Command Line Interface Supplemental

Rev#: 1.0

29th April 2025

- *This document is provided to the recipient subject to and in accordance with the terms of non-Disclosure agreement.*
- *Provision of Technical Documentation and Samples Standard Terms and Conditions; and/or*
- *Component OEM Agreement signed by recipient and Memjet.*

TO THE EXTENT PERMITTED BY LAW, THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING BUT NOT LIMITED TO, IMPLIED WARRANTIES OF ACCURACY, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF ANY THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

Memjet will not be liable for any misstatements or omissions, including but not limited to misstatements or omissions in relation to instructions for the operation, use or maintenance of any equipment, samples, components or software. Recipient acknowledges and agrees that whilst care has been taken in compiling this information, it may contain estimates and draft information, and may not be current, accurate or complete. The information contained herein is subject to change without notice.

In the event of any conflict between the terms of this document and any executed agreement between recipient and Memjet, the terms in such executed agreement shall control and prevail. For the sake of clarity, only warranties expressly made in executed agreements will be binding on recipient and Memjet. Nothing herein should be construed as constituting an additional warranty

Revision History

Doc Version.	Date	Details	Ref
1.0	29th April 2025	Initial Release	

Table of Contents

Figures

Tables

1 Introduction & purpose

- To provide a quick reference guide of useful command lines for the reader.
- Some are particularly useful when triaging.
- The assumption is that the reader has familiarity with using Linux, SSH and/or PUTTY terminals.
- This supporting information does not replace the Officially released DuraFlex documentation readily available.
- Commands listed within are compatible with the Live Release build versions listed in Table 1
- The table below also provides the sub package version IDs within the targeted Release build.

Table 1 - Flex software release revision information.

Live release version	Gymea	Kareela	Dynamo / Kirrawee	Glenbeigh	PDL	Ultron
R5.3.3	MJ5.2.4-16	MJ5.3.3-14	MJ5.12.2-3	MJ4.2.0-6	MJ6.5.0-2	0.8RC85
R5.2.2	MJ4.13.8-27	MJ5.2.1-4	MJ5.8.1-2	MJ4.2.0-6	MJ6.5.0-2	0.8RC85
R5.0.2	MJ4.9.3-8	MJ5.0.2-10	MJ5.7.0-1	MJ4.2.0-6	MJ6.3.1-2	0.8RC85

2 Frontend mode - PES interface

2.1 Overview

Most common operational tasks are accessible via the PES interface. This is accessible by using the SSH application within the CMD prompt or direct via PUTTY terminal. There are 2 modes available:

- Frontend: This mode will only accept PES client commands
- Combined: This mode uses commands from both PES client frontend and backend. See section #3 for further information.

2.2 Accessing the operational command line/interface:

2.2.1 Remote SSH to target Ross board

Example: *ssh duraflex@192.168.100.200*

The IP address used in this example may differ from the IP address used in the target system.

- Type the following: *cd /opt/memjet/PDL/test_rigs/latest/bin/*
- Type the following: *python start.py --mode=frontend*
  Note: For combined mode steps and function see section **#3**.

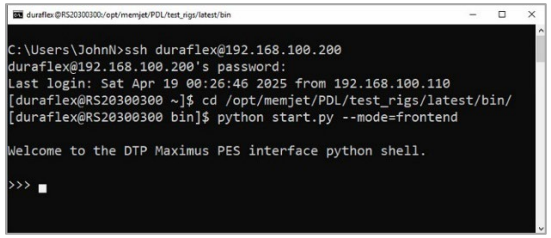


Figure 1 - Example screen shot of using SSH terminal

- To use the pes commands, start string with pes. followed by tab, tab (twice):
  NOTE: frontend will not accept backend commands

```
>>> pes.
pes.POSITIONS                   pes._abc_cache                  pes.pes_host
pes.__abstractmethods__         pes._abc_negative_cache         pes.prepareToPrint(
pes.__class__(                  pes._abc_negative_cache_version pes.replaceWipers(
pes.__delattr__(                pes._abc_registry               pes.sequencer
pes.__dict__                    pes.abortEventSession(          pes.setJobAllowNext(
pes.__doc__                     pes.circulateInk(               pes.shutDownEngine(
pes.__format__(                 pes.clearJobQueue(              pes.start(
pes.__getattribute__(           pes.delegate                    pes.startAlgorithm(
pes.__hash__(                   pes.drainInkFromPrinthead(      pes.startDepriming(
pes.__init__(                   pes.engine_state                pes.startMovingPrintheads(
pes.__metaclass__(              pes.error_handler(              pes.startPriming(
pes.__module__                  pes.event_client                pes.startPrinting(
pes.__new__(                    pes.finishPrinting(             pes.startServicing(
pes.__reduce__(                 pes.generateJobId(              pes.start_client(
pes.__reduce_ex__(              pes.getSettings(                pes.start_event_client(
pes.__repr__(                   pes.getStatus(                  pes.stop(
pes.__setattr__(                pes.indexWipers(                pes.stop_client(
pes.__sizeof__(                 pes.initialiseEngine(           pes.stop_event_client(
pes.__str__(                    pes.log                         pes.storeSettings(
pes.__subclasshook__(           pes.pausePrinting(
pes.__weakref__                 pes.pes_client
>>> pes.
```

Figure 2 - Example screen shot showing available pes commands

2.2.2 PES command references

Most of the useful PES commands can be found in the following documentation (SharePoint Access required):

- DuraFlex Operations Guide
- DuraFlex Installation & Commissioning Guide
- DuraFlex Trouble Shooting Guide

2.3 High level PES commands

2.3.1 Commonly used high level PES commands

Table 2 – Commonly used high level PES client commands.

Description	Input String	Description
Cap the Printhead	startMovingPrintheads()	• Printhead should be in capped position when not in use to prevent nozzles dehydrating.
Light Service	startServicing([], ServiceType.LIGHT)	• Perform at the beginning and end of the shift to make sure nozzles are clean. • If any printhead health issues, perform light service to restore nozzle health.
Medium Service	startServicing([], ServiceType.MEDIUM)	• Perform if a light service is not successful in recovering minor Printhead print quality defects. • Medium service uses a higher suction vacuum than light service.
Heavy Service	startServicing([], ServiceType.HEAVY)	• A heavy service consumes significant waste ink and time and is not needed during normal operations. • Only perform this rarely and only if successive light and medium services do not recover print quality. • **CAUTION:** Never perform more than two heavy services back-to-back, or the WIMM tank may overflow.

		• If the OEM system has a custom WIMM drain system, the default minimum time can be modified via the *hwparamstore.json* file.
Replace Wiper	replaceWipers()	• Moves the wiper carriage to the loading/unloading position • Resets the current wiper and indexing counts
Prime	startPriming()	• Circulates ink from the ink reservoir to the printhead. • Initialize the printing system before priming to avoid the valve being left in an unknown position.
Deprime	startDepriming()	• Drains ink from the printhead to the ink reservoir. • Initialize the printing system before depriming to avoid the valve being left in an unknown position.
Remove the Printhead	startMovingPrintheads()	• Use this to move the printhead into position for replacement. • Additional syntax will need to be used to define which position the user wants to move to.
Initializes the print engine	initialiseEngine()	• Use this to transition from an OFF state • Ensures print engine is ready for next operational stage
Shuts down the print engine	shutDownEngine()	• Transitions the print engine to an OFF state • Used as part of error recovery • Used whenever the engine is to be shut down.

3 Combined mode – PES & Backend

3.1 Overview

Most components (Switches, sensors, motors, etc.) can be checked and driven by logging into the system via combined mode. PES commands can also be used in this mode along with backend commands (See Table 2).
In some situations, particularly when triaging, you may require access to the mechanical control board. Please refer to section #4 for this information if the combined mode command is not listed.

Note:

- **ALL commands are case sensitive.**
- **The Commands listed in this section are to be used during triage only. They are NOT to be used in any other instance**

CAUTION

Great care and attention MUST be taken when using these commands. Failure to take appropriate precautions may result in ink leaks OR damage to the print head.

Using these commands are at the risk of the Field Engineer using them. Memjet does NOT hold any responsibility for any damage that can occur from misuse.

3.2 Accessing the command line/interface

1. Open a CMD Prompt

Figure 3 – MS CMD Prompt icon.

2. Type the following string: *ssh root@<host name or IP address>*
   Example: *ssh duraflex@192.168.100.200*
3. Type the following string: *cd /opt/memjet/PDL/test_rigs/latest/bin/*
4. Type the following string: *sudo systemctl stop delegation*
5. Type the following string: *python start.py --mode=combined*

NOTE: When finished with combined or backend modes, **ensure** you restart the delegation service by typing the following string:

- *sudo systemctl start delegation*

```
[root@RS20300300 bin]# sudo systemctl stop delegation
[root@RS20300300 bin]# python start.py --mode=combined
```

Figure 4 – Stopping delegation & entering combined mode

3.3 Combined mode low level commands.

3.3.1 Status checks: PRINT MODULE SWITCHES

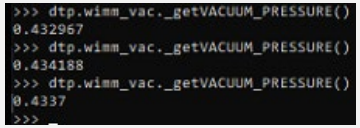
Table 3 – Command list for system switches.

Description	Input String	Example Response	Comments
WIPER PRESENT switch	dtp.ss.wiper.SENSOR_VAL_PRESENT	'active'	• Active = wiper cartridge is present/installed/switch closed. • Inactive = wiper cartridge is absent/missing/switch open
WIPER LOW switch	dtp.ss.wiper.SENSOR_VAL_LOW	'inactive'	• Active = wiper cartridge is LOW/OUT. • Inactive = wiper cartridge is OK
WIPER HOME switch	dtp.ss.wiper.SENSOR_VAL_HOME	'inactive'	• Inactive = wiper in HOME check position • Active =Wiper not in HOME check position
PH CAPPED switch	dtp.ss.lifter.SENSOR_VAL_CAP	'inactive'	• Active = Capper NOT in Cap position • Inactive = Capper in capped position

CAPPER HOME POSITON switch	dtp.ss.cap.SENSOR_VAL_HOME	'active'	• Active = Capper is out in capped position • Inactive = Capper parked in the home position
CAPPER OUT POSITION switch	dtp.ss.cap.SENSOR_VAL_OUT	'active'	• Active = Capper in Capped Position • Inactive = Capper in HOME position
PH in PRINT POSITION switch	dtp.ss.lifter.SENSOR_VAL_PRINT	'active'	• Active = Print head NOT in PRINT position • Inactive = Print head in print position
LIFTER HOME switch	dtp.ss.lifter.SENSOR_VAL_HOME	'inactive'	• Inactive = Cradle in a raised position • Active = Cradle either in a capped OR print position

3.3.2 Status checks: SENSORS

Table 4 – Command list for system sensors.

Description	Input String	Example Response	Comments
Check vac pressure value at the sensor	dtp.wimm_vac._getVACUUM_PRESSURE() **OR** dtp.ss.wimm_vac.VACUUM_PRESSURE	0.4337 	• Must be in combined mode • Can observe vac changes when the system is performing an action that requires the use of the VAC pump (i.e. medium service). • Command repeat is needed to capture the next response (Up arrow and enter key will work)

Check BIDS Ink float switches	dtp.ids.read_ink_level()	[<DtpInkLevelStatus.FULL: 'full'>,
		<DtpInkLevelStatus.FULL: 'full'>,
		<DtpInkLevelStatus.FULL: 'full'>,
		<DtpInkLevelStatus.FULL: 'full'>]
			• Will check all float switches in a single command.
			• Order of colours = **K,C,M,Y**
			• States are as followed:
			  ○ OVERFLOW = Ink level high
			  ○ FULL (Nominal) = Lower float switch high
			  ○ LOW = Ink level low

3.3.3 Component driving: PINCH VALVE

Table 5 – Command list for driving the Pinch Valve.

Description	Input String	Example Response	Comments
Check the Valve driver state	dtp.ids.valve._getSTATE	'idle'	• Idle = present and ready to move • Timeout = not responding (Dynamo not responding OR issue surrounding the pinch valve electrical connection)
Check the current position of the Pinch valve	dtp.ids.POSITION_NAME()	'closed'	• The response is dependent on driver state and physical position. • If an unknown position is displayed, check physical electrical connectivity and/or driver status.

Move pinch valve to INK position	dtp.ids.valve.go_ink()	'ink'	• Allows ink to flow • Typically, in this position during operational states (Printing, flush, circulate, etc…) • May cause drooling if ink present in lines
Move pinch valve to AIR position	dtp.ids.valve.go_air()	'air'	• Closes ink lines pre valve • Typically used when de-priming • May cause drooling if ink present in lines
Move pinch valve to CLOSED position	dtp.ids.valve.go_closed()	'closed'	• Closes INK INLET & AIR lines to the print head • Typically used when shut down • Some operational algorithms require this in a closed position

3.3.4 Component driving: INK/WASTE PUMPS

Table 6 – Command list for driving Bulk Ink supply/Waste Output pump.

Description	Input String	Example Response	Comments
Stop Ink fill pump	dtp.ids.stop_refill_all()	No Response displayed. Carefully monitor the ink fill pumps.	• Stops ALL active ink fill pumps.
Starts ALL Ink fill pumps	dtp.ids.start_refill_all()	No Response displayed. Carefully monitor the ink fill pumps.	• Starts ALL ink fill pumps. • **Function relies on current ink level. If the ink level is nominal, the pump will not drive (Avoids OVERFILL)**

3.3.5 Component driving: CIRCULATION PUMPS

WARNING:

- For testing purposes ONLY. Do not drive 100% or exceed 100%
- Driving these pumps whilst the pinch valve is closed (See **Table 5**) will likely cause ink to drool out of the print head nozzles.

Table 7 – Command list for driving the Circulation Pumps

Description	Input String	Example Response	Comments
Drive the circulation pump assembly (BOTH pumps) at the same target rate	dtp.ids.circ_pump.start(<VALUE>)	No Response displayed.	• Drives BOTH pumps simultaneously. • <VALUE> = driving value in %. Example: dtp.ids.circ_pump.start(25) • **Ensure appropriate precautions have been met BEFORE driving and Ink drooling will likely occur**
To Stop driving the circulation pumps.	dtp.ids.circ_pump.stop()	No Response displayed.	• Stops BOTH pumps simultaneously.

3.3.6 Component driving: VALVES

Table 8 – Command list for driving system valve/s

Description	Input String	Example Responses	Comments
CHECK the status of the Capper drain valve	dtp.ss.cap._drain_valve.state()	<UltronBinaryActuatorStatus.IDLE: 1> <UltronBinaryActuatorStatus.BUSY: 2>	• Expected response is dependent on driver state and physical position. • The valve is in a NC state (Normally Closed) when zero volts is supplied. • NC state is expected when the cap is not draining.
OPEN the Capper drain valve	dtp.ss.cap._drain_valve.start()	No Response displayed. Just a slight delay whilst the valve opens	• An audible click will be heard when the valve opens • **Ensure the vales is not left on for prolonged periods of time**
CLOSE the Capper drain valve	dtp.ss.cap._drain_valve.stop()	No Response displayed. Just a slight delay whilst the valve opens	• An audible click will be heard when the valve closes

3.3.7 Component driving: LIFTER MOTOR

Table 9 – Command list for driving the Lifter Motor

Description	Input String	Example Response	Comments
Moves the Cradle into HOME (Raised) position	dtp.ss.go_home()	No Response displayed. Just a delay whilst the carriage moves	• The capper will retract back to its home position • **Ensure the movement area is clear of obstruction**
Moves cradle into CAPPED position	dtp.ss.go_cap()	No Response displayed. Just a delay whilst the carriage moves	• The capper will extend to its capping position • **Ensure the movement area is clear of obstruction**
Moves cradle into PRINT position	dtp.ss.go_print_position()	No Response displayed. Just a delay whilst the carriage moves	• The capper will retract back to its home position • The Cradle will move down fully into print position • **Ensure the movement area is clear of obstruction**

3.3.8 Component driving: MAINTENANCE MODULE SERVO MOTOR

WARNING:

- Ensure the movement area is clear of obstruction
- Ensure the LIFTER is in a raised position BEFORE driving
- Ensure the CAPPER is in its home (retracted) position BEFORE driving
- Ensure the WIPER is in the HOME position BEFORE extending the capper OR moving the LIFTER

Table 10 – Command list for checking & driving Maintenance Module Servo Motor

Description	Input String	Example Response	Comments
Check the current position of the Wiper cartridge	dtp.ss.wiper.position	0.0	• The example is an expected response indicates the wiper is in its parked position.

Move the wiper carriage into the home (parked) position	dtp.ss.wiper.go_home()	No Response displayed. Just a delay whilst the wiper carriage moves	• The wiper will retract back to its home/parked position until the home position switch changes state (see **Table 5** for further information). • Servo position when checked, is expected to be 0.0 • **Ensure the movement area is clear of obstruction**
Move the wiper carriage into the wiper indexing position	dtp.ss.wiper.go_index()	No Response displayed. Just a delay whilst the wiper carriage moves	• The wiper carriage will move into the position typical for indexing the wiper cartridge. • Servo position when checked, is expected to be -18.0 • **Ensure the movement area is clear of obstruction**
Move the wiper carriage into the wiper cartridge replacement position indexing position	dtp.ss.wiper.go_replace()	No Response displayed. Just a delay whilst the wiper carriage moves	• The wiper carriage will move into the position typical for replacing the wiper cartridge. • Servo position when checked, is expected to be -467.0 • **Ensure the movement area is clear of obstruction**
Move the wiper carriage into the starting position of any service wipe	dtp.ss.wiper.go_wipe_start()	No Response displayed. Just a delay whilst the wiper carriage moves	• The wiper carriage will move into the position typical when a service wipe begins. • Servo position when checked, is expected to be -431.987488 • **Ensure the movement area is clear of obstruction**
Move the wiper carriage into the end position of any service wipe	dtp.ss.wiper.go_wipe_end()	No Response displayed. Just a delay whilst the wiper carriage moves	• The wiper carriage will move into the position typical when a service wipe begins. • Servo position when checked, is expected to be -83.0 • **Ensure the movement area is clear of obstruction**

3.3.9 Component driving: CAPPER MODULE SERVO MOTOR

WARNING:

- Ensure the movement area is clear of obstruction
- Ensure the LIFTER is in a raised position BEFORE driving
- Ensure the CAPPER is in its home (retracted) position BEFORE driving
- Ensure the WIPER is in the HOME position BEFORE extending the capper OR moving the LIFTER

Table 11 – Command list for driving Capper Module Servo Motor

Description	Input String	Example Response	Comments
Move capper out to Capped position	dtp.ss.cap.go_cap()	No Response displayed. Just a delay whilst the capper moves	• Ensure the movement area is clear of obstruction • Ensure the Lifter is in a raised position BEFORE driving
Move capper out to home (retracted) position	dtp.ss.cap.go_home()	No Response displayed. Just a delay whilst the capper moves	• Ensure the movement area is clear of obstruction • Ensure the Lifter is in a raised position BEFORE driving

3.3.10 Component driving: Waste Ink Management Module

Table 12 – Command list for Waste Ink Maintenance Module

Description	Input String	Example Response	Comments
Check the current state of the WiMM pump	dtp.wimm_vac.ACTUATOR_STATE	'off'	• Checks state of the Ultron pump output port. • Does not rely on feedback from the pump but will show if the output command is still active. • Useful if triage remotely.
Toggle Vac pump state between ON & OFF	dtp.wimm_vac.start() dtp.wimm_vac.stop()	No Response displayed.	• Enables Ultron output ON (+24v) or OFF (0v) • Check sensor value as shown in Table 4 • **Ensure the pump is not left on for prolonged periods of time**

4 Mechanical Control (Kirrawee/Ultron)

4.1 Overview

All mechanical components (Switches, sensors, motors, etc.) can be driven independently by logging into the Ultron board directly. In some situations, particularly when triaging, you may may require direct access to the mechanical control board.
This section covers access and some useful command strings may may aid, during triage.

Note:

- **ALL commands are case sensitive and require the user to manually input each string.**
- **Commands can be used during triage only. They are NOT to be used in any other instance**

CAUTION

Great care and attention MUST be taken when using these commands. Failure to take appropriate precautions may result in ink leaks OR damage to the print head.

Using these commands are at the risk of the Field Engineer using them. Memjet does NOT hold any responsibility for any damage that can occur from misuse

4.2 Accessing the command line/interface

1. SSH to target Ross board
   - Use either **SSH** OR **PUTTY** terminal
   - Example: *ssh root@192.168.100.200* OR *RS20300300.local*
2. Check to ensure you have internal communication with the Ultron Board

Table 13 – Ultron Board Comms Check

Description	Input String	Example Response	Comments
Basic Ultron PCB ping check	ping -c 1 ultron	PING ultron (192.168.3.2) 56(84) bytes of data. 64 bytes from ultron (192.168.3.2): icmp_seq=1 ttl=255 time=29.6 ms --- ultron ping statistics --- 1 packets transmitted, 1 received, 0% packet loss, time 0ms rtt min/avg/max/mdev = 29.626/29.626/29.626/0.000 ms	• Example of a Good/OK response • No need to use ncat for this command

3. Type following command followed by the return key: *nc localhost 9200*
   - This will move the cursor to the next line
   - All commands at this stage are CASE sensitive and must be individually typed
4. Use the tables defined below to target required areas.

4.3 Low Level Kirrawee/Ultron commands

4.3.1 High Level Commands

Table 14 – High level commands

Description	Input String	Example Response	Comments
Enable trace level logging for the Kirrawee/Ultron	set "log_level","trace"	0:"trace"	• Expected response if successful. • This will now show all command and responses to the respective component rather than the list of drivers loaded. • It will revert to off upon power cycle.
Check loaded firmware version	Dyn-ultron:get "fw_version"	0:"0.8.RC85"	• Example of an expected response. • Displays current installed version
Check the LAGGING headboard connectivity	Dyn-ultron:Ph-LAGGING:get "headboard_type"	0:"Meander"	• Example of a Good/OK response • An error would suggest an i2c cable issue
Check the LEADING headboard connectivity	Dyn-ultron:Ph-LEADING:get "headboard_type"	0:"Meander"	• Example of a Good/OK response • An error would suggest an i2c cable issue
List Servo drivers loaded in Ultron	Dyn-ultron:list "servo"	0:"LIFTER","CAP","WIPER","PUMP","PUMPTWO","VALVE"	• Example of an expected response

			• Servos relate to the pump and other motors • Useful syntax information when driving separately
List binary actuator drivers loaded in Ultron	Dyn-ultron:list "binaryActuator"	0:"CAP_SOLENOID","AIR_VENT_PUMP","AEROSOL_FAN"	• Example of an expected response • Displayed list relate to the Capper solenoid valve, air vent pump and aerosol fan components • Useful syntax information when driving separately
List sensor drivers loaded in Ultron	Dyn-ultron:list "sensor"	0:"VAC_PRES_SENSOR","INK_TEMP_SENSOR"	• Example of an expected response • Displayed list relate to the Vac pressure sensor (wimm), and the ink temperature sensor components. • Useful syntax information when driving separately
List system switch driver loaded in ultron.	Dyn-ultron:list "flag"	0:"LIFT_HOME","LIFT_CAP","LIFT_PRINT","CAP_HOME","CAP_OUT","WIPER_HOME","WIPER_PRESENT","WIPER_LOW","VALVE_LIM_A","VALVE_LIM_B"	• Example of an expected response • Displayed list is intuitive. • Lists suitable syntax of all switches available. • Useful syntax information when triaging suspected switch faults. • Similar to Table-3
Check thee Ink Fill/waste pump is driver is loaded and provide drive settings	Dyn-ultron:list "BIDS_FILL_K"	0:"regulator","BIDS_FILL_K","ink_fill_k",52.599998,"bids_lo_k",11171.102539,60000.000000	• Example of an expected response • Displayed list relate to the Vac pressure sensor (wimm), and the ink • Useful syntax information when driving separately.
	Dyn-ultron:list "BIDS_FILL_C"	0:"regulator","BIDS_FILL_C","ink_fill_c",52.599998,"bids_lo_c",11171.102539,60000.000000	
	Dyn-ultron:list "BIDS_FILL_M"	0:"regulator","BIDS_FILL_M","ink_fill_m",52.599998,"bids_lo_m",11171.102539,60000.000000	
	Dyn-ultron:list "BIDS_FILL_Y"	0:"regulator","BIDS_FILL_Y","ink_fill_y",52.599998,"bids_lo_y",11171.102539,60000.000000	
Call the status of ULTRON and ALL respective attributes	status	See Table 28 in **APPENDIX.B** for an expected output	• Example in APPENDIX.B, is a unit in a PRIMED_IDE and CAPPED state.

4.3.2 Status checks: PRINT MODULE SWITCHES

Table 15 – Low Level Print Module Switch states

Description	Input String	Example Response	Comments
WIPER PRESENT switch	Dyn-ultron:WIPER_PRESENT:get "value"	0:"active"	• Active = wiper cartridge is present/installed/switch closed. • Inactive = wiper cartridge is absent/missing/switch open
WIPER LOW switch	Dyn-ultron:WIPER_LOW:get "value"	0:"inactive"	• Active = wiper cartridge is LOW/OUT. • Inactive = wiper cartridge is OK
WIPER HOME switch	Dyn-ultron:WIPER_HOME:get "value"	0:"active"	• Inactive = wiper in HOME check position • Active =Wiper not in HOME check position
PH CAPPED switch	Dyn-ultron:LIFT_CAP:get "value"	'inactive'	• Active = Capper NOT in Cap position • Inactive = Capper in capped position
CAPPER HOME POSITON switch	Dyn-ultron:CAP_HOME:get "value"	0:"active"	• Active = Capper is out in capped position • Inactive = Capper parked in the home position
CAPPER OUT POSITION switch	Dyn-ultron:CAP_OUT:get "value"	0:"active"	• Active = Capper in Capped Position • Inactive = Capper in HOME position
PH in PRINT POSITION switch	Dyn-ultron:LIFT_PRINT:get "value"	0:"active"	• Active = Print head NOT in PRINT position • Inactive = Print head in print position
LIFTER HOME switch	Dyn-ultron:LIFT_HOME:get "value"	0:"active"	• Inactive = Cradle in a raised position • Active = Cradle either in a capped OR print position

4.3.3 Status checks: INK LEVEL SWITCH

Table 16 – Low Level Ink Level Switch states

Description	Input String	Example Response	Comments
Check BIDS/IDS Ink Tank **HIGH** Float Switch	Dyn-ultron:BIDS_HI_<colour>:get "value"	0:"inactive"	• Example shown is expected under NORMAL operational conditions. • During NORMAL operations, if active, the system will generate an INK_FULL error. • <colour> = Target BIDS/IDS colour (C,M,Y,K)
Check BIDS/IDS Ink Tank **LOW** Float Switch	Dyn-ultron:BIDS_LO_<colour>:get "value"	0:"active"	• Example shown is expected under NORMAL operational conditions. • During NORMAL operations, if inactive, the system will activate the ink supply pump to pull fresh ink into the BIDS. • <colour> = Target BIDS/IDS colour (C,M,Y,K)

Check BIDS/IDS Ink Tank **OUT** Float Switch	Dyn-ultron:BIDS_OUT_<colour>:get "value"	0:"active"	• Example shown is expected under NORMAL operational conditions. • During NORMAL operations, if inactive, the system will generate an INK_OUT error. • <colour> = Target BIDS/IDS colour (C,M,Y,K)

4.3.4 Component driving: PINCH VALVE

Table 17 – Low Level Pinch valve motor drive

Description	Input String	Example Response	Comments
Drive the pinch valve rotor	Dyn-ultron:VALVE: start 70 Dyn-ultron:VALVE: stop	0:<SU> 816 Dyn-ultron:VALVE,"servo","running","on"	• Example of a good response • The valve will not stop at a given point • **Ensure the head is in DEPRIME_IDLE state before using**

4.3.5 Component driving: INK/WASTE PUMPS

Table 18 – Low Level Bulk ink/Waste pump driving

Description	Input String	Example Response	Comments

4.3.6 Component driving: CIRCULATION PUMPS

Table 19 – Low Level Bulk Circulation Pump driving

Description	Input String	Example Response	Comments
Drive Circulation pumps PUMP = C & Y (upper) PUMPTWO = M & K (lower)	Dyn-ultron:PUMP: start <value> Dyn-ultron:PUMPTWO: start <value>	0:<SU> 816 Dyn-ultron:PUMP,"servo","running","on" 0:<SU> 816 Dyn-ultron:PUMPTWO,"servo","running","on"	• Driving the circulation pumps independently is useful during triage process. • <value> = Rotation speed percentage <1 - 100> • **Do not drive to >100** as risk of premature failure can occur.

4.3.7 Component driving: VALVES

Table 20 – Low Level Valve driving

Description	Input String	Example Response	Comments
Moves Capper drain solenoid to OPEN position	Dyn-ultron:CAP_SOLENOID: start	0:<SU> 4 Dyn-ultron:CAP_SOLENOID,"binaryActuator","running","on"	• OPENS the capper drain valve to atmosphere. • Typically used when the system drains the capper.
Moves Capper drain solenoid to CLOSED position	Dyn-ultron:CAP_SOLENOID: stop	0:<SU> 5 Dyn-ultron:CAP_SOLENOID,"binaryActuator","idle","off"	• CLOSES the capper drain valve, sealing a drain line. • Typically used when the system has completed the drain capper algorithm.

4.3.8 Component driving: LIFTER MOTOR

WARNING:

- Ensure the movement area is clear of obstruction
- **CRASH RISK IS VERY HIGH.**
- Ensure the CAPPER is in its home (retracted) position BEFORE driving
- Ensure the WIPER is in the HOME position BEFORE extending the capper OR moving the LIFTER

Table 21– Low Level Lifter Motor driving

Description	Input String	Example Response	Comments
Drive the lifter in target direction (UP/DOWN)	Dyn-ultron:WIPER:start <VALUE>	0:<SU> 183 Dyn-ultron:LIFTER,"servo","running","on"	• <value> = drive speed percentage <1 – 100> • A **NEGATIVE** value moves the lifter **UPWARDS** • A **POSITIVE** value moves the lifter **DOWNWARDS** • **DO NOT DRIVE outside of -25 > 25 values** (slow movement) as risk of premature failure crashing can occur.
Stops the lifter regardless of position	Dyn-ultron:WIPER:stop	0:<SU> 188 Dyn-ultron:LIFTER,"servo","running","on"	• Will halt the motor. • Use is needed as the lifter will **not** stop driving and will ignore the position limit switches. • To utilize and reduce crash risk, use commands in Table 9

4.3.9 Component driving: MAINTENANCE SERVO MOTOR

WARNING:

- Ensure the movement area is clear of obstruction
- Ensure the LIFTER is in a raised position BEFORE driving
- Ensure the CAPPER is in its home (retracted) position BEFORE driving
- Ensure the WIPER is in the HOME position BEFORE extending the capper OR moving the LIFTER

Table 22 – Low Level Maintenance Module Servo Motor driving

Description	Input String	Example Response	Comments
Drive the wiper servo motor in either direction.	Dyn-ultron:WIPER:start -10	0:<SU> 2 Dyn-ultron:WIPER,"servo","running","on"	• A **NEGATIVE** value moves the Wiper towards the green handle side. • A **POSITIVE** value moves the Wiper towards the home/parked position. • **DO NOT DRIVE outside of -10 > 10 values** (slow movement) as risk of premature failure crashing can occur.
Immidiately stops the motor.	Dyn-ultron:WIPER:stop	0:<SU> 2 Dyn-ultron:WIPER,"servo","running","off"	• **Do not drive to >100** as risk of premature failure can occur. • This will cause the servo to error out. A power cycle may be needed to recover.

4.3.10 Component driving: MISC.

Table 23 – Low Level miscellaneous component driving

Description	Input String	Example Response	Comments
Toggle ON/OFF AES fan control output	Dyn-ultron:AEROSOLE_FAN: start	0:<SU> 4 Dyn-ultron:AEROSOL_FAN,"binaryActuator","running","on"	• Drives output for AES fan control • Optional, as most OEM's use external control methods
	Dyn-ultron:AEROSOLE_FAN: stop	0:<SU> 4 Dyn-ultron:AEROSOL_FAN,"binaryActuator","idle","off"	

4.3.11 Component driving: Waste Ink Management Module.

Table 24 – Low level Waste Ink Maintenance Module

Description	Input String	Example Response	Comments
Toggle ON/OFF WIMM Vacuum Pump	Dyn-ultron:WIMM: start 57	0:<SU> 1048 Dyn-ultron:WIMM,"regulator","running","off"	• Enables Ultron output ON (+24v) •
	Dyn-ultron:WIMM: stop	0:<SU> 1060 Dyn-ultron:WIMM,"regulator","stopping","on"	

			• **Ensure the pump is not left on for prolonged periods of time**

5 GYMEA

5.1 Overview

The most common area for interrogating the Gymea surrounds consumable information. This is particularly useful when triaging around printheads and ink dongle information.

Note:

- **ALL commands are case sensitive and require the user to manually input each string.**
- **Commands can be used during triage only. They are NOT to be used in any other instance.**
- **A printhead MUST be present to use some of the PH & QAi commands listed.**

CAUTION

Great care and attention MUST be taken when using these commands. Failure to take appropriate precautions may result in ink leaks OR damage to the print head.

Using these commands are at the risk of the Field Engineer using them. Memjet does NOT hold any responsibility for any damage that can occur from misuse

5.2 Accessing the command line/interface

1. SSH to target Ross board (As defined in section 2.2.1.)
   - Use either **SSH** OR **PUTTY** terminal
   - Example: *ssh root@192.168.100.200* OR *RS20300300.local*
2. Type the following command followed by the return key: *nc localhost 9000*
   - This will move the cursor to the next line
   - All commands at this stage are CASE sensitive and must be individually typed
3. Use the tables defined below to target required areas.

5.2.1 High Level Commands

Table 25 – High level Gymea Commands

Description	Input String	Example Response	Comments
Check current GYMEA & FPGA versions	tcl job version	tcl#Gymea: MJ4.13.8 Build:27, FPGA: 7.06.02tcl#	• Example of a Good/OK response • Version response will vary depending on software build installed (R5.0.2, R5.3.3, etc..)

Checks the status of the QAi subsystem.	tcl cm get_qai_subsystem_state	tcl#READYtcl#	• Example of a Good/OK response • Good to check when using other PH related commands/checks.
Shut down the QAi subsystem	tcl pep shutdown_qai_subsystem	tcl#Oktcl#	• Example of an expected response. • Rarely needed to be used out-side of Memjet. • Most issues relate to the attempts to start the QAi subsystem, which is automated within the DF algorithms under normal operations.
Starts/resumes the QAi subsystem after it has been shut down (OR if an error failed to start)	tcl pep resume_qai_subsystem	tcl#Oktcl#	• Example of a Good/OK response • A lot of commands need the QAi subsystem to be running. • If an error is generated, **Review the Gymea log** for further information.
Check connected QAi enabled components	tcl cm get_qac_states	tcl#States: PRINTER:OK,404F420ABC7A0C0139 PRINTHEAD_0:OK,2060403AD495160139 RIT_C:OK,8037400ABC7A0C0139 RIT_M:OK,B051420ABC7A0C0139 RIT_Y:OK,F04A410ABC7A0C0139 RIT_K:OK,403A420ABC7A0C0139tcl#	• Example of a good response • Shows the ROSS board, print head and connected ink dongles. • Useful when diagnosing Ink dongles or associated RJ12 cabling • Can be called if the printhead has not been prepared for use

5.2.2 Consumables Specific: PRINTHEAD

Table 26 – Print Head Gymea commands

Description	Input String	Example Response	Comments
Checks Print head usage status.	tcl hh is_ph_prepared_for_use	tcl#truetcl#	• Example of a good response • Used to determine the current printhead prepared for use status. • If an error is generated, **Review the Gymea log** for further information.
Prepares the Printhead for usage.	tcl hh ph_prepare_for_use	tcl#Oktcl#	• Example of a good response • **Requires QAi subsystem to be running (table #23)** • Performs some baseline checks to the print head • If an error is generated, **Review the Gymea log** for further information.

Prepares the Printhead for removal from the system.	tcl hh ph_prepare_for_removal	tcl#Oktcl#	• Example of a good response • **Requires QAi subsystem to be shutdown (table #23)** • Ensure the print head is DE-PRIMED BEFORE REMOVAL (Table #2) • If an error is generated, **Review the Gymea log** for further information.
Check ph prime status	tcl pep ph_primed	tcl#truetcl#	• Check the prime status of the printhead • Useful if the head was removed without de-priming • State stored in printhead. • Requires good connections to the printhead and QAi subsystem to be running **(table #23).**
Force primed status of the installed printhead	tcl pep ph_primed true	tcl#Oktcl#	• Force changes the prime status of the printhead. • Useful if the head was removed without de-priming. • State stored in printhead. • Can be useful if a failure occurs during certain sequences.
	tcl pep ph_primed false	tcl#Oktcl#	
VPOS readback check	tcl pep get_vpos PRINTHEAD_ALL	tcl#4.40 4.40tcl#	• Checks the last known VPOS value of each printhead (LEADING & LAGGING). • MUST have a known temp value (This is obtained during priming algorithm.
To Read values of interest from printhead QA chips	tcl mm get_ph_values PRINTHEAD_0	tcl#Error – The printhead is not ready for usetcl#	• Example of a head not ready for use. See **APENDIX A** for an example good response. • Printhead 0 = **LEADING**. Contains licensing information and consumable data • Printhead 1 = **LAGGING**. Contains configuration data
	tcl mm get_ph_values PRINTHEAD_1		

5.2.3 Consumables Specific: INK TANKS/DONGLES

Table 27 – Ink Tank/Dongle commands

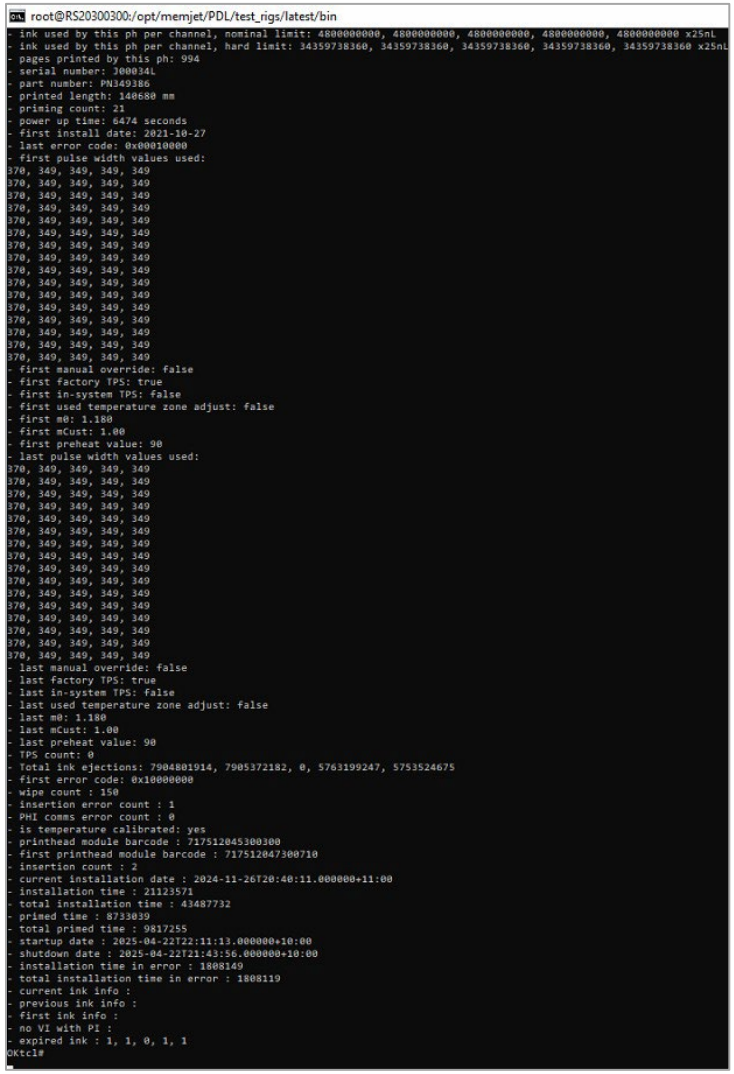
Description	Input String	Example Response	Comments
Check connectivity of a Virtual Ink dongle.	tcl cm qac_present RIT_<colour>	Tcl#device presenttcl#	• Example of an expected response when a suitable dongle is connected. • <colour> = Target colour (C, M, Y, K)
Check the initial Virtual Ink fill limit of a given attached ink dongle.	tcl cm get_vc_max RIT_<colour>	tcl #MAX: K:80000000tcl#	• Example response from a connected 2 liter dongle • <colour> = Target colour (C, M, Y, K)
Check the remaining Virtual Ink of a given attached ink dongle.	tcl cm get_vc_left RIT_<colour>	tcl #LEFT: K:38663325tcl#	• Example response showing the remaining Vi available from the connected 2 liter ink dongle.

			• <colour> = Target colour (C, M, Y, K)

6 APPENDIX.

6.1 APPENDIX.A – Example response

Figure 5 - Example of a Good Response (Section 5.2.2) – tcl mm get_ph_values PRINTHEAD_0



6.2 APPENDIX.B – Ultron Example response

Table 28 - Ultron status command response

0:<SU> 210 Dyn-ultron,"dynamo","ready"|Dyn-ultron:VALVE_LIM_A,"flag","inactive"|Dyn-ultron:CAP_OUT,"flag","active"|Dyn-ultron:BIDS_FILL_Y,"regulator","running","off"|Dyn-ultron:HEARTBEAT_LED,"binaryActuator","idle","off"|Dyn-ultron:AIR_VENT_PUMP,"binaryActuator","idle","off"|Dyn-ultron:WIMM,"regulator","idle","off"|Dyn-ultron:VALVE_LIM_B,"flag","active"|Dyn-ultron:LIFTER,"servo","idle","off"|Dyn-ultron:BIDS_FILL_K,"regulator","running","off"|Dyn-ultron:BIDS_FILL_M,"regulator","running","off"|Dyn-ultron:AEROSOL_FAN,"binaryActuator","idle","off"|Dyn-ultron:Ph-LEADING,"phHwMulti",false,false,false,false,false,false,false,true,true|Dyn-ultron:PUMP,"servo","idle","off"|Dyn-ultron:CAP_HOME,"flag","active"|Dyn-ultron:CAP_SOLENOID,"binaryActuator","idle","off"|Dyn-ultron:VALVE,"servo","idle","off"|Dyn-ultron:CAP,"servo","idle","off"|Dyn-ultron:Ph-LAGGING,"phHwMulti",false,false,false,false,false,false,false,true,true|Dyn-ultron:LIFT_CAP,"flag","inactive"|Dyn-ultron:LIFT_HOME,"flag","active"|Dyn-ultron:PUMPTWO,"servo","idle","off"|Dyn-ultron:BIDS_FILL_C,"regulator","running","off"|Dyn-ultron:WIPER_HOME,"flag","inactive"|Dyn-ultron:WIPER,"servo","idle","off"|Dyn-ultron:LIFT_PRINT,"flag","active"

6.3 APPENDIX C - NGQ Bus Map

6.3.1 Overview

- This information is particularly useful when reviewing log files in an escalation (Subject dependent).
- QAi information is managed by the Gymea Service (See Section #5)
- The printhead communications protocol which drive the printhead is entirely separate to the I2C/LSS bus connection to the NGQ chip on each die array.
- The NGQ chip is a customized ST Microelectronics part. The rest of the printhead is an entire proprietary design. The I2C/LSS bus on each die array has a separate 2.5V power supply pin and separate clock and data pins to all other communications. There are no other devices connected to each of these two I2C/LSS busses.
- The two printhead bus connections feedback to the FPGA
- The Glenbeigh has 4 bus connections and are detailed in the table below:

Table 29 - NGQ Bus Map table

Bus ID	Description
bus 0	• connects to the printer QA chip and Twin port • Physically located embedded onto the ROSS board
bus 1	• Connects to the inktank or dongle dock
bus 2	• Connects to printhead die array ***0*** i2c bus • die array **0** NGQ device (connected to **bus 2**) is configured with an address of ***0x3***. • It is the device which is configured with the license and manufacturing data.
bus 3	• Connects to printhead die array ***1*** i2c bus • die array **1** NGQ device (connected to **bus 3**) has an address of ***0x20 (32)*** . • This device is only used to store calibration and other application information**.**