Foxboro

by Schneider Electric

Foxboro Evo™
Process Automation System

Software Utilities

B0193JB *M*

B0193JB

Rev M
December 12, 2014

SOFTWARE LICENSE AND COPYRIGHT INFORMATION

Contents

Tables

Preface

The purpose of this document is to provide a top level overview of I/A Series® software utilities supported on The Mesh and Nodebus network releases for I/A Series software for both the UNIX and Windows® platforms. The command descriptions and command options are documented only for the most recent version of each software utility. Certain software utilities were originally developed on the Nodebus Unix and Windows NT platforms and later ported to The Mesh Solaris™ and Windows XP® (and later) platforms with new commands and options added. This document does not distinguish which software utilities and versions are supported on which platforms.

Most of the software utilities are supported on all platforms and are located in `/opt/fox/bin/tools` with a few exceptions, such as the `bpatch` utility, which is installed in `/usr/fox/disp_tools`. Use the search facilities for your particular operating system (for example, Windows Explorer or Solaris 10 File System Utility) to search for any software utility which is not found in `/opt/fox/bin/tools`.

— NOTE

If the search results fail to locate a software utility, assume that the utility is not supported for your I/A Series system.

Command descriptions assume that you are an experienced UNIX or Windows user and familiar with those aspects of the I/A Series system that these utilities affect. To access a utility, start a UNIX terminal window or a Windows command prompt, go to the directory in which the tool resides, and type the appropriate shell level command invocation.

This document organizes the utilities into chapters for these functional categories: Object Management, System Management, Miscellaneous, and Legacy Historian.

— NOTE

Application Processors (APs) with the Solaris™ operating system are not allowed on The Mesh control network.

Document Conventions

This document uses the following conventions.

[parameter]	optional parameters
as is	verbatim text
<placeholder>	user-supplied values
|	alternative parameters
...	repetition of a parameter

Revision Information

For this release of the document (B0193JB-M), the following changes were made:

Chapter 2 "System Management Utilities"

- Updated section "dbvu280" on page 24.

1. Object Management Utilities

This chapter covers software utilities that use the Object Manager (OM) API or internal interfaces to access I/A Series Objects, which include OM Objects, Application Objects (AO), and Control and I/O (CIO) Objects.

The utilities in this document can be used to work with object types including the following:

- ♦ OM objects - flat named objects of object type *Alias*, *Process*, *Device*, *Variable*, and *OM Socket*.
- ♦ Application Objects (AOs) - hierarchical named objects of object type *Variable* in the form of `application:object.attribute`.
- ♦ CIO objects - hierarchical named objects of object type Variable in the form of `compound:block.parameter`.

Utilities that examine the OM databases and execute basic primitive operations are also included.

The table below lists the Object Management Utilities with a brief functional description.

Command	Description
oma	A menu-driven interface that provides user access to I/A Series objects in an I/A Series environment. This interface appears when the oma command is issued at the shell level.
omcrt	Creates specified OM objects
omdel	Deletes specified OM objects
omfnd	Checks for the existence of specified I/A Series objects
omget	Retrieves the value, status, and timestamps for a specified I/A Series object
omgetimp	Provides the same functionality as the omget utility and also imports an I/A Series object to eliminate multicast operations on future data access operations
omset	Sets the value and/or status for a specified I/A Series object
omsetimp	Provides the same functionality as the omset utility and also imports the I/A Series object to eliminate multicast operations on future data access operations
omary50	Establishes a user interface identical to the Pulp and Paper industry tool omary
rsom	Provides a user interface to examine the OM databases (for example, omopen lists, address tables, etc.) in control stations
som	Provides a user interface to examine the OM databases (for example, omopen lists, address tables, etc.) in the local workstation

Command descriptions appear on the following pages. To see an online usage summary for each of these commands, type the utility name with no parameters. The oma, rsom, and som utilities contain online help. In addition, many of these utilities have an associated `.man` file on Unix systems that is also contained in the same directory as the software utility.

NOTE
Application Objects and CIO Objects only support object type *Variable*.

oma

The oma utility provides a menu-driven interface to I/A Series objects in an I/A Series environment. This interface appears when you issue the oma command at the shell level.

From the oma interface, you can issue a variety of directives that invoke OM API library functions to inspect and manipulate I/A Series objects - for example, compound and block parameters for CIO Objects. OM objects created by FoxView or Display Manager variables are also available through the oma command.

Compound and Block parameters reside in a Control Processor. OM objects reside in I/A Series workstations.

The oma Environment

The oma environment provides four pages, each of which displays up to 30 object names. You can add, delete, and modify objects. Pages are identified by number (0 to 3) and each has an associated current file. To examine this file, type two question marks (??) while the page is visible. To go to a page, type its number (0 through 3). To go to a page with the current file, type the number twice.

Identifying Objects

When you add an object to the menu, you must provide a full pathname, a slot number and, optionally, a data type. The pathname is converted to upper case and is added to the oma names menu. You can use either the pathname or slot number to identify the object in subsequent operations.

Several commands let you create OM objects which will persist after you exit the oma environment.

Slot Numbers

To create a slot number, use the add command (a or aa). A slot number identifies an object relative to the menu page that is current. You can also use absolute slot numbers with several commands; these identify an object with reference to the beginning of the menu. An absolute slot number identifies the menu page as shown in the following table.

Table 1-1. Absolute Slot Numbers

abs_slot_no	page	menu slot number
0-29	0	0-29
30-59	1	0-29
60-89	2	0-29
90-119	3	0-29

For example, the command **k**, which removes a name from the menu, requires a slot number as a parameter; thus, the sequence:

```
1
k 15
```

removes an object from slot 15 on page 1.

The command **kk** also removes a name from the menu, but operates on absolute slot numbers. To remove the name in slot 15 on page 1, type:

```
kk 45
```

Data Types

When you assign a slot number, you can also specify a data type for the object. A data type specification is a single character from the following table.

Table 1-2. Data Type Specifications

Type Character	Length	Data Type
C	1 byte	character
I	2 bytes	integer
F	4 bytes	float (default)
S	n bytes	string
B	1 byte	Boolean
L	4 bytes	long integer
T	1 byte	short integer
P	2 bytes x 2	packed Boolean
M	4 bytes x 2	packed long
X	1 byte x 16	Boolean array of 16 elements
Y	4 bytes x 16	integer array of 16 elements
Z	4 bytes x 16	real array of 16 elements.

Accessing Objects

You can access objects via oma commands in three ways.

1. Via the OM GETVAL library function.
   - Use the **g** command to get the value of a specific object. If the data type of the retrieved object differs from the menu, oma corrects it internally. (Use **d** to update the menu.)
   - Use the **b** (bulk get) command to get the values for all objects listed on the menu.
   - Use the **j** command to get the value and print it in hexadecimal and as a 32-bit pattern. This command is used for the ALMSTA and BLKSTA parameters.

If retrieval is unsuccessful with the `b` command, `oma` displays a blinking `-1`.

- ♦ Use the `v` command to get the **`erh-key`** and **`open-id`** of every 16 bits in a specified volatile monitor table.
- ♦ Use the `u` command to get the active pattern for all 16 bits.
- ♦ Use the `t` command to get the trip pattern for all 16 pairs of 2-byte integers.

2. Via the OM SET_CONFIRM library function.

   Use the `s` command to assign a value to an object.

   For most data types, the value is decimal, hexadecimal, octal, real, character, or string. For packed boolean and packed long, hexadecimal is assumed. If you do not provide a value, `oma` uses the value retrieved with the last `g` command. The value must be compatible with the object's data type. For example, you can assign a hexadecimal value only to objects of data type INT, SHORT, or LONG.

3. Via the OM library functions, OMWRITE and OMREAD.

   For read and write operations, you are responsible for opening and closing files as appropriate. See the command summaries for the commands you need.

   The value must be compatible with the object's data type. For example, you can assign a hexadecimal value only to objects type variable of data type INT, SHORT, or LONG.

Command Summaries

The following table summarizes all the `oma` commands. Commands can refer to three kinds of files. The suffixes suggested for these are given below:

1. full path name files `*.fpn`
2. `oma` command files `*.oma`

   Command files can contain comments. Use the character # to delimit comment text.

3. echo files `*.ech`

   An echo file records your interaction with `oma` when the echo feature is turned on.

Parameter values are represented as follows:

<file name>	file identifier
<slot>	slot number (0 to 29)
<absolute slot>	slot number (0 to 119)
<list_no>	0 to 3
<access>	`r` (read), `w` (write), or `b` (both read and write)
<type>	data type (upper or lower case letter identifying data type)
<value>	the parameter value

NOTE

When using the ‘s’ and ‘ss’ commands to set a packed-boolean or a packed-long parameter, the value should be a hexadecimal mask followed by a hexadecimal value. When using the ‘@’ command to set up the set value for a packed-boolean or a packed-long parameter, the value should be just a hexadecimal a value. An OM write list can only include configurable input parameters, supervisory setpoint parameters, or object type variable. Use ‘*’ as a placeholder when there is no intention to set the value.

<status>[[~]a] [[~]b] [[~]o]

where

a	to turn on its ACK status bit
~a	to turn off its ACK status bit
b	to turn on its BAD status bit
~b	to turn off its BAD status bit
o	to turn on its OOS status bit
~o	to turn off its OOS status bit

Omit any status bit that does not need to change.

NOTE

The CP software, as part of the control algorithms, automatically clears the BAD bit, the ERROR bit, and (conditionally) the OOS bit when setting a parameter value without setting its status bits. Use a ‘\’ as a place holder when there is no intention to set the status bits.

Table 1-3. Command Summaries

Command	Action	Syntax
a	add object name	a <slot> <fpn> [<type>]
4	add sequence-block name and parameters ACTIVE, MA, STMO, OP-ERR	4 <slot> <compound:block>
5	add sequence-block name and parameters ACTIVE, MA, STMO OP-ERR, and STMRQ	5 <slot> <compound:block>
k	remove object name	k <slot>
kk		kk <absolute slot>
ka	clear menu	ka

Table 1-3. Command Summaries (Continued)

Command	Action	Syntax
r | rr	read names from file	r <file name> rr <file name>
w | ww	write names to file	w <file name> ww <file name>
:	switch to the *vi* text editor to edit current file (for `.fpn` and `.command` files)	:
n | nn	read commands from a `.command` file	n <file name> nn <file name>
g	get value	g <slot|file name>
gg		gg <absolute slot>
b	bulk get	b
j	print 32 bits and hex	j <slot>|<file name>
jj		jj <absolute slot>
t	get trip pattern (for monitor TRIPAT)	t <slot>|<file name>
tt		t <absolute slot>
u	get case_act_patt (for monitor ACTPAT)	u<slot>|<file name>
uu		uu <absolute slot>
v	set volatile monitor table (for monitor VOLMON)	v <slot>|<file name>
vv		vv <absolute slot>
s	set confirm	s <slot>|<file name>=[<value>[<status>]]
ss		ss <absolute slot>=[<value>[<status>]]
m	make a list (setup)	m <list_no> <access> <slot> [{+ <slot>}]
i	ignore list (free)	i <list_no>
p	print list	p <list_no>
o	open list	o <list_no>
c	close list	c <list_no>
x	read values of objects referred to by slot numbers in list	x <list_no>
!	write values to objects	! <list_no>
@	set up list of values to write	@<list_no><value1><status1>[<value2><status2>...]
?	inspect list of values	?<list_no>
+	create a shared variable	+ <slot>
++		++ <absolute slot>
-	delete a shared variable	- <slot>
--		-- <absolute slot>
e	open echo file	e <file name>
f	close echo file	f <file name>
*	establish repetition factor	* <integer>

Table 1-3. Command Summaries (Continued)

Command	Action	Syntax
y	describe error	y <error number>
z	wait <integer> seconds	z <integer>
h	get online help for specified command(s)	h <oma command> [<oma command>...]
hh	get all online help	hh
l	escape to UNIX shell level	l [UNIX command]
&	view SEQUENCE source file	& <slot>
q l qq	quit oma l quit oma and shells	qlqq
d	redraw menu	d
??	print file names for entire menu	??

Examples

1. Invoke oma with a file:

   ```
   $oma tstcase1.fpn
   ```

2. Using files:

   Add names to a menu.

   ```
   a 1 comp12.on b
   a 0 comp12:pid12.meas
   a 19 comp12:mon12_1.II0001 L
   a 3 just_a_name
   ```

   Delete a name.

   ```
   k 1
   ```

   Write menu to a file.

   ```
   w comp12.fpn
   ```

   After you execute these commands, the file **comp12.fpn** contains the following lines:

   ```
   a 0 comp12:pid12.meas F
   a 3 just_a_name F
   a 19 comp12:mon12_1.II00001 L
   ```

   Read names from files.

   ```
   r testcase1.fpn
   r testcase13.fpn
   ```

   The menu contains the names from **testcase13.fpn**.

   Assuming COMP:AIN is in manual mode.

3. To set the parameter in slot 3 with value 30, ACK bit on, BAD bit off, and OOS bit off:

   ```
   s 3 = 30  a~b~o
   ```

4. To set the packed-boolean parameter, COMP:AIN.INHALM, with mask 0x0103, value 0x0102, BAD bit off:
   ```
   s COMP:AIN.INHALM  p = 0x0103 0x0102 ~b
   ```
5. To set the packed-boolean parameter, COMP:AIN.INHALM, with OOS bit off:
   ```
   s COMP:AIN.INHALM p = * ~o
   ```
6. To set the packed-long parameter, COMP:AIN.INHALM, with mask 0xffffffff and value 0x01020000:
   ```
   s COMP:AIN.INHALM m = 0xffffffff 0x01020000
   ```
7. (m) to create a list (list #2) of three parameters for both (b) omread and omwrite,
   (o) to open the list, and
   (x) to perform the omread

Slot#	Parameter Name	Value Type
6	COMP:AIN.BAP	I (2-byte integer)
8	COMP:AIN.INHALM	P (2-byte packed-boolean)
10	COMP:AIN.KSCALE	F (4- byte float)

```
m 2 b 6 + 8 + 10
o 2
x 2
```

8. (@) to set up the set values for the list of parameters
   (?) to inspect the set values
   (!) to perform omwrite

COMP:AIN.BAP	with BAD bit off and OOS bit off
COMP:AIN.INHSTA	with value 0x00ff
COMP:AIN.KSCALE	with value 3.0

```
@ 2  * ~b~o 0x00ff | 3.0 |
? 2
! 2
```

9. (c) to close the list
   (i) to delete the list

```
c 2
i 2
```

omcrt

Use omcrt to create one or more OM objects of any object type (for example, *Variable*) and associated data types (for example, *Float*).

You can use omcrt in a System Start-up script to create OM objects at start-up. You can access omcrt from a UNIX or Windows shell level command environment using the format and parameters given below. To create more than one object of the same object type and data type with a single command, you can specify additional names on the command line.

NOTE

OM objects of different object types can have the same name. For example, you could create an OM process object named SYSMON and an OM variable object of data type *Float* (float variable) named SYSMON.

Format:

```
omcrt [-adpvs] [-bcfils | -pb | -pl | -pn portnum] [-n length] name
```

Parameters:

-adpvs	OM object type to create: -a alias -d device -p process -v variable (default if not data type of string) -s OM socket These objects are mutually exclusive.
-bcfils -pb -pl -pn	These options define the data type for object type variable (-v) and object type alias (-a) or the port number for object type OM socket (-s). For object type alias (-a), only data type string (-s) is allowed. For object type OM socket (-s), only data type portnum (-pn) is allowed. The data types for object type variable (-v) are: -b boolean -c character -f float -i integer -l long -s string -pb packed boolean -pl packed long The default type for variables is long (-l).

-n	This option can be used to specify the length of an object with data type string (-s). The default length for strings is 80. The minimum length is one and the maximum 255.
<name>	Name of one or more OM objects to be created.

Limitations:

You can only create multiple objects with a single invocation if the options are the same for all objects. If you need different types, use omcrt repeatedly. The maximum number of objects that can be created at one time is 30.

For more information, refer to the *Object Manager Calls* (B0193BC) document.

Examples:

- Create a process name DEVMON: omcrt -p DEVMON
- Create an OM float variable name TANK1: omcrt -v -f TANK1
- Create 2 OM integer variables named INTEGER1 and INTEGER2:
  omcrt -v -i INTEGER1 INTEGER2
- Create an OM alias named SYSPRINTER: omcrt -a SYSPRINTER

omdel

Use omdel to delete specified OM objects. To delete more than one object of the same object type at a time, you must specify additional names on the command line. You can access omdel from a UNIX or Windows shell level command environment using the format and parameters given below.

Format:

```
omdel [-adpvs] <name> [<name>...]
```

Parameters:

-adpvs	OM object type to delete: -a alias -d device -p process -v variable (default) -s OM socket The options are mutually exclusive.
<name>	name of one or more OM objects to be deleted.

Limitations:

You can only perform multiple deletes on a single invocation if the OM objects are the same object type. If you need to delete OM objects of different types, use omdel repeatedly. The maximum number of objects that can be deleted at one time is 30.

For more information, refer to the *Object Manager Calls* (B0193BC) document.

Examples:

- Delete a process name DEVMON: `omdel -p DEVMON`
- Delete an OM float variable name TANK1: `omdel -v TANK1`
- Delete 2 OM integer variables named INTEGER1 and INTEGER2:
  `omdel -v INTEGER1 INTEGER2`
- Delete an OM alias named SYSPRINTER: `omdel -a SYSPRINTER`

omfnd

Use omfnd to determine if a specified I/A Series object exists on an I/A Series system. You can access omfnd from a UNIX or Windows shell level command environment using the format and parameters given below. The result of the search is written to standard output.

Format:

```
omfnd [-adpvs] <name>...}
```

Parameters:

-adpvs	type of I/A Series object to find: -a OM alias -d OM device -p OM process -v OM, AO or CIO variable (default) -s OM socket The options are mutually exclusive.
<name>	name of one or more I/A Series objects to be found.

To find more than one I/A Series object of the same type with a single command, specify additional <names> on the command line.

Limitations:

You can only determine the existence of multiple objects using a single invocation if all the objects are the same type. If you need to determine the existence of different object types, use omfnd repeatedly. The maximum number of objects that can be specified on the command line is 30.

For more information, refer to the *Object Manager Calls* (B0193BC) document.

Examples:

- Find an OM process object name DEVMON: `omfnd -p DEVMON`
- Find an OM float variable name TANK1: `omfnd -v TANK1`
- Find a CIO object named RAMP:PID.OUT: `omfnd -v RAMP:PID.OUT`
- Find an Application Object named APP1:LEVEL.MEAS:
  `omfnd -v APP1:LEVEL.MEAS`

omget

Use omget to get the value, status and timestamp of one or more I/A Series objects and write them to standard output. You can access omget from a UNIX or Windows shell level command environment using the format and parameters given below.

Format:

```
omget [-av]  [-x] {<name>...}
```

Parameters:

-av	I/A Series object type to retrieve: -a OM alias -v OM, AO or CIO variable (default)
-x	Retrieve timestamp
<name>	name of one or more I/A Series objects to be retrieved.

Limitations:

You can only retrieve the values of objects of the same type with a single omget. If you need to retrieve the values of different object types, use omget repeatedly. The maximum number of objects that can be specified on a command line is 30.

For more information, refer to the *Object Manager Calls* (B0193BC) document.

Examples:

- Get the value and status of an OM float variable name TANK1: `omget  -v TANK1`
- Get the value, status, and timestamp of a CIO object named RAMP:PID.OUT:
  `omget -v -x RAMP:PID.OUT`
- Get the value and status of an Application Object named APP1:LEVEL.MEAS:
  `omget  APP1:LEVEL.MEAS`

omgetimp

The omgetimp command obtains the same information as the omget command, but omgetimp automatically uses the OM Import Table to save the station address of the object which will eliminate future multicast messages. Use omgetimp for consecutive calls to the same OM object or for consecutive calls to data with the same compound or application for CIO Objects and Application Objects respectively. The initial omgetimp for an I/A Series object will use a multicast message but all subsequent gets for the same I/A Series object will use the OM Import Table address entry to perform a direct send message to the station that sources the data. You can access omgetimp from a UNIX or Windows shell level command environment using the format and parameters given below.

Format:

```
omgetimp [-av]  [-x] {<name>...}
```

Parameters:

-av	I/A Series object type to retrieve: -a OM alias -v OM, AO or CIO variable (default)
-x	Retrieve timestamp
<name>	name of one or more I/A Series objects to be retrieved.

Limitations:

You can only retrieve the values of objects of the same type with a single omgetimp. If you need to retrieve the values of different object types, use omgetimp repeatedly. The maximum number of objects that can be specified on a command line is 30.

For more information, refer to the *Object Manager Calls* (B0193BC) document.

Examples:

- Import and get the value and status of an OM float variable name TANK1:
  `omgetimp -v TANK1`
- Import and get the value, status, and timestamp of a CIO object named RAMP:PID.OUT: `omgetimp -v -x RAMP:PID.OUT`
- Import and get the value and status of an Application Object named APP1:LEVEL.MEAS: `omgetimp APP1:LEVEL.MEAS`

omset

Use omset to set the value and/or status of one or more I/A Series objects. You can access omset from a UNIX or Windows shell level command environment using the format and parameters given below.

Format:

```
omset [-av] [-bcfils <value> -pb mask:hexvall-pl
mask:hexval] <name>
```

Parameters:

-av	I/A Series object type to set: -a OM alias -v OM, AO or CIO variable (default)
-bcfils -pb -pl	datatype of object(s) to be created. For OM object type alias (-a), only string (-s) is allowed. For OM, AO and CIO object type variable (-v), the following types are allowed: -b boolean -c character -f float -i integer

	-l long (default) -s string -pb packed boolean -p packed long When you set a string value, omset automatically adds surrounding single quotes. To set a boolean (-b), supply T for TRUE and any other value for FALSE. For -pb and -pl, value defaults to hex.
<name>	name of one or more I/A Series objects to be set
mask:hexval	Used only in combination with types -pb and -pl, this parameter defines the hex mask and hex value. If a mask value is not given the default value is 0xffff for -pb and 0xffffffff for -pl.

Limitations:

You can only set objects of the same object type to the same value at a time. If you need different types or different values, use omset repeatedly.

For more information, refer to the *Object Manager Calls* (B0193BC) document.

Examples:

- Set the value of an OM float variable name TANK1 to 77.54:
  `omset -v -f 77.54 TANK1`
- Set the value of an OM integer variable name INT12 to 1300:
  `omset -v -i 1300 INT12`
- Set the value of a CIO float variable object named RAMP:PID.OUT to 56.43:
  `omset  -f 56.43 RAMP:PID.OUT`

omsetimp

The omsetimp command obtains the same information as the omset command, but omsetimp automatically uses the OM Import Table to save the station address of the object which will eliminate future multicast messages. Use omsetimp for consecutive calls to the same OM object or for consecutive calls to data with the same compound or application for CIO objects and AO objects respectively. The initial omsetimp for an I/A Series object will use a multicast message but all subsequent sets for the same I/A Series object will use the OM Import Table address entry to perform a direct send message to the station that sources the data. You can access omsetimp from a UNIX or Windows shell level command environment using the format and parameters given below.

Format:

```
omsetimp [-av] [-bcfils <value> -pb mask:hexvall-pl
mask:hexval] <name>
```

Parameters:

-av	I/A Series object type: -a OM alias -v OM, AO or CIO variable (default)
-bcfils -pb -pl	datatype of the I/A Series object(s) to be set. For the OM object type alias (-a), only string (-s) is allowed. For OM, AO and CIO variable (-v), the following data types are allowed: -b boolean -c character -f float -i integer -l long (default) -s string -pb packed boolean -pl packed long When you set a string value, omsetup automatically adds surrounding single quotes. To set a boolean (-b), supply T for TRUE and any other value for FALSE. For -pb and -pl, value defaults to hex.
<name>	name of one or more I/A Series objects to be set
mask:hexval	Used only in combination with types -pb and -pl, this parameter defines the hex mask and hex value. If a mask value is not given the default value is 0xffff for -pb and 0xffffffff for -pl.

Limitations:

You can only set objects of the same type to the same value at a time. If you need different types or different values, use omsetimp repeatedly.

For more information, refer to the *Object Manager Calls* (B0193BC) document.

Examples:

- Import and set the value of an OM float variable name TANK1 to 77.54:
  `omsetimp -v -f 77.54 TANK1`
- Import and set the value of an OM integer variable name INT12 to 1300:
  `omsetimp -v -i 1300 INT12`
- Import and set the value of a CIO float variable object named RAMP:PID.OUT to 56.43: `omsetimp -f 56.43 RAMP:PID.OUT`

omary50

omary50 was designed as a tool to provide a user interface identical to the Pulp and Paper Industry tool omary. omary50 provides the following additional functions:

- ability to delete OM data array(s)
- ability to get profile plot statistical data
- ability to get and set long, short, and byte data
- ability to use packed data as input for the set options
- a special set option (-newset) to create data arrays without timing out
- data arrays may be anywhere on the network (not limited to local station)
- ability to update the ARYNAME time (-update option)

Format:

```
omary50 ARYNAME [-p -get -{new}set -delete -stats [s e]
[skip] -size - time -update -help]
<stdin/stdout>
```

Parameters:

ARYNAME	the Trigger Connection name (shared variable) or the OM Data Array name without the 01 suffix.
-p	pack/unpack flag For -get: does not unpack OM short/byte data array. For -[new]set: does not pack OM short/data array. This option lets you create your own packed data array, set it to omary50, and retrieve it without unpacking the data.
-get	get ARYNAME data and send output to stdout. The format of the stdout output is identical to the -set stdin input requirements.
-newset	create new OM data array and an OM shared variable (long) to be used for trigger connection and update time, and set the OM data array using input from stdin. This option is identical to the -set option except that it does not time-out on the data access.
-set	same as -newset, but also creates the OM shared variables when it fails to access the variables. The data set is FDR'ed and is compatible with the WP30 Display Manager. The stdin input must have the following format:

```
[DATA_TYPE]
hscale:<hihg_scale_value>
lscale:<low_scale_value>
halm:<high_alarm_value>
```

```
lalm:<low_alarm_value>
ref:<reference_value>
mlow:<pack_data_minimum_low_value>
cmult:<pack_data_multiply_value>
point_value_1
point_value_2
     .
     .
     .
last_point_value
```

where:

DATA_TPE	pt_float, pt_long, pt_short, pt_byte (default is pt_float)
mlow:	used for pt_short and pt_byte values to pack the data (bias
cmult:	used for pt_short and pt_byte values to pack the data (multiply factor)
point_value:	floating point format value (e.g., 565), or for NANs, it should be the string NAN.

The number of plot points is calculated by the number of point values defined (NANs included).

-delete delete the OM shared variables

-stats [s e] [skip] retrieves statistical data from the OM data array and outputs them to stdout:

High scale - the high scale value

Low scale - the low scale value

High alarm - the high alarm value

Low alarm - the low alarm value

Maximum - the maximum value of samples

Minimum - the minimum value of samples

Average - the average value of samples

Variance - the variance of prime-sigma

prime-sigma - the prime-sigma (deviation of average)

2-prime-sigma - 2 x prime-sigma

skip pnts - the skip point option (if not 0)

start pnt# - the starting point # (1 based) if not 1 end pnt# - the last point used (1 based) if less than array size

total plot pnts - total number of plot points in OM data arrays

#plot pnts calc - number of plot points in calculation (NAN's excluded and points skipped excluded)

option [s e]:s==starting point number (1 based, default==1)

e==ending point number (1 based, default is last point)

[skip]skip points (0 based value, default==0 in sample)

For more detail about statistical data (including algorithms), refer to *REAL-TIME Process Control* (author: Paul Badavas) Chapter 2.

-size	outputs the number of plot points stored into the OM data array
-time	outputs the last time the OM data has been updated (time retrieved from ARYNAME variable).
-update	updates ARYNAME variable with current time. This can be used to force updates on Display Manager screens.

Examples:

```
omary50 PAPER -newset < paper
```

Creates the necessary OM data arrays (PAPER01...PAPER<n>) and the OM long shared variable PAPER to hold the last time the array is updated. The scale and point data is read from the file paper.

```
omary50 PAPER -get
```

Gets the OM data array data and prints it to stdout.

```
omary50 PAPER -stats
```

Gets the statistical data of all the profile plot points stored in array PAPER01...PAPER<n>.

```
omary50 PAPER -stats 10 60
```

Gets the statistical data of point 10 through point 60, stored in array PAPER01...PAPER<n>.

```
omary50 PAPER -stats 1
```

Gets the statistical data of all points, skipping every other point (that is, points #1,3,5,7,...n).

```
omary50 PAPER -size
```

Gets the number of plot points stored in PAPER01...PAPER<n>.

```
omary50 PAPER -set < paper1
```

Updates data array scales and point values with data from file `paper1`.

rsom

The rsom utility transfers control to a command environment in which you can view the Object Manager subsystem databases in remote control stations. The rsom commands are for Foxboro® diagnostic purposes and should be used only under the direction of Foxboro Field Service. You can access rsom from a UNIX or Windows shell level command environment using the format and parameters given below.

Format:

```
rsom
```

Available commands are given below. Note that an underlined character is a valid abbreviation for the command.

Command	Argument	Function
FILE	##	File given screen
HELP or ?	command	Display help information
MORE Help	command	Display OM commands
MORE	command	Display more operational data
NEXT or NXT		Display next screen
QUIT		Terminate session
SCR	##	Display given screen
ADR		Display OM address table
CONN		Display OM Server connection table
DBID	open id	Display open list header
ID	open id	Display omopen table header
IMP		Display import entries
LIST		Display object directory entries
NADR	open id	Display network address table for optimized list
OBJ	object name	Display object value record for object directory entry
OPDB		Display local and remote open lists
OPDBX		Display lists and xdata information
OPNL		Display local lists for omopen table
OPVR	open id	Display list points for a specified open list
PQTBL		Display OM Process Queue Table
SCAN		Display OM Scanner database
SCONN		Display OM Scanner connection table
XTBL		Display xdata for local lists for omopen table

som

The som utility transfers control to a command environment in which you can view the Object Manager subsystem databases for the workstation where som is executing. The som commands are for Foxboro diagnostic purposes and should be used only under the direction of Foxboro Field Service.

You can access som from a UNIX or Windows shell level command environment using the format and parameters given below.

Format:

```
som
```

Available commands are given below. Note that an underlined character is a valid abbreviation for the command.

Command	Argument	Function
FILE	##	File given screen
HELP or ?	command	Display help information
MORE Help	command	Display OM commands
MORE	command	Display more operational data
NEXT or NXT		Display next screen
QUIT		Terminate session
SCR	##	Display given screen
ADR		Display OM address table
CONN		Display OM Server connection table
DBID	open id	Display open list header
ID	open id	Display omopen table header
IMP		Display import entries
LIST		Display object directory entries
NADR	open id	Display network address table for optimized list
OBJ	object name	Display object value record for object directory entry
OPDB		Display local and remote open lists
OPDBX		Display lists and xdata information
OPNL		Display local lists for omopen table
OPVR	open id	Display list points for a specified open list
PQTBL		Display OM Process Queue Table
SCAN		Display OM Scanner database
SCONN		Display OM Scanner connection table
XTBL		Display xdata for local lists for omopen table

2. System Management Utilities

This chapter covers System Management utilities, which help locate I/A Series stations and report on system monitors. They enable you to upload and reboot stations, check and report on Control Databases, and execute global searches for selected objects.

System Management utilities provide the functionality to locate I/A Series stations and report on system monitors. In addition, the utilities enable you to upload and reboot stations, check and report on Control Databases, and execute global searches for selected objects. The table below details command names and utility descriptions.

Command Name	Description
cpoint	checkpoint stations with Control databases
dbvu	report on checkpoint status (CP60 and earlier CPs)
dbvu280	report on checkpoint status (FCP280)
dbvu270	report on checkpoint status (FCP270 or ZCP270)
fist	locate stations on an I/A Series system
frev	report on system monitors
glof	global search for specified objects
iaboot	reboot local station
iaboot_upld	upload a station

cpoint

Use `cpoint` to checkpoint stations that contain Control Databases. If successful, `cpoint` logs a System Monitor message printed on the System Alarm printer, indicating the status of the checkpoint (“Checkpoint Success” or “Checkpoint Failed”).

— NOTE

Unlike the checkpoint facilities in System Management and the Integrated Control Configurator, `cpoint` does not check whether the control database is being changed while the checkpoint is performed.

To ensure useful results, space the checkpoints several minutes apart. If you invoke `cpoint` by means of a script, add `sleep 300` as a five-minute delay between checkpoints. This ensures that the Integrated Control Configurator does not update the database you are checkpointing. Scheduling using `cron` is not recommended unless you can guarantee that the Integrated Control Configurator does not update the database that you are checkpointing.

You can access `cpoint` from a UNIX or Windows shell level command environment using the format and parameters given below.

Format:

```
cpoint <lbug> <host>
```

Parameters:

<lbug> - letterbug of the control station to checkpoint

<host> - AP host of the control station to checkpoint

— NOTE

If lbug is invalid and host is valid, cpoint does not log a System Monitor message.

dbvu

dbvu is the CP60 or earlier (or GW, PW, or ACM) checkpoint file inspection tool.

— NOTE

For CP270 and later control processors, you must use “dbvu270” on page 26 instead of this command.

You can access dbvu from a UNIX or Windows shell level command environment.

The -t, -d, and -e options (in addition to the CP checkpoint file) require matching versions of the CP map file and CP image file (same release, unpacked or uncompressed) to provide precise definitions of all control blocks and their parameters.

In the case of the Foxguard (ACM) checkpoint file, a special VRTX16 version of map and image DBVUACM.map and DBVUACM (instead of the Motorola 68040/VRTX32 version of map and image actually running in the station), are provided, solely for the usage of the dbvu -[tde] command.

For a large CP checkpoint file, the -t option requires a large amount of disk space for the output file. You may choose to pipe the output to a user-written script to filter out what is needed from the output.

Format:

```
dbvu    [-spblru] -C<CP_checkpoint_file>
dbvu    -t -C<CP_checkpoint_file> -M<CP_map_file> -I<CP_image_file>
          [<compound_name> [<block_name>]]
dbvu    -d -C<CP_checkpoint_file> -M<CP_map_file> -I<CP_image_file
dbvu    -e -C<CP_checkpoint_file> -M<CP_map_file> -I<CP_image_file
```

Parameters:

-C<CP_checkpoint_file>	CP checkpoint file name
-M<CP_map_file>	CP map file name
-I<CP_image_file	CP image file name

-s	report on period-phase
-p	report on compound-block-period-phase
-b	report on blocks-with-bad-input-parameter-data-type
-l	report on blocks-with-secured-linkage
-r	report on blocks-with-remote-linkage
-u	report on blocks-with-local-unresolved-linkage
-t	report on compound-block-parameters (not available for CP270 or later processors)
-e	report on compound-block-errors (not available for CP270 or later processors)
-d	report on station-parameters relating to CP loading, alarm devices, and supervisory control (not available for CP270 or later processors)
(no option)	report on strings, linkages, periods, and phases when only -C<CP_checkpoint_file> is present

Examples (CP270 or later CPs):

```
dbvu            -CDBC32CP3.UC    > /tmp/t
dbvu    -spblru -CDBC32CP3.UC    > /tmp/t
```

Examples (CPs Previous to CP270):

```
dbvu            -CDBC32CP3.UC    > /tmp/t
dbvu    -spblru -CDBC32CP3.UC    > /tmp/t
dbvu    -t      -CDBC42CP4.UC   -MOS1C40.map    -IOS1C40 C42CP4_STA
                                                STATION
dbvu    -t      -CDBC32CP3.UC   -MOS1C30.map    -IOS1C30      > /tmp/t
dbvu    -t      -CDBTRICN1.UC   -MDBVUACM.map   -IDBVUACM     > /tmp/t
dbvu     -d     -CDBC12CP1.UC   -MOS1UC.mp2     -IOS1UC
dbvu     -d     -CDBC12DP1.UC   -MOS1DCT.mp2    -IOS1DCT
dbvu     -d     -CDBC12TP1.UC   -MOS1HTG.mp2    -IOS1HTG
dbvu     -e     -C/usr/fox/sp/files/DBPCAT00.UC -M/usr/preserve/vrtx.map -I/vrtx
                (inspecting a PW checkpoint file on a PW platform)
```

— NOTE

For more information on using regular expressions such as those listed above, refer to Appendix B “Regular Expressions - Quick Reference Guide” on page 51.

dbvu280

dbvu280 is the FCP280 checkpoint file inspection tool. It provides the functionality that dbvu is unable to support for the FCP280.

NOTE

For the FCP270/ZCP270, use "dbvu270" on page 26 instead of this command. For CP60 and earlier control processors, use "dbvu" on page 22 instead of this command.

You can access dbvu280 from a Windows command prompt environment only. After opening the Windows command prompt, navigate to the folder D:\opt\fox\bin\tools prior to executing the command.

Format (FCP280):

```
dbvu280    [-spteblrudT] -L <CP LETTERBUG>
-OR-
dbvu280    [-spteblrudT] -D <CP_checkpoint_file>
```

NOTE

Spaces required for proper operation between all -L and -D options.

NOTE

Both -L and -D options cannot be used together.

Parameters:

-L <CP LETTERBUG>	FCP280 letterbug
-D <CP_checkpoint_file>	FCP280 checkpoint file name

-s	report on period-phase
-p	report on compound-block-period-phase
-t	report on all compound-block-parameters (also, see text below)
-e	report on all compound-block-errors
-b	report on blocks-with-bad-input-parameter-data-type
-l	report on blocks-with-secured-linkage
-r	report on blocks-with-remote-linkage
-u	report on blocks-with-local-unresolved-linkage

-d	report on station-parameters relating to FCP280 loading, alarm devices, and supervisory control
-T	print the string pool/table

— NOTE

Executing dbvu280 with no options [-spteblrudT] supplied results in the string table and a linkage report being printed.

For example, select the FCP280's default checkpoint files to examine by typing the following command:

```
dbvu280     -p    -L A1CP81
```

Select an FCP280's checkpoint file which is different from the default by typing the following command:

```
dbvu280     -p    -D D:\usr\fox\sp\files\BBA1CP81.UC
```

The above filename syntax assumes that the command is executed from the standard Windows command prompt shell. If executed from within the NuTCRACKER shell (sh), then the file syntax would be as follows: /usr/fox/sp/files/BBA1CP81.UC.

— NOTE

dbvu280 must be executed from `D:\opt\fox\bin\tools`.

Examples (FCP280):

Executing the following command will result in the printing of the string table and a linkage report for the station name A1CP81.

```
dbvu280     -s    -L A1CP81
```

When using the -t option, you may specify a regular expression to match blocks against.
Executing the following command will only print the station block parameters found in the file BBA1CP81.UC:

```
dbvu280     -tSTATION          -D D:\usr\fox\sp\files\BBA1CP81.UC
```

If your regular expression becomes complex or includes spaces, it is recommended to enclose the argument in quotes. For example, the following command will print the parameters for the station block and any block or compound with ECB in its name:

```
dbvu280     -tSTATIONIECB                    -L A1CP81
```

It is also important that the regular expression follows the -t without any spaces in between. For example, "dbvu280 -tSTATION" is good but "dbvu280 -t STATION" will result in undefined behavior.

— NOTE —

For more information on using regular expressions such as those listed above, refer to Appendix B "Regular Expressions - Quick Reference Guide" on page 51.

dbvu270

`dbvu270` is the FCP270/ZCP270 checkpoint file inspection tool. It provides the functionality that `dbvu` is unable to support for the CP270s.

— NOTE —

For the FCP280, use "dbvu280" on page 24 instead of this command.
For CP60 and earlier control processors, you must use "dbvu" on page 22 instead of this command.

You can access `dbvu270` from a Windows shell level command environment only. It is not supported from a NuTCRACKER shell, or in any Unix/Solaris/SPARC command environments.

MapOffsets.txt and BlockTypeMap.txt File Requirements

Before using `dbvu270`, you must create a `MapOffsets.txt` file which will contain the parameter offsets - the byte offsets in the CP image file (OS1C70 or OS1Z70) where the block parameter definitions are found - required by the inspection tool.

A script called `mkblkma.sh`, supplied with I/A Series software v8.6 or later, creates this file. This script must be run from within a NuTCRACKER shell. It requires a map file for the current version of the CP image and checkpoint file you are working with to generate the `MapOffsets.txt` file. The available map files are shipped with I/A Series software and are in the directory /usr/fox/sp/files, typically named FCP270.MAP and ZCP270.MAP.

The `mkblkma.sh` script will only produce a `MapOffsets.txt` file for one map file, and must be run multiple times to create the appropriate `MapOffsets.txt` file for each map file. It is recommended that you rename each `MapOffsets.txt` file to indicate its source map file and prevent overwriting.

To execute the script and create a `MapOffsets.txt` file, proceed as follows:

1. Open a command prompt, such as **`Start`** -> **`Run...`**, type **`cmd`** and press <**`Enter`**>.
2. Type the following in the command prompt window:

   `D:` and press <**`Enter`**>

   `ncenv` and press <**`Enter`**>

   `sh` and press <**`Enter`**>.

   `cd /opt/fox/bin/tools` and press <**`Enter`**>.
3. Execute the `mkblkma.sh` script. Here is an example:

   ```
   ./mkblkma.sh /usr/fox/sp/files/FCP270.MAP
   ```

The `MapOffsets.txt` file is created in your current working directory, in this case, `D:\opt\fox\bin\tools`. This file may be renamed so future executions of the script will not overwrite existing files. For example, the file generated by the FCP270.MAP file can be renamed to `FCPMapOffsets.txt` to indicate its source.

As well, to use dbvu270, the BlockTypeMap.txt file must be in the current working directory; otherwise, dbvu270 will fail. The file defines the available block types for dbvu270. The BlockTypeMap.txt file should never be edited.

When you have finished renaming the MapOffsets.txt file and confirmed the location of the BlockTypeMap.txt file, dbvu270 can be used to examine checkpoint files.

Format (CP270s):

```
dbvu270    [-spteblrudT] -I<CP_image_file> -O<MapOffsets.txt_file>
               -D<CP_checkpoint_file> [-F or -Z]
```

NOTE

Spaces required for proper operation between all -I, -O and -D options.

Parameters:

-I<CP_image_file>	CP image file name
-O<MapOffsets.txt_file>	Name of the MapOffsets.txt file
-D<CP_checkpoint_file>	CP checkpoint file name
-F[1]	Indicates FCP270 generated the checkpoint file OR
-Z[1]	Indicates ZCP270 generated the checkpoint file

1. It is required to specify either -F or -Z.

-s	report on period-phase
-p	report on compound-block-period-phase
-t	report on all compound-block-parameters (also, see text below)
-e	report on all compound-block-errors
-b	report on blocks-with-bad-input-parameter-data-type
-l	report on blocks-with-secured-linkage
-r	report on blocks-with-remote-linkage
-u	report on blocks-with-local-unresolved-linkage
-d	report on station-parameters relating to CP loading, alarm devices, and supervisory control
-T	print the string pool/table

Example (CP270s):

```
dbvu270   -s   -I D:\usr\fox\sp\files\OS1C70    -O FCPMapOffsets.txt    -D D:\usr\fox\sp\files\DBA1FCP1.UC    -F
```

Executing the command above will result in the printing of the string table and a linkage report.

When using the -t option, you may specify a regular expression to match blocks against. For example, executing the following command will only print the station block parameters:

```
dbvu270   -tSTATION   -I D:\usr\fox\sp\files\OS1C70   -O FCPMapOffsets.txt   -D D:\usr\fox\sp\files\DBA1FCP1.UC   -F
```

If your regular expression becomes complex or includes spaces, it is recommended to enclose the argument in quotes. For example, the following command will print the parameters for the station block and any block or compound with ECB in its name:

```
dbvu270   "-tSTATION|ECB"   -I D:\usr\fox\sp\files\OS1C70   -O FCPMapOffsets.txt   -D D:\usr\fox\sp\files\DBA1FCP1.UC   -F
```

It is also important that the regular expression follows the -t without any spaces in between. For example, "dbvu270 -tSTATION" is good but "dbvu270 -t STATION" will result in undefined behavior.

— NOTE

For more information on using regular expressions such as those listed above, refer to Appendix B "Regular Expressions - Quick Reference Guide" on page 51.

fist

Use fist to locate stations on an I/A Series network (I/A Series Nodebus, I/A Series control network or The Mesh control network). For each station you specify on the command line, fist prints an identifier and a MAC (ethernet) address. The identifier is a composite, consisting of the following sub-identifiers:

- site identifier
- LAN identifier
- nodebus identifier

If fist cannot find a station or discovers any other error, it issues a report identifying the error by number. You can access fist from a UNIX or Windows shell level command environment using the format and parameters given below.

Format:

```
fist <station_letterbug>[<station_letterbug>...]
```

Parameters:

<station_letterbug> the letterbug of a station.

Output:

The fist utility produces a listing with the station identifier, address, and error description in tabular form, as shown below:

Station	Network Site Identifier	Address (hex)	Error Description
CP0001	I000102	00006CC00106	
CP0004	I000103	00006CC0011A	
YYYYYY			Not found (-1)

frev

Use frev to find the firmware revision number of every station and every ECB with an EEPROM from all system monitors. The frev output reports the time and date at which you execute the utility, the workstation on which you execute it, system monitor names, system monitor host workstation or PWs, and a table of data about each station.

— NOTE

To execute frev, you must have a workstation that is configured to support a system monitor.

Since frev places a heavy load on the workstation, run it only when the system can tolerate the additional load.

Format:

```
frev [<SYSTEM_MONITOR_DOMAIN>]
```

Parameters:

<SYSTEM_MONITOR_DOMAIN> uppercase name of a System Monitor

Output:

For each station, frev outputs the following information:

- station letterbug
- L1 ECB name
- L2 ECB name
- fault tolerant flag
- revision number for primary
- revision number for shadow
- status

The following is an excerpt of a frev report:

```
EEPROM Revision number report:
Thu Nov 19 11:48:21 1992
run on SUNW66
```

```
                                                SYSMN2-->AP100B
Station    L1 ECB    L2 ECB    FT          Primary     Shadow      Status
AP100B                                     2.14
CM100B                                     2.18
UCE001
           UCE001                          10.11
```

Notice the following reporting conventions:

- Station names appear in alphabetical order within each system monitor domain.
- If a level 1 ECB has any level 2 ECBs (as do intelligent field devices), the level 2 ECBs appear in the column labeled ECB2.
- For fault tolerant stations, the FT field contains an FT indication; otherwise it is blank.
- The firmware revision number appears in the column labeled Primary. For fault tolerant stations, this column contains the firmware revision number of the primary module.
- The Shadow column contains the firmware revision number of the shadow module of a fault tolerant station. This field is blank for stations that are not fault tolerant.
- The status field is blank except in case of an error, in which case, it contains an error indication.

Errors:

If frev cannot find a system monitor, it prints the message “not found” next to the system monitor’s name at the top of its table. It still prints the level one and level two ECB information and lists all the system monitor’s stations along with their fault tolerant information. The primary and shadow field remains blank.

Files:

Files needed by frev:

```
/usr/fox/sysmgm/smonlist.cfg
/usr/fox/sysmgm/dom_<sysmonitor>.cfg
/usr/fox/sp/hldb
```

Error codes are contained in:

```
/usr/include/fox/ipc.h
```

For I/A Series software v8.x or later (which do not include `ipc.h`), these error codes are listed in Appendix A “IPC Error Codes” on page 47.

glof

Use glof to find the MAC address of the station containing an I/A Series object. The output contains the Object Manager completion code and the PSAP address of the station. You can access glof from a UNIX or Windows shell level command environment using the format and parameters given below.

Format:

glof [option]<object>[<object>...]

Parameters:

[option]	If not supplied, the <object> is interpreted as a letterbug. Option may also be: -a OM alias -p OM process -v OM, AO or CIO variable
<object>	Object to be found.

Output:

Output from glof gives the PSAP address of the station using 26 hex bytes in the following order: two bytes for ssap_id, two bytes for tsap_id, two bytes for nsap_len, and 20 bytes for NSAP address.

The NSAP address contains, in order, one byte for alternate format indicator (AFI), two bytes for site id, two bytes for LAN id, two bytes for node id, and six bytes of station id (MAC address). The last seven bytes are unused.

The Object Manager completion code for a successful find is 0; for a name not found, the code is -1.

Error codes for glof are located in `/usr/include/fox/om_ecode.h`.

NOTE

glof does not work properly on an AP51.

iaboot

The iaboot utility is used to reboot stations. You can access iaboot from a UNIX or Windows shell level command environment using the format and parameters given below.

Format:

iaboot <letterbug>

Parameters:

<letterbug> Letterbug of station to reboot

Errors:

The only explicit output from iaboot is caused by an IPC error.

Error codes are in `/usr/include/fox/ipc.h`

For I/A Series systems v8.x or later (which do not include `ipc.h`), these error codes are listed in Appendix A “IPC Error Codes” on page 47.

NOTE

iaboot cannot be used in conjunction with an ACM while it is running redundant. If there is an iaboot while running ACM redundant, only the Primary is rebooted.

iaboot_upld

The iaboot_upld utility has different behaviors, depending on whether it is used with I/A Series software earlier than v8.x, or with I/A Series software v8.x or later.

With I/A Series Software Pre-V8.x

Use iaboot_upld to upload a station. The station goes off line during upload processing and then automatically reboots. You can access iaboot_upld from a UNIX or Windows shell level command environment using the format and parameters given below.

Format:

```
iaboot_upld <station>
```

Parameters:

<station> Letterbug of station to upload.

Errors:

When iaboot_upld fails to communicate to the host via IPC, the following messages display. The results of the upload are logged to the system printer with either of the following two messages.

"Equipment failed"

"Memory Dump Successful;. File name =..." or "Memory Dump Failed. File name =..."

Memory dumps are located in `/usr/fox/sysmgm/softmgr/dump` directory.

IPC error codes are in `/usr/include/fox/ipc.h`.

For I/A Series systems v8.x or later (which do not include `ipc.h`), these error codes are listed in Appendix A "IPC Error Codes" on page 47.

With I/A Series Software V8.x or Later

The use of the iaboot_upld utility with I/A Series software v8.x or later varies, depending on whether it is uploading an FCP280, FCP270, or ZCP270, or legacy station image.

For Uploading Legacy (CP60 or Earlier) Station Image

You can use iaboot_upld to upload a legacy station (CP60 or earlier) hosted by a workstation with I/A Series software v8.2 or later over the ATS. The station goes off line during upload processing and then automatically reboots. You can access iaboot_upld from a UNIX or Windows shell level command environment using the format and parameters given below.

Format:

iaboot_upld <station> U

Parameters:

<station> Letterbug of station to upload.

U Upload dump image from single/primary module.

Errors:

When iaboot_upld fails to communicate to the host via IPC, the following messages display. The results of the upload are logged to the system printer with either of the following two messages.

"Equipment failed"

"Memory Dump Successful;. File name =..." or "Memory Dump Failed. File name =..."

Memory dumps are located in `/usr/fox/sysmgm/softmgr/dump` directory.

IPC error codes are in `/usr/include/fox/ipc.h`.

For I/A Series software v8.x or later (which do not include `ipc.h`), these error codes are listed in Appendix A "IPC Error Codes" on page 47.

For Uploading FCP280/FCP270/ZCP270 Image

To use the iaboot_upld utility to upload an FCP280, FCP270, or ZCP270 image, refer to the section "Memory Dumps" in the following documents, depending on the Control Processor you are using:

- *Field Control Processor 280 (FCP280) User's Guide* (B0700FW)
- *Z-Module Control Processor 270 (ZCP270) User's Guide* (B0700AN)
- *Field Control Processor 270 (FCP270) User's Guide* (B0700AR)

3. Miscellaneous Utilities

This chapter describes bpatch, getpars, sipc, and other miscellaneous utilities available in the software utilities associated with the UNIX and Windows workstation operating environments.

bpatch

Use bpatch to view and modify binary files.

NOTE

Before using bpatch make a backup copy of the file.

When you issue a bpatch command at the shell level, control is transferred to a command environment. The command environment displays file contents and a command line on which you issue directives. If you are using a Wyse terminal that is directly connected to the 50 Series, you must use the following keys to move through the file:

`control-h`	move left
`control-j`	move down
`control-k`	move up
`control-l`	move right

To use the backspace key to end an edit, make sure that your terminal is set with the backspace key as DEL/CAN (using VT100 setup).

To access online help for bpatch, type ? on the command line. You can access bpatch from a UNIX or Windows shell level command environment using the format and parameters given below.

Format:

```
bpatch <filename>
a = asci
h = hexadecimal
```

getpars

Use getpars to collect information about compound variables from a control station. You can access getpars from a UNIX or Windows shell level command environment using the format and parameters given below.

Format:

```
getpars [options.]
```

Options:

-U<CP letterbug wildcard filter>

If -U is not supplied, the default is *.

-u<CP letterbug> name of CP letterbug to be added to the list. A maximum of 40 -u options can be specified.

-C<compound_name wildcard filter>>

If -C is not specified, the default is *.

-p <compound parameter name>:<print format>

special parameter names:

CP for ZCP letterbug

format specifiers:

[-]n.m[f|g|e]	float
[-]n[.m]s	string
[-] [0]c	character
[-] [0]n[1]x	hexadecimal
[-] [0]n[1]o	octal
[-] [0]n[1]d	decimal

[1] is for 4-byte integer (e.g., ALMSTA, BLKSTA)

<value>?, <value>*, or NA is displayed if the parameter is out of service, bad, or not applicable to the compound or block, respectively.
A maximum of 20 -p options may be specified.

-B <block name wildcard filter>

If -B is not specified, the default is *.

-t <block type name>

-m <block parameter name> :print format>

You can use the special parameter names:

CP for CP letterbug

CMPNM for compound name

For print format specifiers, see -p, above.
A maximum of 40 -m options can be specified.

-n If -n is not specified, one header is printed per new CP in a compound report and one header per new compound in a block report. You can print CP and CMPNM per output line to facilitate further ASCII data processing on the output file by other UNIX tools, such as grep and awk.

If -n is specified, one parameter header is printed at the start. You can print CP and CMPNM per output line to facilitate further ASCII data processing upon the output file (i.e., grep and awk).

-f<option filename>

Each line in the file has the following:
<option character without leading '-'>
<space or tab> <option string>
<space or tab> <format string without leading ':'> (p,m only)
An f or h option is ignored in an option file.

-h Print help information.

sipc

The sipc utility transfers control to a command environment. You can view the Invensys Foxboro Inter-Process Communication (IPC) subsystem software database on the workstation where sipc is executing.

CAUTION
Use the sipc utility only under the direction of Foxboro Field Service.

You can access sipc from a UNIX or Windows shell level command environment using the format and parameters given below.

Format:

```
sipc
```

Available commands are listed below. Notice that an underlined character is a valid abbreviation.

Command	Argument	Function
FILE	##	File given screen
HELP or ?	command	Display help information
MORE	command	Display more operational data
NEXT or NXT		Display next screen
PREV or PRV		Display previous screen
QUIT		Terminate session
SCR	##	Display given screen
CDT	name	Display CDT_PRT entry
DT	name	Display DT_PRT entry
EVENT or EVN	name	Display EVN entry
LIST or LST	CDT/DT	Display CDT/DT names
TABLE or TAB	command	Display a table
USAGE or USG		Display table usage
BYTE	command	Changes display mode to BYTE_MODE
WORD	command	Changes display mode to WORD_MODE

rsipc

The rsipc utility transfers control to a command environment. You can view the Foxboro Inter-Process Communication (IPC) subsystem software database on a remote control station.

CAUTION

Use the rsipc utility only under the direction of Foxboro Field Service.

You can access rsipc from a UNIX or Windows shell level command environment using the format and parameters given below.

Format:

```
rsipc
```

Available commands are listed below. Notice that an underlined character is a valid abbreviation.

Command	Argument	Function
FILE	##	File given screen
HELP or ?	command	Display help information
MORE	command	Display more operational data
NEXT or NXT		Display next screen
PREV or PRV		Display previous screen
QUIT		Terminate session
SCR	##	Display given screen
CDT	name	Display CDT_PRT entry
DT	name	Display DT_PRT entry
EVENT or EVN	name	Display EVN entry
LIST or LST	CDT/DT	Display CDT/DT names
TABLE or TAB	command	Display a table
USAGE or USG		Display table usage
BYTE	command	Changes display mode to BYTE_MODE
WORD	command	Changes display mode to WORD_MODE
ASM	command address (hex)	Disassemble instructions
OSMAP	command	Display OS memory layout
XAM	command address (hex)	Displays requested Station memory

4. Legacy Historian Utilities

This chapter covers the legacy historian utilities and unavailable programs on the AP50 platform.

These utilities are for use in an I/A Series system with pre-V8.x software. The AIM*Historian is available for V8.x or later software versions, discussed in the *AIM*Historian User's Guide* (B0193YL) document.

cfgpts

Use cfgpts to extract the configuration of collection points from an existing database into a pure ASCII file. You can also take pure ASCII input and place it into a database. This tool is handy for bulk configuration and configuration backup. Refer to the instructions in the document cfgpts.doc.

dmpcfg

dmpcfg lists parts of a Historian configuration in readable format. It must be called with one or more of the option letters not separated by spaces.

Usage: dmpcfg [c] [p] [r] [o] [m] [g]

Option:

c	List collection points (data from “all_points” and “tnd_memb”).
Pkey	Unique point number, never reused.
Point Name	Alphabetically sorted. For display reasons, the point names are truncated to 30 characters.
Description	The point description is truncated to 15 characters. If the description is longer than 15 characters, an exclamation mark is printed after it.
Delta	Dead Band.
Indx	Index into _tdata and SAM file number (1-500_).
Rate	Sample Rate.
nm	New_member flag.
ext	Number of records configured for extended sampling.

NOTE

If the point has no entry in tnd_member (such as when there is a deleted collection point, or a point is mistakenly configured as a reduction member only) the text “Not a collection point” replaces the data from tnd_memb.

Option:

p	Implies option (c). In addition, one line is printed for each group to which the point is a member.
Description	See above
Dit	Not used, always 1.0
Owner Group	Owner group name, reduction or sample archive group.

Option:

r	List reduction group configuration data.
Group Name	Reduction Group name.
U	Usage flag, 2 = multiple.
C	Configuration Status.
S.Per	Sample Period.
R.Per	Reduction Period.
RSpan	Data Retention Span.
(RPP)	Number of Records per point, RPP = RSpan/R.PER + 1.
DeLay	Phase delay.
MSpan	Minimum input scan required to produce OK data.
ISpan	Input Span, used for cascade groups only. Normally same as Reduction Period.
VB	Size of buffers for intermediate values.

Option:

o	Implies option (r). In addition, all operations defined for the group are listed.
Operations	Operation Type and Operation Name.
Source	Source operation name, for cascade groups only.
L0_Lim	Histogram Low Limit.

Hi_Lim	Histogram High Limit, not used for other operations.

Option:

g	Implies option (r). In addition, all member groups of cascaded reduction groups are listed.
Member Group	Member Group name.
(Span)	Not used, always 15m.

Option:

m	Implies option (r). In addition, all member points of non-cascaded reduction groups are listed.
Member Points	Unique Point number, (not the _tdata index!), and the point name.
Description	Point description, truncated at 15 characters, with a warning exclamation mark for longer descriptions.

— NOTE

The greatest verbosity is obtained with all options or “dmpcfg pogm”. It is useful to list reduction groups with these options sets: “dmpcfg r” and “dmpcfg ogm”.

dmpnam

dmpnam reads all records from both “_mdata” and “_tdir”, and normally prints the entire contents of the _mdata records including the following:

- Point name
- Point description (truncated after the first 15 characters)
- Change delta
- Status word
- Point number (= index) = SAM file number
- Sample collection rate
- New-member flag
- Maximum number of data records in the SAM file (100 sample records)

A warning is printed with each record, if:

- The name in _tdir does not match the name in _mdata
- The _mdata entry id missing for an existing _tdir entry
- The _tdir entry is missing for an existing _mdata entry

dmpstat

The new 50 Series version of "dmpstat" analyses the entire SAM files and shared memory. The old version only checked the samples in _tdata. This program prints an overview of the SAM file and/or shared memory contents. For each point with a non-zero point ID, it prints the following:

#	= the point index (0-histsize)
N	= the number of samples with a non-zero time stamp
Span	= the time span between the oldest and most recent samples
TSLU	= the Time Since Last Update (relative to the current system time)
ID	= the point name
Min	= the minimum sample value found in the record
Max	= the maximum sample value found in the record
RAOSBCT	= the individual fields in the sample status words (an 'x' is printed if the status field is not the same for all samples)

hist_stat

hist_stat reports several useful things about the local AP50 Historian including the following:

- The configured name of the local Historian
- The configured size of the local Historian
- The required size of the virtual shared memory
- The directory path of the sample data files (SAM files)
- A detailed status report of the Historian's shared memory segment, including its key number, segment ID, virtual address, and items returned by shcmctl()
- The open-points list identifiers returned by omopen() calls
- The values of the shared status variables for sampling, reduction, archiving, and message collection
- Active Historians on the network that responds to a "dbname" IPC broadcast

hopchk

hopchk performs an omread() on all OM list that were last opened by the Historian. It prints a summary of the number of points that have an OM status of zero or 0x80. A zero status indicates that the point is not scanned by the OM. This is typical for deleted points that were not replaced by others. A status of 0x80 occurs when the omread() has not modified the local buffer. This is typical of closed lists, for example, when the Historian is OFF.

The report for each OM list is subdivided into groups of 23 points. This is the way that the connect messages get sent to start-up. An entire block of 23 points with a 0 status may indicate unsuccessful omopen() at start-up. This condition has been seen with overloaded CPs especially CP10s.

hopspy

hopspy prints the OM open points list IDs by the Historians. It then prompts for one of the IDs and prints the current value of the 50 points on the list.

The OM IDs are obtained from the file "_listid", and the actual samples are received with an omread() call. The most common return codes from omread() are as follows:

0	= OK
-4	= list not open (e.g. Historian is OFF)
-5	= list only partially configured (typically the last list or any list with a deleted point)

hsv_spy

hsv_spy prints the current value of the "Historian Statistical Variables" (HSV) at twenty second intervals. Use 'Ctrl-C' to stop this function.

The HSV is a set of six shared variables that are created in hist_init as part of the reboot sequence. Their names are: H1_<dbname> through <dbname>, where <dbname> is the historian name in all upper case. Like other I/A shared variables, the HSV can be trended or collected by a Historian.

NOTE

Do **not** use these shared variables for plant management or other business purposes. They are intended for software development and performance testing. Their usage may be changed without notice at any time in the future.

Current usage of the Historian Statistical Variables is as follows:

H1_<dbname>	Number of samples received form the OM dqchange() call. Standardized to "samples per minute".
H2_<dbname>	Highest number of samples received from dqchange() during any 2-second scan cycle in the last twenty seconds.
H3_<dbname>	Number of SAM file updates performed. Standardized to "updates per minute".
H4_<dbname>	Highest number of SAM files updates performed during any 2-second scan cycle in the last twenty seconds.
H5_<dbname>	Elapsed time spent in the sampling_ctl process collecting samples. Standardized to "ticks per minute". One tick equals 1/60 of a second. Derived from the return code of the times() function.
H6_<dbname>	Longest duration (elapsed time in ticks) of any one scan of sampling_ctl in the last twenty seconds. Derived from the return code of the times() function.

htest

Use htest to test most of the Historian Library functions in libhist.a. The functions are described in the *Historian* document. `htest` prompts for the Historian name, the function and the function arguments. It calls the function and prints the results.

infospy

This program prints the contents of the agrou.info file, which is a list of SAM files in an archive database that was generated by an extended sample archive group. The SAM file number, the point name, and the oldest and most recent time stamps are printed for each record.

The current working directory must be that archive directory.
For example "`/opt/fox/hstorian/archive/htlrg2a22f.dbs`".

mdew1

Use this utility as an alternate method of entering MDE data from a terminal or from an ASCII file.

mytime

This utility converts a UNIX style (long int) time value to ASCII.

Usage: for example: mytime 759450000.

redinfo

Use redinfo to examine the Historian shared variables and to send GET_STAT messages to reduction_ctl. You must run this utility on a local AP, as it picks up the database name from `/etc/histin`.

redinfo first prints the value of the Historian control variables. Note that get_val times out at about ten seconds on a non-existent variable. Thereafter it prompts for a reduction group name (type -1 to quit). The internal status block for that group is retrieved and printed, including some unused variable.

.stat	= Request status, 0 = ok, -32 = ENOT_DEF = group does not exist
gstat.state	= Reduction group status, e.g. 3 = SCHEDULED
gstat.schd_tim	= Time of next collection for a non cascaded group. Time of next reduction for cascaded group.
gstat.period	= Reduction period in seconds
gstat.delay	= Phase delay

samspy

Use samspy to examine sample data in the local AP50 database. The data is read from shared memory, current SAM files, and archived or played-back SAM files. The data is displayed one screen at a time. samspy can also find the point number for a given point name or match partial point names with names in _tdir (see UNIX function strstr(3)). The 50 Series version of samspy is not sensitive to the current working directory.

The prompts are intuitive. They show possible answers in (parenthesis), and the current default answers in [brackets].

- Pressing **RETURN** selects the default
- Pressing **+** steps to the next number
- Pressing **-** or **_** steps to the previous number

For data interpretation, refer to SDS 5010.

tapespy

Use tapespy to display the contents of a Historian archive tape in selectable detail. If run without arguments, you are prompted to specify the level of detail desired in the output. A prompt requesting that you press **RETURN** displays for each new page of output.

Alternately the verbosity level may be defined as a command line argument. In this mode, all prompts are omitted. This is useful when the output is redirected to a file.

Usage: tapespy [-C]
where 'C' may be one of the option letters:

a	Print only one line per archive containing: - a sequential archive number - the letter <s> or <r> to indicate sample or reduction type archive - the archive database name - the time and date the archive was backed up on tape - the size of the archive in blocks - the accumulated size of the archives "sum=..."
h	Print each archive header in more detail
f	Print the size and name of each individual file in the archives
d	Print the return code of each I/O call (debug mode).

NOTE

In the output, I/O messages start with "==>", and the debug messages with the I/O function return code start with " -- ".
All sizes are in blocks. The block size is printed near the beginning of the output (BUFSIZ=).
The size of the directory file ".sizes" (size_size) is not included in the archive size (arch_size), but both are included in the accumulated archive size "sum=..".
The condition "==> tape I/O error, read()=-1, error=5." is typically used to detect the end of data on the tape, but can also indicate other problems. On recycled tapes, end of data is typically indicated by a missing banner and reported as: "Stale data found, end of Historian archives."

testhd

Use testhd to simulate data retrieval functions. The communication between Historians can be verified as well as the proper operation of the data collection processes of a remote or local Historian.

Guided by prompts, you can build data retrieval request messages and send them to hs_fetch or hr_fetch of any Historian on the network. The contents of the response messages are printed on the terminal.

Source level knowledge is required to build and interpret the messages.

Programs not Available on the AP50 Platform

dmpidx	Replaced by samspy
dmpstat	Not ported yet to Historian 50
dmpten	Replaced by samspy
watchdog	Considered obsolete on both platforms

Appendix A. IPC Error Codes

This appendix lists the IPC error codes formerly held in the obsolete ipc.h file.

The following tables provide the IPC error codes for any of the packages which once included `/usr/include/fox/ipc.h` (which is found only in I/A Series systems with software with revisions earlier than v8.x).

NOTE

Symbolic references for IPC error codes error codes up to -999 are reserved for GM-MAP standard error codes. Error codes less than -999 are Invensys generated.

Table A-1. llc Error Codes

Error Variable	Value	Definition
E_INV_CHAN	-1	invalid channel id
E_NO_ACT	-2	no c_activate
E_NO_CHANS	-3	no channels available
E_TIME_OUT	-4	time out
E_MAX_SIZE	-5	maximum buffer size exceeded
E_NO_SERV	-7	service not available
E_INV_ACT	-9	no such action ever happened before
E_INV_NAM	-11	bad station or process name
E_NOT_REG	-12	process not registered
E_INV_UE	-13	UE name not known
E_DISCON	-15	“disconnect” command has been issued
E_ILL_EFN	-16	invalid event flag number
E_TWO_DIS	-18	second disconnect
E_DISC_IND	-19	disconnect received
E_MULT_RCV	-20	second receive call outstanding
E_ANS_OUT	-21	second answer call on same channel
E_NOT_CHK	-22	didn't check last operation
E_NO_DEST	-23	connect destination not found
E_UE_DEST	-24	destination UE does not exist
E_UE_UNACK	-27	UE name unacceptable
E_ANS_ABORT	-28	answer aborted
E_ABORT_CON	-29	abortion of connect request
E_ILL_EOM	-30	EOM out of range
E_ILL_ACT	-31	invalid action

Table A-1. llc Error Codes (Continued)

Error Variable	Value	Definition
E_MULT_ACT	-40	second c_activate
E_ILL_MASK	-41	invalid mask

Table A-2. Invensys Generated Error Codes

Error Variable	Value	Definition
IN_PROGRESS	0	in progress
FOUND	1	object found
CALL_COMPLETE	1	call has been successful
NOT_FOUND	-1	object not found
CHAN_PENDING	-1	operation is pending
E_UE_EXISTS	-1000	UE exists
E_INV_BUFPTR	-1001	invalid buffer pointer
E_NO_AL_ENT	-1002	no alias entry
E_ALS_NOT_CHK	-1003	didn’t check last operation on an alias
E_ONGO_IPC	-1004	ongoing IPC transactions
E_GROUPID	-1005	invalid groupid
E_INV_INFOPTR	-1006	invalid info pointer
E_INV_ANAME	-1007	invalid alias
E_INV_DNAME	-1008	invalid d_name
E_MAX_CHAN	-1010	illegal value for maximum channels
E_NO_RES	-1012	out of resources
E_INV_MID	-1013	invalid message id
E_SERVICE	-1021	illegal value for service
E_MSG_REJECTED	-1041	message rejected
CONN_PENDING	-1050	pending connection
E_ILL_AL_VAL	-1065	user exceeded max number of aliases
E_ILL_NAM_SZ	-1066	illegal size for name
E_NO_DT_ALIAS	-1069	no dt alias
E_INV_MY_NAME	-1070	invalid my_name given in info structure
E_ILL_LOCA	-1071	letterbug not found
E_INV_SEC_CODE	-1080	illegal security code
E_INV_SEG_SEL	-1081	illegal segment selection
E_INV_TIME	-1082	illegal timer value
E_INV_TIMEID	-1083	illegal timer service id
E_NO_XLATE	-1084	could not translate user message

Table A-3. Error Codes For User's VENIX Library Interface

Error Variable	Value	Definition
E_FILE_ACCESS_ERR	-2000	IPC device file access bad
E_INV_FILEDES	-2001	invalid file descriptor

Table A-4. Error Codes For VENIX Signals

Error Variable	Value	Definition
E_SIGNAL	-2002	signal occurred
E_SIG_ABORT	-2003	abort signal from VENIX

Appendix B. Regular Expressions - Quick Reference Guide

This appendix provides a quick reference guide for using regular expressions (regex).

Regular Expressions - Quick Reference Guide

Anchors

^	start of line
$	end of line
\b	word boundary
\B	not at word boundary
\A	start of subject
\G	first match in subject
\z	end of subject
\Z	end of subject or before newline at end

Non-printing characters

\a	alarm (BEL, hex 07)
\cx	"control-x"
\e	escape (hex 1B)
\f	formfeed (hex 0C)
\n	newline (hex 0A)
\r	carriage return (hex OD)
\t	tab (hex 09)
\ddd	octal code ddd
\xhh	hex code hh
\x{hhh..}	hex code hhh..

Generic character types

\d	decimal digit
\D	not a decimal digit
\s	whitespace character
\S	not a whitespace char
\w	"word" character
\W	"non-word" character

POSIX character classes

alnum	letters and digits
alpha	letters
ascii	character codes 0-127
blank	space or tab only
cntrl	control characters
digit	decimal digits
graph	printing chars -space
lower	lower case letters
print	printing chars +space
punct	printing chars -alnum
space	white space
upper	upper case letters
word	"word" characters
xdigit	hexadecimal digits

Literal Characters

Letters and digits match exactly	a x B 7 0
Some special characters match exactly	@ - = %
Escape other specials with backslash	\. \\ \$ \[

Character Groups

Almost any character (usually not newline)	.
Lists and ranges of characters	[▨]
Any character except those listed	[^▨]

Counts (add ? for non-greedy)

0 or more ("perhaps some")	▨*
0 or 1 ("perhaps a")	▨?
1 or more ("some")	▨+
Between "n" and "m" of	▨{n,m}
Exactly "n", "n" or more	▨{n}, ▨{n,}

Alternation

Either/or	▨|▨

Lookahead and Lookbehind

Followed by	▨(?=▨)
NOT followed by	▨(?!▨)
Following	(?<=▨)▨
NOT following	(?<!▨)▨

Grouping

For capture and counts	(▨)
Non-capturing	(?:▨)
Named captures	(?<name>▨)

Back references

Numbered	\n \gn \g{n}
Relative	\g{-n}
Named	\k<name>

Character group contents

x	individual chars
x-y	character range
[:class:]	posix char class
[^:class:]	negated class

Examples

[a-zA-Z0-9_]
[[:alnum:]_]

Comments

(?#comment)

Conditional subpatterns

(?(condition)yes-pattern)
(?(condition)yes|no-pattern)

Recursive patterns

(?n)	Numbered
(?0) (?R)	Entire regex
(?&name)	Named

Replacements

$n	reference capture

Case foldings

\u	upper case next char
\U	upper case following
\l	lower case next char
\L	lower case following
\E	end case folding

Conditional insertions

(?n:insertion)
(?n:insertion:otherwise)

http://www.e-texteditor.com

Index