number`**
  The **`number`** of backups to maintain. The script defaults to 10. When the `save_all.cfg` file is in place the default is set to **`number`**. Entering an invalid value here like "`fortyfive`" will result in reverting to the default of 10 backups.
  Example: `ARCHNR=30`
  Default: `ARCHNR=10`

- **`UPLOAD=TRUE/FALSE`**
  By default the script will **not** do an upload for the stations selected. Setting the value to TRUE forces an upload for all selected stations. This can be VERY time consuming but if this is what you want then set this variable to "**`TRUE`**". There is no commandline switch to disable uploads.i.e.: When you set **`UPLOAD=FALSE`** you can do UPLOADS with the `-u` option from the command line, but with **`UPLOAD=TRUE`** there is NO way to disable the upload!
  Example: `UPLOAD=TRUE`
  Default: `UPLOAD=FALSE`

- **STATKB=number**
  This variable defines the number of kilobytes diskspace is required per station for the *save_all*. This is an estimate and as such may be a little optimistic. This number is used only to calculate the amount of space that could be required for the *save_all*. For example: When you make a *save_all* for 10 stations, the space requirement is 10 times 2000 bytes which results in a estimated required amount of 20.000 bytes free on the harddisk. The script will check if this amount of space is available prior to making the *save_all*. Be conservative with this one.
  Example: `STATKB=4000`
  Default: `STATKB=2000`
- **LOG=<path>/<file>**
  This variable defines the *save_all* logfile name and location.
  Example: `LOG=/opt/customer/applics/logs/save_all.log`
  Default: `LOG=$CURDIR/save_all.log`

4.2.3 A Configuration file example

A Configuration file could be something like this:

```
[save_all.cfg]
SA_DIR=/opt/customer/control/backup/
CPFILE=
LBFILE=
APFILE=
ARCHNR=52
UPLOAD=FALSE
STATKB=3500
LOG=/etc/save_all.log
```

In this example *save_alls* for all configured stations, hosts and volumes will be made, the number of backups is set to 52 to allow one year of backups when running *save_alls* once a week. The *save_alls* are going to be stored under `/opt/customer/control/backup` and 3500 bytes per *save_all* is reserved when calculating the required filesystem space. The log file can be found in the `/etc` directory

4.3 Save_all examples

Some typical examples of the `save_all.sh` script being used.
Remember that these basic administration tasks (the values shown are based on the defaults) are always carried out:
- A directory `/opt/SAVEALLS` will be created.
- In that directory you will find a directory for every station and volume configured on your system so you should see directories like:
  `/opt/SAVEALLS/CP6001` and:
  `/opt/SAVEALLS/volwrk` etc. will be in here.
- A directory for the backup *save_alls* i.e.: `/opt/SAVEALLS/backup`.
- A directory for the include files `/opt/SAVEALLS/include`. For each station found, there will be a file in there with the station name.
- A directory named `/opt/SAVEALLS/log` where all the log files are maintained.
- `#include` file referred to in your sequences are searched and if found, are placed in a tar file in the "include" directory.
- 10 backups are maintained unless you specify another number in the example.

4.3.1 Making a save_all for "all" stations and volumes

The simplest implementation of making automatic *save_alls*: "Make *save_alls* for all Control Stations from all hosts and maintain 10 backups". This would be a nice one to start with and this exactly what the next command will do for you:

```
1AWB11#save_all.sh all<cr>
```

Entering this command on an I/A 50 Series host will make a *save_all* in `/opt/SAVEALLS` for all stations and volumes found in `/etc/cplns` and `/etc/lblns`.

On an I/A 70 Series hosts, the function is the same except you will only make *save_alls* for volumes and stations that are hosted locally by the AW running the *save_all*. The system files containing the exsisting stations, volumes and hosts on a specific I/A Series system are maintained by the **System Definition** process and as such are declared valid for use by the save_all.sh script.

4.3.2 Make save_all for volumes only

To maintain *save_alls* for all "volumes" found on your system run the command:

```
1AWB11#save_all.sh –s vol<cr>
```

This will select all stations containing "vol" as part of the letterbug or name and only "*volumes*" conform to this which will get you a *save_all* for all volumes only.

4.3.3 Make save_all for stations and volumes one or more hosts

To maintain *save_alls* for a selection of hosts run the command:

```
1AWB11#save_all.sh –h AW5101 AW5014 50<cr>
```

and you will get *save_alls* for all stations and volumes hosted by both the **AW5101** and **AW5014** and all hosts containing **50** in their letterbug.

4.3.4 Make save_all for station and volumes on the current host

To maintain *save_alls* for all station and volumes hosted by this host only, run the command:

```
1AWB11#save_all.sh local<cr>
```

and you will get *save_alls* for all stations and volumes hosted by the **1AWB11**.

4.3.5 Making save_all some stations only

Making *save_alls* for a subset of control stations can be done by running the command:

```
1AWB11#save_all.sh –s MG 71<cr>
```

This will get you a *save_all* for all stations and volumes on the system containing either **MG** or **71**. For example **MG300B**, **GW7160**, **vol071** etc.

4.4 Running scheduled and unattended save_alls

It would be very nice to run the save_all.sh script at fixed intervals taking a boring but important job off your hands and still get it done anyway!
Both the I/A 70 Series and the I/A 50 Series platform provide the functionality for this and both do this different of course!

4.4.1 Scheduling save_alls on I/A 50 Series hosts

The I/A 50 Series platform is UNIX based. This platform provides "**cron**" as the scheduler for repeating tasks with "**crontab**" beeing the tool to maintain the configuration file.
Make sure you are logged in as "**root**" to edit the crontab file for "**root**".

To use "**vi** " as the editor that will be used to make the changes, you may need to set the shell variable EDITOR in either the *Bourne* or the *C-shell*.
In the *Bourne shell* it is done like this:

```
1AWB11#EDITOR=vi ; export EDITOR<cr>
1AWB11#crontab –e<cr>
```

In the *C-shell* it is done like this:

```
1AWB11#setenv EDITOR vi<cr>
1AWB11#crontab –e<cr>
```

With the command crontab –e, you start the crontab editor. An entry in crontab could be:

```
# Automatic save_all procedure:
```

```
30 2 * * 0   cd /opt/tools ; ./save_all.sh all -b 25 > /dev/null 2>&1
```

After making the modification exit **vi** with "**ESC :wq**" to install the *crontab* file and notify the *cron* dæmon of this fact.

After completion you will run the save_all.sh script at **02:30 hours on every Sunday** for all stations on all hosts and maintain 25 backups.

4.4.2 Scheduling *save_alls* on I/A 70 Series hosts

On the I/A 70 Series (Windows NT) platform, the scheduler is a service that may or may not be running on your system. To find out if the scheduler is running, open a "Command prompt box" and enter the command: at<cr>

There can be different responses to this command,
If you see this:

```
C:\>at<cr>
There are no entries in the list.
```

You are ready to configure and add scheduled commands, however on a standard I/A Series system it is quite likely to get this response:

```
C:\>at<cr>
The service has not been started.
```

In which case we must make sure the **Schedule** or **Task Scheduler** service is started at boot time. This can be done by following this procedure:

1. Open the **Control Panel** and locate/open the **"Services"** icon.
2. In the list on the left part of the screen, locate the "**Schedule** or **Task Scheduler**" service (it probably shows "manual").
3. Click on the "**Startup**" button to get the dialog,
4. Check the box that reads "**Automatic**" and then click **OK**.
5. Manually start the service.
6. Run the "at" command again to see if the response has changed to "There are no entries in the list". When the response is correct you are ready to proceed with the scheduler configuration.

Since "at" or scheduled commands cannot be edited, it may be better to take a two step approach by creating a small batch file that contains the *save_all* command with all the desired parameters and the "at" command only executes this batch file at a specific time and interval.

This way you can make changes to your *save_all* command without having to re-enter the entire "at" command.

So the first step is to make a small "batch" file that will carry out the actual *save_all*.

Here is a sample make_sa.bat file which will be called in the at command:

```
[make_sa.bat]
D:
cd \opt\tools
sh ./save_all.sh all -b 25
exit
```

An example "at" command:

To run the save_all.sh script every **Sunday at 02:30 hour** enter this (until **<cr>**):

```
C:\>at 02:30 /every:sunday "d:\opt\tools\make_sa.bat"<cr>
Added a new job with job ID = 1
```

To check the line was added to the scheduler run "at" command once again:

```
C:\>at<cr>
Status ID   Day                  Time        Command Line
-------------------------------------------------------------------------
        1   Each Su              02:30       d:\opt\tools\make_sa.bat

C:\>
```

Another example:

To run the save_all.sh script every **first day of the month at 09:00 hour** enter this (until **<cr>**):

```
C:\>at 09:00 /every:1 "d:\opt\tools\make_sa.bat"<cr>
Added a new job with job ID = 1
```

This will make the *save_alls* on a regular basis.

FOXBORO

5. Error and status messages

The `save_all.sh` script will always attempt to make as many *save_alls* as possible. Because of this, the script will try to overcome any problems encountered and deal with most events occurring and will try to report these.

"Errors", as these problems and events are sometimes called, are reported to the screen and a little more detailed in the log file when applicable. The log file usually contains at least the information sent to the screen and is always found in the directory where you execute the script. (This not necessarily the same as where script is located).

In this section you can find some of the more common errors reported. The test log that these were taken from are from our lab system where almost nothing works so it's great for testing ;-).

5.1 Log file header

The log file contains some statistics as where the *save_alls* are going, which stations match the selection criteria and where they are hosted and etc.

A little example log file after issuing **save_all.sh –h 1AWB11** at the prompt:

```
Matching hosts:
1AWB11
Stations selected for save_all:
Station      Host name
1C10BF       1AWB11
1C3AB1       1AWB11
1C3ABF       1AWB11
1C3BB1       1AWB11
1C3BBF       1AWB11
1C4AB1       1AWB11
1C4ABF       1AWB11
1C4BB1       1AWB11
1C4BBF       1AWB11
1C6AB1       1AWB11
1C6ABF       1AWB11
1C10B1       1AWB11
1IS3B1       1AWB11
volwrk       1AWB11
Available space on system:         483859 kb
Estimated space required on system: 28000 kb
####################################################
# Automatic save-all procedure started
# Version 2.21 / Feb 18, 2003
# Date:               Tue Feb 18 09:24:26 GMT 2003
# Started on:         1AWB11
# Save-all Location:  1AWB11  /opt/SAVEALLS
# Backup location:    /opt/SAVEALLS/backup
# Include file location: /opt/SAVEALLS/include
# External config file: /opt/tools/save_all.cfg
####################################################
####################################################
 Save_all starting for: 1C10BF hosted by 1AWB11
----------------------------------------------------
Found a previously stored save_all for this station.
Making back-up first
...backing-up 1C10BF
Back-up is: /opt/SAVEALLS/backup/11Apr2002_1C10BF.tar.Z
Save_all for 1C10BF started
Etc,etc
```

5.2 No save_all program found.

The complete error message as it is shown on screen:

```
No save_all program found.
This program must be run from 50/70 Series AP or AW.
aborting...
```

When you get this message you are, most likely, running the script on an AP20 or WP. The required program and API are not available on those platforms.

5.3 Processing "<path>/save_all.cfg" file.

Whenever you have a file named `save_all.cfg` located in the directory where you execute the `save_all.sh` script, you get this line on screen and in the logfile:

```
Processing "/opt/tools/save_all.cfg" file.
```

When this is the case, the script will process the `save_all.cfg` file and use the override values set in there.

5.4 No station(s) specified after –s

You entered the `-s` parameter but did not specify a `string` to select stations or volumes. The script will exit with usage info.
Omitting the `-s` parameter will select as many stations and volumes as possible with respect to the `-h` parameter.

5.5 No host(s) specified after -h

You entered the `-h` parameter but did not specify a `string` to select one or more hosts. The script exits with usage info. If all hosts are required, it is best to omit this parameter.

5.6 -b option is not a valid/number value, reverting to default of 10

You entered the `-b` option (for the number of backups to maintain) but did not enter a valid number on the commandline. This must be a positive numeric value. An incorrect entry will make the script continue with the default of 10 backups to make sure that automated entries with an incorrect `-b` entry won't stop the *save_all* process.

5.7 Script must be executed from D-drive

On I/A 70 Series hosts only this indicated that the `save_all.sh` script was installed on another than the D: drive. It must be installed on that drive to function.

5.8 Not enough space available for selected stations

If the disk space is not adequate to store the *save_alls* for the selected stations/volumes, you get this message in the log file (and on screen):

```
Available space on system:          205595 kb
Estimated space required on system: 234000 kb
Not enough space available for selected stations
```

The disk must be cleaned up. Maybe you can reduce the estimated diskspace requirements in the `save_all.cfg` file or reduce the number of backups that will be maintained or you could get yourself a AW51D,E,F,G box with a 72Gig RAID5 disk.
This feature was built into the script because we did not want the disk to be filled up with *save_alls*. So there is a default of 10 backups and we check for the required space before proceeding.
Note that diskspace requirements are calculated for the selected stations and volumes only so you could try to make a smaller selection that reduces the number of stations matching the selection criteria.

5.9 No previously stored save_all found to back up

This is not an error. This indicates that the script did not encounter an old *save_all* for the station or volume reported.

```
##################################################
 Save_all starting for: 1C10BF hosted by 1AWB11
-------------------------------------------------
No previously stored save-all found to backup.
Save_all for 1C10BF started
```

This should better not be reported every time of course. If it does this usually indicates a problem with the script. You get this message for every station and volume when the script runs for the very first time.

5.10 There were errors for <STATION>

When a station does not respond, an error is reported:

```
There were errors for 1C10BF.
Check /opt/tools/saveall.log for more details.
```

The log file will contain the error as reported by the ICC driver task:

```
There were errors for 1C10BF.
FAIL     1 open 1C10BF!: Tue Feb 18 08:31:00 2003 -14 ICCopen error: class= 1 er
ror= 0 text= icConnectCp: icCpConnect problem
```

The error above indicates the station is not found. This is definitely NOT good in a production system.

5.11 Database locked, skipping <STATION>

When you see this message in the log file (and on screen), the station or volume was found in use by another ICC session. It can be that the station is opened in the Control Configurator on some other screen.

```
###################################################
 Save_all starting for: 1C3AB1 hosted by 1AWB11
---------------------------------------------------
Database locked, skipping 1C3AB1
###################################################
```

If this message shows up every time the script is run it could mean the ICC was not properly closed and the "lock" file is still there when it shouldn't be.

5.12 <HOST> is not responding or unavailable

This should not happen in a normal system unless the HOST is down for maintenance (during tape backup for instance).

```
Contacting 3AP51F...
3AP51F is not responding or unavailable
```

You really should check this out. If the host is no longer part of the configuration you should update your System Definition to match reality.

5.13 No hosts matching selection criteria

When the selection after the −h parameter did not result in a matching host you get this message after which the script will exit with usage info:

```
1AWB11# save_all.sh -h 20
Version 2.21 / Feb 18, 2003
Processing "/opt/tools/save_all.cfg" file.
No hosts matching selection criteria
```

You probably made a typo. Remember that LETTERBUGS are case sensitive on I/A 50 Series (UNIX) hosts, which means the option:

```
# save_all.sh −h aw51<cr>
```

will not ever result in any matching hosts on I/A 50 Series. All letterbugs/logical names must be entered in UPPERCASE. On I/A 70 Series hosts (which are not case sensitive), this is not an issue and it can only be *your* fault if your selection doesn't find any stations!
If you are sure that the host specified does exsist on your system, check the save_all.cfg file. If this points to an APFILE that does not contain the desired letterbug you will get this message.

5.14 No stations matching selection criteria

When the selection after the −s parameter did not result in a matching station or volume you get this message after which the script will exit with usage info:

```
1AWB11# save_all.sh -s 12
Version 2.2.1 / June 10, 2003

Processing "/opt/tools/save_all.cfg" file.
Creating station letterbug file...
```

```
    No stations matching selection criteria

    1AWB11#
```

The same note as applies as with the previous message concerning case sensitive letterbugs. On I/A 50 Series, all letterbugs/logical names must be entered in UPPERCASE
If you are sure that the station(s) or volume(s) specified do(es) exsist on your system, check the `save_all.cfg` file. If this points to an `CPFILE` or `LBFILE` that does not contain the desired letterbug you will get this message.

5.15 There were errors with include files!!!

The *save_all* itself will not be affected by this error.
When the script tries to locate the `#include` files referred to in the sequence files, it may encounter errors. This is not a fatal error but we still thought it was worth mentioning. You get a message on screen reading:

```
    There were errors with include files!!!
    Check /opt/tools/saveall.log for more details.
```

There can be two kinds of errors in the log file.

5.15.1 Include file <path/filename> does not exist

This message indicates that an "include file", referred by one or more of the sequence logic files, could not be found:

```
    WARNING: Include file /opt/fox/ciocfg/sequeninclude/thisfile.inc
    does not exist!!
```

The script attempts to locate these `#include` files on the station's host processor. If the files are not found there, you get this error message. The *save_all* itself will not be affected by this.

5.15.2 File <path/filename> contains relative paths

This error message is related to the use of relative path addressing include files in the sequences:

```
    File /rem/AW5101/opt/fox/ciocfg/sequeninclude/DROGER/BRANDER contains relative
    paths:
                  * SOURCE    : ../SEQUENINCLUDE/DROGER    *
```

At the moment this manual was written, we are unable to handle this kind of sequence programming. This may be solved in a later version.
So when creating sequences the `save_all.sh` script has some constraints:

Including files in sequences must be done with absolute paths which implicates that:
this syntax: `#include "../include/myincludefile"` will not work but:
This syntax: `#include "/opt/fox/ciocfg/include/myincludefile"`
or this: `#include "myincludefile.inc"` will.
(this will locate the include file in `/usr/fox/ciocfg/sequeninclude` or `/opt/fox/ciocfg/sequeninclude` depending on the host platform.

The *save_all* itself will not be affected by this error.

5.16 Restoring most recent save_all for <STATION>

Nothing happened to your Control Station. In this case the script did find a previous *save_all* it has created a backup for it. If the new *save_all* fails for any reason, the script will restore that backup to the default *save_all* directory for this station or volume:

```
    ##################################################
     Save_all starting for: 1C10BF hosted by 1AWB11
    --------------------------------------------------
    Found a previously stored save_all for this station.
    Back-up is: /opt/SAVEALLS/backup/11Apr2002_1C10BF.tar.Z
    Save_all for 1C10BF started
    There were errors for 1C10BF.
```

FOXBORO

```
Restoring most recent save_all for 1C10BF
Most recent save_all restored
###################################################
```

Since the station directory is emptied prior to a new *save_all* we would end up with an empty directory. This procedure assures that the `/opt/SAVEALLS/1C10BF` directory contains at least the last successful *save_all* for this station.

This should only happen when the station was reported OFF LINE during the `save_all.sh` script execution. In an ordinary production environment this is not very likely to happen. Note that the station is found OFF LINE when this happens and this is not the type of error message you get when the station was in use by the Integrated Control Configurator!

When you check out the log file, you get the error message as the ICC API reports it:

```
###################################################
 Save_all starting for: 1C10BF hosted by 1AWB11
---------------------------------------------------
No previously stored save-all found to backup.
Save_all for 1C10BF started
There were errors for 1C10BF.
FAIL     1 open 1C10BF!: Tue Feb 18 08:31:00 2003 -14 ICCopen error: class= 1 er
ror= 0 text= icConnectCp: icCpConnect problem
###################################################
```

This is something to investigate.

5.17 <NUMBER> archived save_all(s) will be deleted (10 second delay)

Since the `save_all.sh` script maintains a maximum configured number of backups, the oldest one will be deleted once this maximum is reached and this is done silently. However, when you have configured (in crontab/scheduler) to maintain 25 backups on the commandline (i.e. not in the `save_all.cfg` configuration file) and an engineer attempts to run `save_all.sh` with the defaults (still set to maintain 10 backups) the script will be forced to throw away 15 valuable backups.

```
15 archived save_all(s) will be deleted
Press CTRL-C to abort
Continuing in 10 seconds
```

When this happens the user gets 10 seconds to abort. If there is no response within that 10 seconds, the old *save_alls* will be deleted . This procedure is repeated for every station and volume selected with a 10 sec. timeout for every station or volume selected!

The script is made to create as many *save_alls* as possible and a *save_all* overrules keeping an old backup which is why this is not considered a "show stopper".

5.18 Remote host access not supported on 70 Series

Whenever you select a station or volume that is hosted by I/A 70 Series host other than your local machine you get this message in the log file (and on screen):

```
###################################################
 Save_all starting for: 2CP601 hosted by 2AW702
---------------------------------------------------
Remote host-access not supported on 70 Series
###################################################
```

It's because remote host access is not supported on I/A 70 Series :-(. The implication of this is that you should install and run the `save_all.sh` script on every I/A 70 Series AW hosting volumes, integrators or stations.

5.19 This system appears to be configured for IACC

(Version 2.2.1 of `save_all.sh` and higher)

The `save_all.sh` script was around before the IACC was. *Save_alls* created with `save_all.sh` are not compatible with **IACC** (the new **I**ntelligent **A**utomation **C**onfiguration **C**omponent, part of ArchestrA).

In order to make sure we don't break anything the script will check for the presence of IACC and will exit:

```
This system appears to be configured for IACC.
```

```
Check for Control_Cfg entry under /usr/fox/config when in doubt.
Save_all script can't continue, exiting...
```

As you can see, the script assumes that if the **Control_Cfg** entry is missing in /usr/fox/config on the host where you run the script, IACC is in place.

5.20 Cannot create log file with this name: <path>/<name>

This message appears whenever you have used the option "LOG" in the external configuration file and specified a log file that cannot be created. This can be because a directory with the same name already exsists. The script will exit when this happens and you will not get any usage info either. Choose another name and try again.

5.21 Unknown argument or syntax error.

You screwed up. Read the manual once more from the start or at least try to decipher the usage display. If you are not that kind of person, read the next chapter which gives you the source, so be prepared.

FOXBORO

6. The `save_all.sh` script.

The save_all script in its ugliest form

[**save_all.sh**]

```
#!/bin/sh
VERSION="Version 2.2.1 / June 12, 2003"
##############################################################################
# Script to create and maintain the system wide savealls in a pre-configured
# directory on the local harddrive of an AP or AW.
# (see variable list for current location)
#
# A creative conception by:    M. de Waal(MdW) & R. Deen (RD)
# Special guest appearance by: R. de Groot (RdG) & C. van Diepen (CvD)
# Thanks to B. Marsman (BM) for his valuable testing, input and bug fixes
#
# Additional functionality added by Stan Brown (SBD)
#
##############################################################################
#  Copyright (c)2000-2003 Invensys Process Systems/The Cassandra Project
#  All Rights Reserved.
#
#  This is free software; you can redistribute it and/or modify it
#  under the terms of the GNU General Public License as published by
#  the Free Software Foundation; either version 2 of the License, or
#  (at your option) any later version.
#
#  This software is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  merchantability or fitness for a particular purpose. See the GNU
#  General Public License (http://www.fsf.org/copyleft/gpl.htm)for
#  more details.
#
#  To obtain the GNU General Public License, write to:
#
#      Free Software Foundation, Inc.
#      59 Temple Place - Suite 330
#      Boston, MA  02111-1307
#      USA
##############################################################################
# Revision history:
# Feb 23, 2000 - Initial release.
# Feb 24, 2000 + Added usage information.
#              + Added backup procedure for older save_all's.
#              - A lot of testing, seems OK.
#              + Added log-file: saveall.log.
#              - Major clean-up of original script.
# Feb 25, 2000 - A lot of problems solved, it seems to work now!
#              - Compressing the backup tar files to save space.
#              - All tests pas%&&*!@~~!@#$!____!!@$$#%%$^&^^&
#                core dumped.
# Feb 28, 2000 - Let's call it Version 1.0 (Amsterdam)!
#              + Fixed: Scripts did not work in crontab, added ./
#              + Added check for control station and remote hosts.
# Feb 29,2000  - Version 1.1 (Breda)!
#              + Added 20 Series host support
#              + Cleaned up the script. Tested remote stations handling.
#              + Improved reporting to screen and log file.
# Mar 01,2000  - Version 1.2 (Coevorden)!
#              + Added Windows NT Support, Added archive management.
#              + Added free space check.
# Mar 02,2000  - Version 1.3 (Deventer)!
# (RD)         + Enhanced NT Support. Improved reporting.
#              + Bug fix: script would fail with no SAVEALL directory
#                (UNIX only).
# Mar 10,2000  - Version 1.3.2 (Deventer Noord-Oost)
# (BM)         + Bug fixed: Deletion of old backups was done BEFORE making
#                new backup, causing one backup more then specified.
#              + Timestamp for backups now saveall-time instead of backup-
#                time, including hour:minutes, so more then 1 backup per day
#                is possible.
# Aug 22,2000  - Version 2.0
# (MDW/SDB/RD) + Added -u option to include upload before save_all (SBD)
#              + Added the -h HOST option to make save_all's for HOST (RD)
#              + Fixed: restore the "before last" save_all when
#                last save_all failed (RD)
#              + Added recursive search for include files in sequences (MDW)
#                Include files are stored in "$SA_DIR/include"
# Nov 09, 2000 + Time of backup fixed (only occurred on NT-platform)
```

```
# Feb,Mar 2001  + Remote host includes added
# (MDW/RD)      + Includes saved with "relative" paths
#               + Cleaned up script and tested on NT platform
#               + Added external config file support (filename: save_all.cfg)
# Dec 12, 2001 - Version 2.1
# (MDW/RD)      + Bug fix. Timeout was set to 0 after first CP with surplus
#                 backups, resulting in no prompt/timeout for all others.
#               + Bug fix. No validation done on user CPFILE file.
#               + Enhancement: show if external "save_all.cfg" used at startup.
#               + Enlarged STATKB from 2000 to 5000 kb.(estimated reqd diskspace
#                 per station)
#               + Bug fix. Remote locked stations would not be
#                 identified as such.
#               + Bug fix. Diskspace check always counts all controlstations,
#                 even if only a few where selected.
#               + Renamed all tmp files, added "sa_" prefix
#               + Removed "schrink" command from "upload" function, reason was
#                 possible workfile corruption.
#               + Added include search for SFC files
# May 2002     - Verson 2.2
# (RD)          + Version  new variable "PF" for temp files (prefix).
#               + Added save_all of volumes in this version. (by request)
#               + Modified create save_all dir using mkdir -p command.
#               + Entering bogus -b option, no longer stops save_all,instead
#                 continues with default value in script or config file.
#               + Added few cleanup statements when script would exit.
#               + Logfiles are now saved with save_all date and old logs are
#                 in "log" directory.
#               + Some minor fixes to logfile.
# Jan 2003     - Version 2.21
# (MDW/RD)      + Bug fix. Sequence include search could not
#                 handle #include <filename> syntax
#                 Usually the syntax is #include "filename"
#               + Modified: Include files were always searched on "remote
#                 station", even if the host was the local station
#               + Bug fix. Temp files where not cleared after exit. Could result
#                 in multiple save_all's.
#               + Added feature. Script is now IACC aware. Will not start if
#                 system is prepared for the new IACC platform.
#               + Bug fix. Changed the crontab example because the cfg file
#                 would not be found the script home directory
# Apr 2003     + Also search for include files relative to the ciocfg directory
# (MDW)          Many people still use this form of defining include files, or
#                it was 'already' like that in the sequencecode. So we
#                implemented this searchpath at your desire.
# May 2003     + Added rudimentary interrupt handling. At least cleanup tmp
# (RD)           files after CTRL_C and CTRL_D.
#               + Added "local" command line switch to save_all all stations
#                 hosted by local host only.
#               + Added -tst parameter for debugging purpose. Keeps tmp files.
############################################################################
# BUGS/CONSIDERATIONS/WISHES:
#
# - To run from cron, redirect output to /dev/null 2>&1 for instance:
#   30 2 * * 6   cd /opt/tools ; ./save_all.sh -h AW5101 -s MG3 -b 25 > /dev/null
2>&1
#
# - When forcing 25 backup's with a line in crontab, executing a commandline
#   save_all with -b is less than 25 will remove the surplus backups!! User is
#   warned when this happens and gets $TIMEOUT seconds to abort per station.
#
#  -Including files in sequences must be done with absolute paths
#   Correct:   #include "../include/thefile.inc"
#   Incorrect: #include "/opt/fox/ciocfg/include/thefile.inc"
#   Incorrect: #include "thefile.inc" (this will locate the include file in
#                            /usr/fox/ciocfg/sequeninclude
#                        or  /opt/fox/ciocfg/sequeninclude)
#                        depending on the host platform.
#
# - No fix yet for config files for Profibus stored in
#   /usr/fox/sp/files/devices
#
############################################################################
# Example save_all.cfg file
############################################################################
# # START of file
# # Here make you own choices for the standard parameters
# # DO NOT use the "root" directory for anything at all!!
# # This file will not be overwritten by new version save_all.sh script
# # Format:
# # VARIABLE=VALUE
```

```
# #******************************************************************************
# # This file name MUST be save_all.cfg to work with save_all.sh version 2.x
# # and MUST be located in the same directory as save_all.sh
# #******************************************************************************
#
# SA_DIR=/opt/SAVEALLS
# CPFILE=/etc/cplns
# LBFILE=/etc/lblns
# APFILE=/etc/aplns
# UPLOAD=FALSE
# ARCHNR=15
# STATKB=2000
# # END of file
################################################################################
################################################################################
# Variable definition starts here
################################################################################
PF="sa221_tmp"                          # Prefix for tmp files.
PLATFORM=`uname -s`                      # define platform
SN_NAME=`uname -n`                       # this station's name
TIMESTAMP=`date '+D%m%dT%H%M'`           # The date and time now
CURDIR=`pwd`                             # the current directory.
LOG=$CURDIR/save_all.log                 # use this for logfile
PROG=`basename $0`                       # Returns this script's name without
                                         # the path
SA_DIR="/opt/SAVEALLS"                   # Location of save_alls.
CPFILE="/etc/cplns"                      # List for selected stations (all).
APFILE="/etc/aplns"                      # List for selected hosts (all).
LBFILE="/etc/lblns"                      # List for selected volumes (all).
LBCPFILE=`cat "$CPFILE" "$LBFILE"`       # System file with all control station,
                                         # (not just CP's), DO NOT EDIT.
TRUE_APFILE="/etc/aplns"                 # System file with all application
                                         # processors, DO NOT EDIT.
SLDB="/usr/fox/sp/sldb"                  # Control Station / host relation
USRCP="FALSE"                            # No user CPFILE by default
USRAP="FALSE"                            # No user APFILE by default
USRLB="FALSE"                            # No user LBFILE by default
HLDB="/usr/fox/sp/hldb"                  # Station ID file for configured stations
UPLOAD="FALSE"                           # No upload by default. Thanks Stan Brown
HOST_SLCT="no"                           # No host filter, All hosts selected
PATH=$PATH:/usr/local                    # Add /usr/local to PATH
EXEC_DIR="/opt/fox/ciocfg/api"           # the location of the api utils.
LOCK_DIR="/usr/fox/sp/locks"             # the lock files
CIO_DIR="/opt/fox/ciocfg"                # CIOCFG directory
SEQINCL="/opt/fox/ciocfg/sequeninclude/" # Sequence include directory
ICC="/usr/fox/config/Control_Cfg"        # Where is the ICC located
STATKB=2000                              # minimum of free kbytes needed per
                                         # station (estimate!!!)
ARCHNR=10                                # default number of backup archives in
                                         # backup dir
BIDLIST="|207|C002|"                     # Stations without remote host support.
                                         # Syntax is |HLDB_Value_for_bad_station
                                         # Note pipe symbol. Currently applies to
                                         # NT only (ID's are from $HLDB file)
TIMEOUT=10                               # General time-out time
CFG=$CURDIR/save_all.cfg                 # External configuration file
TESTRUN="FALSE"                          # a test run parameter
if [ "$PLATFORM" = "Windows_NT" ]        # Define executable Names
then
    SA_EXEC="save_all.ksh"
    SORT="/nutc/mksnt/sort"
    UNIQ="/nutc/mksnt/uniq"
    FIND="/nutc/mksnt/find"
else
    SA_EXEC="save_all"
    SORT="/usr/bin/sort"
    UNIQ="/usr/bin/uniq"
    FIND="/usr/bin/find"
fi
# Remove all the temp-files
rm $CURDIR/"$PF"* >/dev/null 2>&1
################################################################################
# Handle interrupts
################################################################################
trap "echo
      echo Interrupt received! aborting save_all... ;\
      echo Removing temp-files
      rm $CURDIR/"$PF"* >/dev/null 2>&1
      echo If save_all was in progress, station may be locked!
      exit 5  " 1 2 15
################################################################################
```

```
# Show version and determine if external configuration is in order
################################################################################
echo "$VERSION"
echo
################################################################################
# If IACC is used, skip the save_all script
################################################################################
if [ ! -d $ICC ]
then
    echo "This system appears to be configured for IACC." | tee -a $LOG
    echo "Check for Control_Cfg entry under /usr/fox/config when in doubt." | tee -a
$LOG
    echo "Save_all script can't continue, exiting..." | tee -a $LOG
    # Remove all the temp-files
    rm $CURDIR/"$PF"* >/dev/null 2>&1
    exit 1
fi
################################################################################
# External config file will override the script defaults
################################################################################
# look for external config file ($CFG)
if [ -f $CFG ]
then
    echo "Processing \"$CFG\" file." | tee -a $LOG
#
    USRSA_DIR=`cat $CFG | grep SA_DIR | cut -d"=" -f2 | sed s/\"//g`
    if [ "$USRSA_DIR" != "" ] && [ "$USRSA_DIR" != "/" ]
    then
        SA_DIR=$USRSA_DIR
    fi
    USRCPFILE=`cat $CFG | grep CPFILE | cut -d"=" -f2 | sed s/\"//g`
    if [ "$USRCPFILE" != "" ] && [ -f "$USRCPFILE" ]
    then
        CPFILE=$USRCPFILE
        USRCP="TRUE"
    fi
    USRAPFILE=`cat $CFG | grep APFILE | cut -d"=" -f2 | sed s/\"//g`
    if [ "$USRAPFILE" != "" ] && [ -f "$USRAPFILE" ]
    then
        APFILE=$USRAPFILE
        USRAP="TRUE"
    fi
    USRLBFILE=`cat $CFG | grep LBFILE | cut -d"=" -f2 | sed s/\"//g`
    if [ "$USRLBFILE" != "" ] && [ -f "$USRLBFILE" ]
    then
        LBFILE=$USRLBFILE
        USRLB="TRUE"
    fi
    USRARCHNR=`cat $CFG | grep ARCHNR | cut -d"=" -f2 | sed s/\"//g`
    if [ "$USRARCHNR" != "" ]
    then
        ARCHNR=$USRARCHNR
    fi
    USRUPLOAD=`cat $CFG | grep UPLOAD | cut -d"=" -f2 | sed s/\"//g`
    if [ "$USRUPLOAD" = "TRUE" ]
    then
        UPLOAD=$USRUPLOAD
    fi
    USRSTATKB=`cat $CFG | grep STATKB | cut -d"=" -f2 | sed s/\"//g`
    if [ "$USRSTATKB" != "" ]
    then
        STATKB=$USRSTATKB
    fi
fi
# Define the "supporting" directories here
BU_DIR=$SA_DIR/backup            # Location of backup save_alls.
PB_DIR=$SA_DIR/profibus          # Location of Profibus config files.
LOG_DIR=$SA_DIR/log              # Location of old log files.
INC_DIR=$SA_DIR/include          # sequence include files directory
################################################################################
# Cleanup and manage old log files
################################################################################
# Manage ARCHNR log files
LOGFILE=`basename $LOG`
mv $CURDIR/$LOGFILE $LOG_DIR/$LOGFILE$TIMESTAMP > /dev/null 2>&1
# No need to keep more than ARCHNR logfiles I think?
LOGLVL=`ls -l $LOG_DIR/$LOGFILE*|wc -l` > /dev/null 2>&1
DELLVL=`expr $LOGLVL - $ARCHNR`
if [ $DELLVL -gt 0 ]
then
    for OLDLOG in `ls -lt $LOG_DIR/$LOGFILE* | tail -$DELLVL | awk '{print $9}'`
```

```
     do
       ls $OLDLOG
       rm $OLDLOG
     done
fi
################################################################################
# Define some functions first
################################################################################
# Function to display command usage and exit
################################################################################
usage_exit ()
{
    echo "Usage: $PROG  -s string [[string] [...]] [-h string [[string] [...]] [-
u][-b backups]"
    echo "  $PROG all|local [-b backups] [-h string1 string2] [-u]"
    echo "  Use regular expression similar to grep ( no * ):"
    echo "  Where: -u        = force upload first"
    echo "           -s string = [part of] station or volume letterbug"
    echo "           -b digit  = number of backup's to be maintained"
    echo "           -h string = [part of] host letterbug"
    echo "Example:  $PROG all"
    echo "   Make a save_all for ALL stations by ALL hosts in"
    echo "   "$SA_DIR", maintaining "$ARCHNR" backups."
    echo "Example:  $PROG local"
    echo "   Make a save_all for ALL stations by this hosts in"
    echo "   "$SA_DIR", maintaining "$ARCHNR" backups."
    echo "Example:  $PROG -s P30 -b 4 -u"
    echo "   Do an upload and make a save_all for stations:"
    echo "   CP3011, MGCP30, ACP30A, ..., in "$SA_DIR", maintaining 4 backups:"
    echo "Example:  $PROG -h `uname -n` all"
    echo "   will make a save_all for all stations hosted by `uname -n`"
    echo "   in "$SA_DIR" maintain "$ARCHNR" backups. (the \"all\" parameter is
optional)"
    echo
    exit 1
}
################################################################################
# Function to do the upload/checkpoint
################################################################################
upload ()    # this function first builds the upload command file for
             # the upload then performs the actual uploading of parameters
             # of the specified CP to the workfile
             #
             # parameter:  $1  letterbug ID of CP to be uploaded
{
    CMD=$CURDIR/$1.upload
    ULOG=$CURDIR/$1.log
    echo OPEN $1 MODIFY byDuc > $CMD
    echo UPLOAD            >> $CMD
    echo CHECKPOINT 300    >> $CMD
    echo CLOSE             >> $CMD
    echo EXIT              >> $CMD
    echo "  `date` " >> $LOG
    echo "  Upload and checkpoint $1" >> $LOG
    ./iccdrvr.tsk -i $CMD -o $ULOG -n ECHO
    cat $ULOG >> $LOG
    rm $ULOG $CMD 2> /dev/null
}
################################################################################
# If no options, show usage info and exit.
################################################################################
if [ ! $# -gt 0 ]
then
    usage_exit
fi
################################################################################
# For the 70 Series platform: Verify that script is installed/executed
# from drive D:.
################################################################################
if [ "$PLATFORM" = "Windows_NT" ]
then
    DRIVE=`pwd|cut -c1`
    if [ "$DRIVE" != "D" ] && [ "$DRIVE" != "d" ]
    then
      echo "Script must be executed from D-drive" | tee -a $LOG
      echo
      exit
    fi
fi
################################################################################
# Because Control configurator API is not available on WP's or Venix,
```

```
# check for executable presence, no use to go on if it's not there...
############################################################################
if [ ! -f $EXEC_DIR/$SA_EXEC ]
then
    echo "No "$SA_EXEC" program found." | tee -a $LOG
    echo "This program must be run from 50/70 Series AP or AW." | tee -a $LOG
    echo "aborting..."| tee -a $LOG
    exit 2
fi
############################################################################
# Set some defaults first: select all stations on all hosts (unless user
# overrides are in place)
############################################################################
for NAME in `echo $LBCPFILE`
do
    echo $NAME >> "$CURDIR"/"$PF"lbcplns
done
TRUE_CPFILE="$CURDIR"/"$PF"lbcplns      # System file with stations and vols
                                        # DO NOT EDIT.
cp $TRUE_CPFILE $CURDIR/"$PF"cplns
cp $APFILE $CURDIR/"$PF"aplns
CPFILE=$TRUE_CPFILE
############################################################################
# Start command-line evaluation
############################################################################
while [ $# -gt 0 ]
do
    case $1 in
      all)
          # This selection indicates all stations in the variable CPFILE
          echo "Save-all for all stations in $CPFILE." | tee -a $LOG
          if [ "$USRCPFILE" = "$CPFILE" ]
          then
             for STA in `cat $CPFILE`
             do
               # Check if the stations are valid for this system
               grep $STA $TRUE_CPFILE >> "$CURDIR"/"$PF"cpln
             done
             # Remove possible duplicate letterbugs due to selection criteria
             $SORT "$CURDIR"/"$PF"cpln|$UNIQ > "$CURDIR"/"$PF"cplns
             CPFILE="$CURDIR"/"$PF"cplns
             if [ ! -s $CPFILE ]
             then
                 echo "No stations matching selection criteria" | tee -a $LOG
                 echo
                 # Remove all the temp-files
                 rm $CURDIR/"$PF"* >/dev/null 2>&1
                 exit 1
             fi
          else
             cat "$TRUE_CPFILE"
             cp "$TRUE_CPFILE" "$CURDIR"/"$PF"cplns
             CPFILE="$TRUE_CPFILE"
          fi
          shift
      ;;
      local)
          HOST_SLCT="yes"
          # This selection indicates selection of local host only
          while [ $# -gt 0 ] && [ `echo $1|cut -c1` != "-" ]
          do
              HOST_NAMES=`grep "$SN_NAME" "$APFILE"`
              echo "$HOST_NAMES" > "$CURDIR"/"$PF"aplns
              shift
          done
          HOSTFILE="$CURDIR"/"$PF"aplns
          echo "Matching hosts:" | tee -a $LOG
          cat $HOSTFILE | tee -a $LOG
          echo
      ;;
      -b)
          # Determine number of backups.
          # move the -b parameter out of the way
          shift
          if [ $# -gt 0 ]
          then
              USRARCHNR=`expr $1 + 0 2>/dev/null`
              if [ "$?" != "0" ] || [ "$ARCHNR" -lt 1 ]
              then
                  echo "-b option is not a number value, revert to default of
""$ARCHNR" | tee -a $LOG
```

```
                    ARCHNR="$ARCHNR"
                  fi
              else
                  echo "-b option is not a valid value, revert to default of ""$ARCHNR" |
tee -a $LOG
                  ARCHNR="$ARCHNR"
              fi
              shift
        ;;
        -u)
              # Do upload before a save_all
              shift
              UPLOAD="TRUE"
              # export UPLOAD
        ;;
        -s)
              # Only save_all some stations.
              # move the -s parameter out of the way and see what we have
              shift
              # the next parameter should be (part of) a Control station lbug
              # and the number of params should be 1 or greater
              if [ ! $# -gt 0 ]
              then
                  echo "No station(s) specified after -s" | tee -a $LOG
                  echo
         usage_exit
              fi
              echo "Creating station letterbug file..." | tee -a $LOG;echo
              while [ $# -gt 0 ] && [ `echo $1|cut -c1` != "-" ]
              do
                  # Check if the stations are valid for this system.
         grep $1 $TRUE_CPFILE >> "$CURDIR"/"$PF"cpln
         shift
              done
              # Remove possible duplicate letterbugs due to selection criteria
              $SORT "$CURDIR"/"$PF"cpln|$UNIQ > "$CURDIR"/"$PF"cplns
              # When we have a user CPFILE do something more.
              if [ "$USRCP" = "TRUE" ]
              then
                  for CP in `cat $CURDIR/"$PF"cplns`
                  do
                    grep $CP $USRCPFILE >> $CURDIR/"$PF"usrcplns
                  done
                  CPFILE="$CURDIR"/"$PF"usrcplns
              else
                  CPFILE="$CURDIR"/"$PF"cplns
              fi
              if [ ! -s $CPFILE ]
              then
         echo "No stations matching selection criteria" | tee -a $LOG
         echo
                  # Remove all the temp-files
                  rm $CURDIR/"$PF"* >/dev/null 2>&1
         exit 1
              fi
        ;;
        -h)
              HOST_SLCT="yes"
              # This selection indicates selection of hosts
              # move the -h parameter out of the way to see which hosts
              shift
              # the next parameter should be (part of) a Host lbug
              # and the number of params should be 1 or greater
              if [ ! $# -gt 0 ]
              then
                  echo "No host(s) specified after -h" | tee -a $LOG
                  echo
         usage_exit
              fi
              while [ $# -gt 0 ] && [ `echo $1|cut -c1` != "-" ]
              do
                  HOST_NAMES=`grep $1 $APFILE`
                  echo $HOST_NAMES >> $CURDIR/"$PF"apln
                  shift
              done
                  for T in `cat $CURDIR/"$PF"apln`
                  do
                    echo $T >> $CURDIR/"$PF"hostfile
                  done
                  if [ ! -s $CURDIR/"$PF"hostfile ]
                  then
```

```
              echo "No hosts matching selection criteria" | tee -a $LOG
                    echo
                    rm $CURDIR/"$PF"* > /dev/null 2>&1
                    usage_exit
                fi
                # Remove possible duplicate hosts due to selection criteria
                $SORT "$CURDIR"/"$PF"hostfile|$UNIQ > "$CURDIR"/"$PF"aplns
                HOSTFILE="$CURDIR"/"$PF"aplns
                echo "Matching hosts:" | tee -a $LOG
                cat $HOSTFILE | tee -a $LOG
                echo
        ;;
        -tst)
            # if set do not make savealls but exit after running init.
            TESTRUN="TRUE"
            shift
        ;;
        *)
            echo "Unknown argument or syntax error." | tee -a $LOG
            usage_exit
        ;;
    esac
done
if [ ! -f $CURDIR/"$PF"cplns ]
then
   echo "No stations specified!" | tee -a $LOG
   echo
   usage_exit
fi
#############################################################################
# printing a list of all the stations / hosts to be processed
#############################################################################
echo;echo "Stations selected for save_all:" | tee -a $LOG
echo "Station       Host name" | tee -a $LOG
for CP in `cat $CPFILE`
do
    HOST=`awk '{if ($1==cp) {print $2} }' cp=$CP < $SLDB`
    if [ $HOST_SLCT != "yes" ]
    then
       echo "$CP        $HOST" | tee -a $LOG $CURDIR/"$PF"cpap
    else
       echo "$CP        $HOST" >> $CURDIR/"$PF"cpap
    fi
done
# When there was a selection of hosts the HOST_SLCT variable is yes
if [ $HOST_SLCT = "yes" ]
then
    rm "$CURDIR"/"$PF"scratch > /dev/null 2>&1
    for AP in `cat "$HOSTFILE"`
    do
       cat "$CURDIR"/"$PF"cpap | grep $AP >> "$CURDIR"/"$PF"scratch
    done
    # let's see what we have
    cat "$CURDIR"/"$PF"scratch | tee -a $LOG
    cat "$CURDIR"/"$PF"scratch | awk ' {print $1}' > $CUR_DIR/"$PF"cplns
    CPFILE=$CUR_DIR/"$PF"cplns
fi
#############################################################################
# Setting up the basic administration.
#############################################################################
#
# Check for all required directories and create if needed
echo "Checking directory administration, please wait..."
if [ ! -d $SA_DIR ]
then
    mkdir -p $SA_DIR
fi
if [ ! -d $BU_DIR ]
then
    mkdir $BU_DIR
fi
if [ ! -d $INC_DIR ]
then
    mkdir $INC_DIR
fi
if [ ! -d $LOG_DIR ]
then
    mkdir $LOG_DIR
fi
if [ ! -d $PB_DIR ]
then
```

```
    mkdir $PB_DIR
fi
echo "Done"
# Create a directory for all the configured control stations
#
echo "Creating station directory if required..."
for CP in `cat $CPFILE`
do
if [ ! -d $SA_DIR/$CP ]
then
    mkdir $SA_DIR/$CP
fi
done
echo "Done"
################################################################################
# Check for diskspace
################################################################################
STATNR=`cat "$CPFILE"|wc -l`
NEEDED=`echo "$STATNR * $STATKB" |bc`
if [ "$PLATFORM" = "Windows_NT" ]
then
    AVAILABLE=`df -k D: |tail -1|awk '{print $3}'|awk -F"/" '{print $1}'`
else
    AVAILABLE=`df -k $SA_DIR|tail -1 | awk '{print $4}'`
fi
echo "Available space on system:          ""$AVAILABLE"" kb" | tee -a $LOG
echo "Estimated space required on system: ""$NEEDED"" kb" | tee -a $LOG
if [ $NEEDED -gt $AVAILABLE ]
then
  echo "Not enough space available for selected stations" | tee -a $LOG
  # Remove all the temp-files
  rm $CURDIR/"$PF"* >/dev/null 2>&1
  exit 1
fi
################################################################################
# The actual save_all part starts here
################################################################################
echo
echo "##################################################"| tee -a $LOG
echo "# Automatic save-all procedure started"| tee -a $LOG
echo "# $VERSION" | tee -a $LOG
if [ $UPLOAD = "TRUE" ]
then
    echo "#  -> Performing upload before save-all"| tee -a $LOG
fi
echo "# Date:                 "`date` | tee -a $LOG
echo "# Started on:           "$SN_NAME | tee -a $LOG
echo "# Save-all Location:    "`uname -n`" "$SA_DIR | tee -a $LOG
echo "# Backup location:      "$BU_DIR | tee -a $LOG
echo "# Include file location:"$INC_DIR | tee -a $LOG
echo "# Writing logfile in:   "$LOG
    if [ -f $CFG ]
    then
echo "# External config file: "$CFG | tee -a $LOG
    fi
echo "##################################################"| tee -a $LOG
#
################################################################################
# If only a testrun exit here
################################################################################
if [ $TESTRUN = "TRUE" ]
then
    echo "Only a test run, no save_alls done."
    exit 4
fi
################################################################################
# Start with routine to make the save_alls
#

for CP in `cat $CPFILE`
do
   HOST=`awk '{if ($1==cp) {print $2} }' cp=$CP < $SLDB`
  if [ "$HOST" = "" ]
  then
    echo "##################################################"| tee -a $LOG
    echo "# $CP is not a valid station         " | tee -a $LOG
    echo "##################################################"| tee -a $LOG
    LOCK="unknown"
  else
    cd $EXEC_DIR
    echo "##################################################"| tee -a $LOG
```

```
    echo " Save_all starting for: $CP hosted by $HOST" | tee -a $LOG
    echo "---------------------------------------------------"| tee -a $LOG
    # Station may be in use by another ICC session
    # So, check if station is locked (also remote hosts)
    if [ "$HOST" = "$SN_NAME" ]
    then
      #echo "Station is hosted locally." | tee -a $LOG
      if [ -f $LOCK_DIR/*"$CP"+ ]
      then
        LOCK="yes"
      echo "Database locked, skipping $CP" | tee -a $LOG
    else
      LOCK="no"
    fi
      else
    # Check platform to see how to connect to remote host
    # Remote connection is not supported on NT!!!!
        STATID=`grep $HOST $HLDB|awk '{print $2}'`
        IDCHECK=`echo "$BIDLIST"|grep "|"$STATID`
    if [ "$PLATFORM" = "Windows_NT" ] || [ "$IDCHECK" != "" ]
    then
        echo "Remote host-access not supported on 70 Series" | tee -a $LOG
            rmdir "$SA_DIR/$CP" # Saveall dir not needed
        LOCK="unknown"
    else
        echo "Contacting $HOST..." | tee -a $LOG
        # Check if remote station is rmounted and where
        MOUNTPOINT=`rmount | grep $HOST | awk '{print $4}'`
        if [ "$MOUNTPOINT" = "" ]
        then
            if [ ! -d /rem/$HOST ]
            then
              mkdir /rem/$HOST
            fi
            rmount $HOST /rem/$HOST 2>/dev/null
                if [ "$?" -ne "0" ]
                then
                    echo "$HOST is not responding or unavailable" | tee -a $LOG
                    LOCK="unknown"
            elif [ -f /rem/"$HOST"/"$LOCK_DIR"/*"$CP"+ ]
            then
              LOCK="yes"
              echo "Database locked, unable to perform the save_all." | tee -a $LOG
            else
              LOCK="no"
            fi
                MOUNTPOINT=`rmount | grep $HOST | awk '{print $4}'`
            rumount $HOST 2>/dev/null
         else
            if [ -f "$MOUNTPOINT""$LOCK_DIR"/"$CP"+ ]
            then
              LOCK="yes"
            else
              LOCK="no"
            fi
        fi
      fi
    fi
      fi
    # Lock check complete
fi

    if [ -d $SA_DIR/$CP/*STA ] && [ $LOCK = "no" ]
    then
  echo "Found a previously stored save_all for this station." | tee -a $LOG
  echo "Making back-up first" >> $LOG
  echo "Making back-up first\r\c"
  NAME=`ls -ld $SA_DIR/$CP/*STA"" | awk '{print $7 $6 $8}' | sed s/\://`
  cd $SA_DIR
  echo "...backing-up $CP" >> $LOG
  echo "...backing-up $CP \r\c"
  tar cf $BU_DIR/$NAME"_"$CP.tar $SA_DIR/$CP > /dev/null 2>&1
      if [ -f $SA_DIR/include/$CP.tar ]
      then
        tar rf $BU_DIR/$NAME"_"$CP.tar $SA_DIR/include/$CP.tar >/dev/null
        rm $SA_DIR/include/$CP.tar
      fi
  compress -f $BU_DIR/$NAME"_"$CP.tar >/dev/null
  echo "Back-up is: ""$BU_DIR""/""$NAME""_""$CP.tar.Z" | tee -a $LOG
      # determine level of backups until now
      LVL=`ls -l $BU_DIR/*"$CP"*|wc -l` 2>/dev/null
      DEL=`echo "$LVL - $ARCHNR" |bc`
```

```
                if [ $DEL -gt 0 ]
                then
                    if [ $DEL -gt 1 ]
                    then
                      echo
                      echo ""$DEL" archived save_all(s) will be deleted"
                      echo "Press CTRL-C to abort"
                      TIMELEFT=$TIMEOUT
                      while [ $TIMELEFT -gt -1 ]
                      do
                        echo "Continuing in $TIMELEFT  \r\c";sleep 1;
                         TIMELEFT=`echo $TIMELEFT - 1|bc`
                      done
                      echo "Removing surplus archives" | tee -a $LOG
                    fi
                    for DELFILE in `ls -lt $BU_DIR/*"$CP"*|tail -$DEL|awk '{print $9}'`
                    do
                      rm $DELFILE
                    done
                fi
rm -r $SA_DIR/$CP/*
            BACKUP="TRUE"
      elif [ $LOCK = "no" ]
      then
echo "No previously stored save-all found to backup." | tee -a $LOG
            BACKUP="FALSE"
      fi
    if [ $LOCK = "no" ]
    then cd $EXEC_DIR
        if [ $UPLOAD = "TRUE" ]
        then
            echo "Starting upload/checkpoint....\r\c" | tee -a $LOG
            echo "Starting upload/checkpoint...." >> $LOG
            upload $CP
            echo "Upload & checkpoint ready        " | tee -a $LOG
        fi
echo "Save_all for $CP started" | tee -a $LOG
# Insert ./ to ensure the scripts runs OK from cron!
./$SA_EXEC $CP $SA_DIR/$CP > /dev/null &
PID=`echo $!`
#echo "PID= $PID"
VAR=0
while [ $VAR -ne 1 ]
do
  ps -p $PID > /dev/null  2>&1
  VAR=$?
  echo "|\r\c";sleep 1;echo "/\r\c";sleep 1
  echo "-\r\c";sleep 1;echo "\\ \r\c";sleep 1
done
if [ ! -f /tmp/output* ]
then
      #
      # Start with routine to search include files
      #
        if [ "$HOST" != "$SN_NAME" ]
        then
            MOUNTPOINT=`rmount | grep $HOST | awk '{print $4}'`
            if [ "$MOUNTPOINT" = "" ]
            then
              rmount $HOST /rem/$HOST 2>/dev/null
            fi
        fi
        HOST_HLDB=`grep $HOST $HLDB | awk ' {print $2} '`
        if [ $HOST_HLDB != "303" ]
        then
          SEQINCL="/opt/fox/ciocfg/sequeninclude/"
        else
          SEQINCL="/usr/fox/ciocfg/sequeninclude/"
        fi
        echo "Searching for sequence include files..." | tee -a $LOG
        # First search for main sequencefiles
        $FIND $SA_DIR/$CP -name "*.s" -print > $CURDIR/"$PF"2search
        ################################################################
        #ADDED FOR SFC SUPPORT
        $FIND $SA_DIR/$CP -name "*.k" -print >> $CURDIR/"$PF"2search
        ################################################################
        touch $CURDIR/"$PF"inc_lines
        touch $CURDIR/"$PF"found
        # Begin search of nested include files
        while [ -s $CURDIR/"$PF"2search ]
        do
```

```
                for FILE in `cat $CURDIR/"$PF"2search|$UNIQ`
                do
                  grep "#include" $FILE >> $CURDIR/"$PF"inc_lines
                  REL=`grep "\.\.\/" $FILE`
                  if [ -n "$REL" ]
                  then
                    echo "File $FILE contains relative paths:" | tee -a $LOG
                    echo "$REL \n" | tee -a $LOG
                  fi
                done
                ###############################################################
                # MODIFIED FOR SFC SUPPORT
                # Replace < and > by quotes
                sed -e 's/[<,>]/\"/g' $CURDIR/"$PF"inc_lines|cut -f2 -d "\"" >
$CURDIR/"$PF"presearch
                # ADDED FOR SFC SUPPORT
                # Replace backslashes with forward slashes and remove drive-letters
                sed "s/\\\/\//g" $CURDIR/"$PF"presearch > $CURDIR/"$PF"presearch2
                sed "s/\\D://g" $CURDIR/"$PF"presearch2 > $CURDIR/"$PF"search
                ###############################################################
                rm $CURDIR/"$PF"2search $CURDIR/"$PF"inc_lines >/dev/null 2>&1
                ERROR="FALSE"
                # Preventing script-looping by include loops
                for NAME in `cat $CURDIR/"$PF"search`
                do
                  INCGREP=`grep $NAME $CURDIR/"$PF"found`
                  if [ -z "$INCGREP" ]
                  then
                      CHAR1=`echo "$NAME"|cut -c1`
                      CHAR2=`echo "$NAME"|cut -c1,2`
                      if [ "$CHAR1" = "/" ] # Absolute path of sequence include file
                      then
                        if [ ! -f "$MOUNTPOINT$NAME" ]
                        then
                          echo "WARNING: Include file \"$MOUNTPOINT$NAME\" does not
exist!!" >> $LOG
                          ERROR="TRUE"
                        else
                          echo "$MOUNTPOINT$NAME" >> $CURDIR/"$PF"2search
                          echo ".$NAME" >> $CURDIR/"$PF"found
                        fi
###############################################################################
                      elif [ "$CHAR2" = ".." ] # Relative path of sequence include
file
                   then
                          # Remove the 2 dots before the file/dir name
                      NAME_REL=`echo $NAME|cut -c3-200`
                      # Only look for includes relative to the CIOCFG dir.
                      # If relative path's are otherwise defined in sequences
                      # that's just bad and sloppy programming!!
                      echo "Looking for relative include file:\n $NAME"  >> $LOG
                      echo " in\n $MOUNTPOINT$CIO_DIR$NAME_REL"  >> $LOG
                          if [ -f "$MOUNTPOINT$CIO_DIR$NAME_REL" ]
                          then
                            echo "$MOUNTPOINT$CIO_DIR$NAME_REL" >> $CURDIR/"$PF"2search
                            echo ".$CIO_DIR$NAME_REL" >> $CURDIR/"$PF"found
                        echo "Found: $MOUNTPOINT$CIO_DIR$NAME_REL" >> $LOG
                          else
                            echo "WARNING: Include file \"$MOUNTPOINT$CIO_DIR$NAME_REL\"
does not exist!!" >> $LOG
                          ERROR="TRUE"
                     fi
                        else # No pathname = default path to include directory
                          if [ -f "$MOUNTPOINT$SEQINCL$NAME" ]
                          then
                            echo "$MOUNTPOINT$SEQINCL$NAME" >> $CURDIR/"$PF"2search
                            echo ".$SEQINCL$NAME" >> $CURDIR/"$PF"found
                          else
                            echo "WARNING: Include file \"$MOUNTPOINT$SEQINCL$NAME\"
does not exist!!" >> $LOG
                          ERROR="TRUE"
                          fi
                        fi
###############################################################################
                  fi
                done
              done

        # If ERROR is true, this means one of the include files in this
        # save-all wasn't found
          if [ "$ERROR" = "TRUE" ]
```

```
        then
            echo "There were errors with include files!!!"
        echo "Check $LOG for more details."
        fi
        # Creating a nice sorted list
        $SORT $CURDIR/"$PF"found|$UNIQ > $CURDIR/"$PF"uniq
        rm $CURDIR/"$PF"found
        # Store include files if there are any
        if [ -s $CURDIR/"$PF"uniq ]
        then
            #tar cf $INC_DIR/$CP.tar -I $CURDIR/"$PF"uniq >/dev/null 2>&1
            if [ -n "$MOUNTPOINT" ]
            then
              cd $MOUNTPOINT
            else
              cd /
            fi
            tar cf $INC_DIR/$CP.tar  `cat $CURDIR/"$PF"uniq` >/dev/null 2>&1
            cd $CURDIR
        fi
        # The Quest of The Lost Includes completed
    echo "Save-all for $CP successfully completed." | tee -a $LOG
  else
    echo "There were errors for $CP." | tee -a $LOG
    echo "Check $LOG for more details."
    cat /tmp/output* >> $LOG

        # Checking presence of SA_DIR and CP to prevent an rm -r "root dir"
        if [ $BACKUP = "TRUE" ] && [ -n "$SA_DIR" ] && [ -n "$CP" ]
        then
            echo "Restoring most recent save_all for $CP" | tee -a $LOG
            uncompress $BU_DIR/$NAME"_"$CP.tar.Z
            rm -r $SA_DIR/$CP >/dev/null 2>&1
            tar xf $BU_DIR/$NAME"_"$CP.tar
            rm $BU_DIR/$NAME"_"$CP.tar
            echo "Most recent save_all restored"
        fi
  fi
    fi
done
echo "###################################################" | tee -a $LOG
echo "# Automatic save-all procedure finished" | tee -a $LOG
echo "# Date:"`date` | tee -a $LOG
echo "###################################################" | tee -a $LOG
# Cleanup after work is done (don't try this at home)
# Remove all the temp-files
####################################################################
rm $CURDIR/"$PF"* >/dev/null 2>&1 # Security issue!!!!???????
####################################################################
```