WEAVE Science Processing and Analysis
Interface Control Document

Project name	WEAVE
Release	~~Draft~~/Final: Version 8.00 Date: 2 May 2024

Author(s):	Jim Lewis Mike Irwin David Murphy Eduardo Gonzalez-Solares Luis Peralta de Arriba Alireza Molaeinezhad Nicholas Walton
Owner:	Kevin Middleton
Client:	WEAVE Consortium
Document Number:	WEAVE-ICD-027

Document History

Document Location	The document can be found at : http://bscw.ing.iac.es/bscw/bscw.cgi/599779

Revision History

Revision date	**Version**	**Summary of Changes**	**Changes marked**
01-Mar-2015	0.10	JRL: Document created	
02-Mar-2015	0.20	JRL: Mods made in response to input from Mike	
16-Mar-2015	0.30	JRL: Incorporated Edu's text. Mods in response to input from Sergio	
23-Mar-2015	0.32	JRL: small changes	
24-Mar-2015	0.33	JRL: small changes to APS section (typos etc) plus a new version of table 6	
28-Mar-2015	0.40	JRL: many changes to extension headers and updates to table 3	
28-Mar-2015	0.50	JRL: updated approvals, distributions and reference lists	
31-Mar-2015	1.00	FDR version	
01-Apr-2015	1.10	Added approvals	PSO
14-Sep-2016	1.20	Many changes to sections 2-4 in order to bring the document into agreement with ICS-027 and current practice	
06-Oct-2016	1.30	JRL: small changes to section 2 main to do with FIBINFO table description	
13-Dec-2016	1.40	JRL: Added section on quick-look	
21-Dec-2016	1.50	JRL: Some very small changes to text and fixes to table and figure cross references	
04-July-2017	1.60	JRL: Adjustments mainly to fibinfo table specification. Also added information about sensitivity function	
24-Jan-2018	1.70	JRL: Table 7 updated. Fixed OBSTART definition in table 2	

16-July-2018	1.80	JRL: Section 3.2.1 added description to show the naming convention for output exposures and OB level stacks. Provenance header keywords also included	
26-July-2018	1.90	JRL: Added section 3.2.2 on L1 data cubes for LIFU.	
10-Sep-2018	1.95	JRL: Added section 3.2.3 on super-stacked MOS output files.	
12-Sep-2018	1.96	AM: Updated section 6.2.1 and 6.2.2 on new structure of APS products (L2).	
20-Jul-2020	7.00	Revision of the whole document reflecting changes of the data model stabilisation	
25-Nov-2020	7.00	Change to TARGPROG string length	2.2.4
07-July-2021	7.00	DM: Added CPS processing rules based on OBCLASS	2.3
26-Nov-2021	8.00	DM: RFC-027 implementation. Added OBWVFILE, OBFBFLAT & CALDATE keywords to L1 PHU, new SENSFUNC HDU defined. Updates to various keywords in Sec 2.3 to agree with RFC and [RD04]	Table 10, 3.2.2.1.2, Table 12, 2.3
21-Mar-2022	8.00	DM: removed reference to dummy bias, added section on validation and ingest with OR. Missing OBCLASS case added for L1 processing. warc data transfer section written.	Sec 2.2, 2.3, 2.5, 3.1.4
22-Sept-2022	8.00	AM: Updated/added data structure for stellar-mode APS IFU	Sec 5.2.3
02-Nov-2022	8.00	ML: WAS signature (7.1) and download 7.2). aps_driver.json and aps_superdriver.json to WAS (3.1.5). Data transfer of complete night (3.1.7)	Added sec. 7.1 and 7.2 Modified sec. 3.1.5 and added 3.1.7 and 3.1.8
28-Jun-2023	8.00	DM: Added dataflow sections. Updates to descriptions relevant to dataflow signalling. Inclued reference to WEAVE Data Model document.	Sec 7,8,9,10 Sec 3.1, 5.1
12-Jul-2023	8.00	AM: line strength indices (Table 36) and emission lines list (Table 37) updated.	Sec 5.2

14-Jul-2023	8.00	DM: review, reference fixes and addition of Table 6. Update of 3.2.1 to describe the PI survey specification table.	Sec. 3.2.1
20-Jul-2023	8.00	SCT: language and content edits	all
02-May-2023	8.00	DM: Proposed changes checked and accepted. Freeze for v8.00	all

Approvals This document requires the following approvals.

Name	Title	Approval Date	Issue Date	Version
Nicholas Walton	CPS System Manager	31-Mar-15	31-Mar-15	1.0
Carlos Allende-Prieto	APS System Manager	31-Mar-15	31-Mar-15	1.0
Marcello Lodi	WAS System Manager	31-Mar-15	31-Mar-15	1.0
Sergio Pico	OCS System Manager	31-Mar-15	31-Mar-15	1.0
Don Carlos Abrams	WEAVE Project Manager	31-Mar-15	31-Mar-15	1.0
Gavin Dalton	WEAVE Principal Investigator	31-Mar-15	31-Mar-15	1.0
Scott Trager	WEAVE Project Scientist	31-Mar-15	31-Mar-15	1.0
Chris Benn	WEAVE Instrument Scientist	31-Mar-15	31-Mar-15	1.0
Kevin Middleton	WEAVE System Engineer	31-Mar-15	31-Mar-15	1.0

Distribution This document has been distributed to:

Name	Title	Issue Date	Version
Johan Pragt	SPE System Manager	1-Apr-15	1.0
Chris Mottram	DET System Manager	1-Apr-15	1.0

TABLE OF CONTENTS

1 INTRODUCTION..10

1.1 Abbreviations..10

1.2 Purpose..10

1.3 References..10
1.3.1 Applicable Documents..10
1.3.2 Reference Documents..11

2 OCS → CPS, OCS → WAS..12

2.1 Data Transfer..12

2.2 Raw Data Requirements..12
2.2.1 FITS Primary Header Unit..13
2.2.2 Detector FITS Image Extension..15
2.2.3 Fibre FITS Table Extension..16

2.3 CPS processing rules..19

2.4 OCS end-of-night signalling to CPS..19

2.5 CPS data validation and ingestion..20

3 CPS → APS, WAS..20

3.1 Data Transfer..20
3.1.1 Input FITS Catalogues..21
3.1.2 L1 Products..21
3.1.3 L1 Superproducts..22
3.1.4 L1 Calibration Files..22
3.1.5 L2 Analysis Products..23
3.1.6 Contributed Software Products..23
3.1.7 Open time survey specification files..23
3.1.8 Data transfer to WAS..24

3.2 L1 Output FITS Files: L1 Products and L1 Superproducts..24
3.2.1 Types of L1 Output FITS Files..24
3.2.1.1 Classification of the Types of L1 Output Files..26
3.2.1.2 When Is Each Type of L1 Output File Created?..26
3.2.2 Extensions of the L1 Output Files..27
3.2.2.1 Contents of the header of each extension..30
3.2.2.1.1 Keywords contained in the primary header..30
3.2.2.1.2 Keywords contained in the image extensions..32
3.2.2.1.3 Keywords contained in the table extensions..33

3.2.3 Structure of Each Type of L1 Output FITS Files 33
3.2.3.1 Structure of the Single Files 34
3.2.3.2 Structure of the Stack and Superstack Files 35
3.2.3.3 Structure of the Stackcube, Stackcubelet, Supercube and Supercubelet Files 36
3.2.3.4 Structure of the 1-d [MOS] Supertarget Files 36
3.2.3.5 Structure of the LIFU and mIFU Supertarget Files 39

3.3 L1 Calibration Files 39
3.3.1 warc Files 39

4 QUICK-LOOK 40

4.1 QL → OCS 41
4.1.1 Data Transfer 49

4.2 QL → Observers 49

5 APS → CPS AND APS → WAS 51

5.1 Data Transfer 51

5.2 L2 Output FITS Files 52
5.2.1 Types of L2 Output FITS Files 53
5.2.2 When is each type of L2 output file created? 54
5.2.3 Structure of L2 Outputs FITS Files 57
5.2.3.1 Science Data Products in MOS mode 57
5.2.3.2 Science Data Products in IFU mode (LIFU, mIFU) 63

6 CPS → USERS 76

7 WAS → USERS 77

7.1 The WAS signature 77

7.2 Data download (WAS → users) 77

8 SPA BACKEND DATAFLOW SYSTEM 78

8.1 The proc_state table 79
8.1.1 Product specification 79
8.1.2 Product metadata 79
8.1.3 Processing and analysis: *procstate* 79
8.1.4 Dataflow states: *flowstate* and *wastate* 80

8.2 The proc_log table 81

8.3 The proc_daemon table 82

8.4 The proc_prefs table 82

8.5 Dataflow events 83
8.5.1 Non-actionable events 83
8.5.1.1 Information updates 83
8.5.1.2 Rescinded actions 83
8.5.2 Actionable events 83
8.5.2.1 OR-actionable events 83
8.5.2.1.1 A trigger to generate further entries in the proc_state table 84
8.5.2.1.2 A trigger to change different rows and/or columns in the proc_state table 84
8.5.2.1.3 Returning a warning or error when users attempt something that is not permitted 84
8.5.2.2 Node-actionable events 84
8.5.2.2.1 Data need to be processed 84
8.5.2.2.2 Data need to be transferred to an external node 84

8.6 Event pathways 85
8.6.1 Normal processing (1) 86
8.6.1.1 OR housekeeping 86
8.6.2 Retraction (2) 86
8.6.2.1 OR retraction housekeeping 87
8.6.2.2 Retraction during dataflow operations 87
8.6.2.3 Cascade retraction 87
8.6.3 Pipeline failure (3) 88
8.6.4 Validation failure 88
8.6.5 External transfers to/from CASU 88

9 SPA DATAFLOW INTERFACING WITH THE OR 89

9.1 Preliminary installation 90
9.1.1 Configuration files 91
9.1.2 Integration with existing Python code 91

9.2 Calls to the OR database 91
9.2.1 Availability of data (for processing, for transfer) 92
9.2.1.1 Terminal recipe 92
9.2.1.2 Python recipe 93
9.2.1.3 SQL equivalent 94
9.2.2 Updating the state of data (processing, transfers) 94
9.2.2.1 Terminal recipe 94
9.2.2.2 Python recipe 95
9.2.2.3 SQL equivalent 95

9.3 Rescinding updates 96
9.3.1 SQL equivalent 96

10 DETAILED NODE USE CASES 97

10.1 L1 pipeline 97

10.2 L2 pipeline 97

10.3 WAS archive 97
10.3.1 Regular data releases 97
10.3.2 Irregular data releases 99

11 FULL DB SCHEMA FOR EACH DATAFLOW TABLE 99

1 INTRODUCTION

WEAVE is a new wide-field spectroscopy facility proposed for the prime focus of the 4.2m William Herschel Telescope. The facility comprises a new 2 degree field of view prime focus corrector with a 1000-multiplex fibre positioner, a small number of individually deployable integral field units, and a large single integral field unit. The IFUs and the MOS fibres can be used to feed a dual-beam spectrograph that will provide full coverage of the majority of the visible spectrum in a single exposure at a spectral resolution of ~5000 or modest wavelength coverage in both arms at a resolution ~20000. The instrument will provide spectroscopic sampling of the fainter end of the Gaia astrometric catalogue, chemical labelling of stars to V~17, and dedicated follow up of substantial numbers of sources from the medium deep LOFAR surveys.

1.1 Abbreviations

The abbreviations and acronyms used in this document can be found in WEAVE-MAN-001.

1.2 Purpose

Figure 1 in WEAVE-SYS-010 (Overview of WEAVE Survey Information/Data Flow), shows the information flow into the SPA from the OCS and how the information will be transferred within the various components of the SPA. This document describes the interface between all these components.

When describing the dataflow system (Sections 8, 9 and 10), database tables and SQL are represented within this document using this styling. Columns with the tables, when referred to in the explanatory text, are *italicised*.

Under draft versions, some text is subject to review or update or removal.

1.3 References

1.3.1 Applicable Documents

Document Identifier	Document Title
[AD01] WEAVE-APS-001	APS Requirements and Technical Description
[AD02] WEAVE-APS-002	APS Management Description
[AD03] WEAVE-APS-003	APS Development Plan
[AD04] WEAVE-APS-004	APS Hardware Plan
[AD05] WEAVE-CPS-001	CPS Requirements and Technical Description
[AD06] WEAVE-CPS-002	CPS Management Description
[AD07] WEAVE-CPS-003	CPS Development Plan
[AD08] WEAVE-CPS-004	CPS Hardware Plan
[AD09] WEAVE-ICD-027	SPA Interface Control Document
[AD10] WEAVE-SPA-001	SPA System Description and Overview
[AD11] WEAVE-SPA-002	SPA Commissioning Plan
[AD12] WEAVE-SPA-003	SPA Risk Management Plan
[AD13] WEAVE-SPA-004	SPA Operations and Maintenance Plan

[AD14] WEAVE-SYS-010	Survey Information and Data Flow
[AD15] WEAVE-WAS-001	WAS Requirements and Technical Description
[AD16] WEAVE-WAS-002	WAS Management Description
[AD17] WEAVE-WAS-003	WAS Development Plan

1.3.2 Reference Documents

[RD01] WEAVE-DET-004	WEAVE UltraDAS Initial Investigations
[RD02] WEAVE-ICD-008	Interface Control Document Master List
[RD03] WEAVE-ICD-022	Top level N-squared diagram
[RD04] WEAVE-ICS-011	FITS File Description of WEAVE Observations
[RD05] WEAVE-MAN-001	Abbreviations and Definitions
[RD06] WEAVE-MAN-004	Product Breakdown Structure
[RD07] WEAVE-MAN-005	Work Package Assignments
[RD08] WEAVE-MAN-006	WEAVE Risk Management Plan
[RD09] WEAVE-MAN-007.2	Log Files – Risk Log
[RD10] WEAVE-MAN-009	Project Management Team Responsibilities
[RD11] WEAVE-MAN-013	WEAVE instrument block diagram
[RD12] WEAVE-MAN-016	Executive Summary
[RD13] WEAVE-MAN-023	Stage Plan: Final Design Stage (Third Stage)
[RD14] WEAVE-MAN-024	Stage Plan: Manufacturing, Assembly, Integration and Testing (Fourth Stage)
[RD15] WEAVE-MAN-037	Activity Network Diagram, Gantt Chart and Task Sheet
[RD16] WEAVE-MAN-039	WEAVE Project Costs and Schedule
[RD17] WEAVE-OCS-004	Observatory Control System PDR Document
[RD18] WEAVE-SCI-001	WEAVE Science Requirements
[RD19] WEAVE-SCI-002	WEAVE Science Case
[RD20] WEAVE-SCI-003	WEAVE Operational Constraints and Operational Requirements
[RD21] WEAVE-SCI-004	WEAVE Operations Concept
[RD22] WEAVE-SCI-006	Requirements from the Concept of Operations Document
[RD23] WEAVE-SYS-001	Instrument Development Specification
[RD24] WEAVE-SYS-002	Requirements Engineering Procedure
[RD25] WEAVE-SYS-009	Preliminary Commissioning Plan
[RD26] WEAVE-WPD-009.01	WPD for the WEAVE Archive System
[RD27] WEAVE-WSF-003	Preliminary design of the focal plane calibration module
[RD28] WEAVE-SPE-016	Spectrograph Mechanical Design
[RD29] WEAVE-SPA-006	CNAMES for IFU targets
[RD30] WEAVE-ICD-030	WEAVE SPA to SQG/QAG Interface Control Document
[RD31] WEAVE-ICD-025	Configure XML Definition
[RD32] WEAVE-SPA-008	The WEAVE Data Model

2 OCS → CPS, OCS → WAS

2.1 Data Transfer

The OCS will transfer the raw data to the CPS and WAS on a daily basis using the protocol (described below) currently used to transfer data from the ING telescopes to CASU. This is based on a secure-copy protocol where the checksum of a transferred file is compared to the original to ensure a correct transfer. The steps are:

- compress the image,
- calculate checksum,
- uncompress the image in a scratch directory to check that the compression has been done correctly and remove the uncompressed image,
- send the compressed image using scp,
- check that the remote and local checksum agree.

The OCS will copy the data to nightly subdirectories on the remote side (CASU and WAS). No changes to the RAW will be made from that point, to ensure consistency between the three locations where the files are stored.

The CPS will ingest the raw files into a database table to check for integrity and to ensure that header keywords necessary to perform the data reduction are present and correct.

The WAS will ingest the raw files on a daily basis and will do a preliminary validation check.

2.2 Raw Data Requirements

The data acquisition for each arm of WEAVE will be done by a separate instance of UltraDAS. This was designed so that the exposure parameters for each arm can be controlled separately, e.g. the exposure times could be different or one arm can be switched off altogether. In what follows, we treat the data from each arm as an independent entity. However, the occasion will arise where it is desirable for red and blue to be combined and it will be necessary to be able to associate the observations from both arms. This can be achieved easily so long as the header keywords are written correctly in the raw data files.

There will be two detectors in each arm and the data from each detector will be stored in a separate image extension. In what follows below, for images from the blue spectrograph, the values of <detector1> and <detector2> will be ‘BLUE1’ and ‘BLUE2’. Similarly, for the red spectrograph they will have the values of ‘RED1’ and ‘RED2’. In addition to the image header units for each detector, there will also be a FITS binary table with information about what has been observed in the current exposure. Our agreed structure for the raw FITS files will be as follows:

Table 1 – Agreed structure of raw WEAVE FITS files.

Extn #	**Name**	**Description**
0	Primary	The primary header unit. This unit header will contain all the information common to the observation as a whole. There will be no image data.
1	<detector1>_DATA	The image data for the first detector. The header will contain detector specific information.
3	<detector2>_DATA	The image data for the second detector.
5	FIBTABLE	The fibre binary FITS table.
6	GUIDINFO	Guide coordinates/errors and fluxes.
7	METINFO	Meteorological information for the exposure.

The ordering of the header units must not change. All image and table extensions must be named using the EXTNAME keyword, apart from the primary. The names are chosen so that it is easy to see the association between header units that refer to the same detector. To save space, the image extensions must be losslessly Rice tile compressed (ideally using *fpack*), but this should not be reflected in the file name – a tile compressed FITS file is still a valid FITS file and hence no name change is needed.

2.2.1 FITS Primary Header Unit

According to [RD4] and common practice, the primary header unit of each file will contain only header cards relating to the general observational parameters. Any detector level parameters will appear in the relevant image extension. That reference contains a full list of FITS header keywords that will be available in the raw data files. Although [RD4] contains a full list of the contents of the primary header, in the table below we list only the header keywords that absolutely must be present in the primary header unit for the reduction to proceed. Ideally, there should be a certain amount of redundancy of information in the headers to make sure there is no ambiguity about the data.

Table 2 – Required keywords from the raw FITS primary header

Keyword	**Type**	**Meaning**
RUN	int	The run number.
CCDXBIN	int	The binning factor in the *x* (spectral) direction.
CCDYBIN	int	The binning factor in the *y* (spatial) direction.
DATE-OBS	char	[yyyy-mm-dd] Date of observation in the FITS-approved format.
UT	char	[hh:mm:ss.sss] UTC of start of observation.
EXPTIME	real	[s] The exposure time.
OBJECT	char	For science observations, this should be the “name” attribute of the <observation> element in the source XML. Otherwise, this keyword should one of the following: ‘ARC’, ‘BIAS’, ‘DARK’, ‘FIBRE_LAMPFLAT’, ‘FIBRE_LAMPFLAT’, ‘FIBRE_TWIFLAT’, ‘FIBRE_SKYFLAT’,

		‘DETECTOR_FLAT’, ‘SALSA_LAMPFLAT’, ‘SALSA_TWIFLAT’, ‘SALSA_SKYFLAT’, ‘SALSA_ARC’, ‘STDFIBRE_LAMPFLAT’, ‘STDFIBRE_TWIFLAT’, ‘STDFIBRE_SKYFLAT’
OBSTYPE	char	The type of observation. For the CPS, the list of possible values must include: ‘ARC’, ‘BIAS’, ‘DARK’, ‘DETECTOR_FLAT’, ‘FIBRE_FLAT’, ‘FLUX_STD’ ‘RV_STD’, ‘SALSA_ARC’, ‘SALSA_FLAT’, ‘SKY’ and ‘TARGET’. Obviously, those in charge of doing instrumental health monitoring will need others.
CAMERA	char	This will indicate to which arm the FITS file refers. The possible values will be ‘WEAVERED’, ‘WEAVEBLUE’.
LATITUDE	real	[deg] The latitude of the telescope corrected for polar motion.
LONGITUD	real	[deg] The longitude of the telescope corrected for polar motion.
HEIGHT	real	[m] The height of the observing floor above sea level.
MJD-OBS	real	[days] The modified Julian date of the start of the observation. This is to be used to associate the data files produced by both arms in a single observation and hence this value must be exactly the same between the two files.
CAT-NAME	char	The catalogue name for exposure.
CAT-RA	char	[hh:mm:ss.sss] Target Right Ascension.
CAT-DEC	char	[sdd:mm:ss.ss] Target Declination.
ST	char	[hh:mm:ss.sss] The local sidereal time at the start of the observation.
AMSTART	real	Airmass at the start of the observation.
AMEND	real	Airmass at the end of the observation.
AIRMASS	real	Effective mean airmass during observation.
VPH	char	The name of the grating in each arm: ‘N/A’, ‘LowRes’, ‘HighRes1’ or ‘HighRes2’.
CENWAVE	int	[nm] An estimate of the central wavelength in this arm.
DISPERSI	real	[nm/mm] An estimate of the dispersion.
ARCSRC	char	The name of the arc lamp used in the observation. Allowed values should be ‘NONE’, ‘Hg’, ‘Ne’, ‘He’, ‘Zn’, ‘Cd’, ‘ThAr’, ‘ThArCr’.
FLATSRC	char	An identifier for flat field source. This could be arm dependent. The values should be: • ‘NONE’ => This is not a flat field exposure. • ‘TWILIGHT’ => This is a twilight flat. • ‘W’ or ‘QTH’ => This is a dome flat. • ‘DARK_SKY’ => This is a dark sky flat. • ‘WL1,WL2,WL3’ => This is a LED flat [RD28]

		The WLn describe the LED wavelengths (in nm) corresponding to each LED . Currently the values for each LED are: LED1 375 , LED2 490, LED3 590, for the Blue Arm. LED1 590 , LED2 780, LED3 940, for the Red Arm. When any of the LED is not powered, its wavelength value will be substituted for '-'. For example: '375,-, 590' or '-,-, 940'
AGXSEE	real	[arcsec] Average value of the mean FWHM of guide stars in the *x*-axis.
AGYSEE	real	[arcsec] Average value of the mean FWHM of guide stars in the *y*-axis.
OBTITLE	char	The title associated with this observation block.
OBID	int	The observation block identifier associated with this observation.
OBSTART	real	[days] The MJD of the start time for the current OB.
OBCLASS	char	The type of observations within the OB, ie. "science", "calibration", "onskytest", "offskytest"
NSALSA	int	Number of salsa observations. 0 for non-salsa configurations.
SALSA_I	int	Index of the salsa observation. 0 for non-salsa configurations.
PLATE	char	The current position of the tumbler, i.e. 'PLATE_A', 'PLATE_B' or 'LIFU'.
OBSMODE	char	The mode of observation. This should have values of 'MOS', 'mIFU' or 'LIFU'.
FPMODE	char	The focal plane mode. This should have values of 'MOS-A', 'MOS-B', 'mIFU' or 'LIFU'.
FLDRA	real	[deg] Field centre RA.
FLDDEC	real	[deg] Field centre Dec.
CCNAME*n*[1-20]	char	The central CNAME (only for IFU observations, empty string for non IFU observations).

2.2.2 Detector FITS Image Extension

Each detector's image data will appear in a single FITS image extension. Below is a list of header items that are essential for the CPS (this ignores the keywords required by the FITS standard, such as NAXIS).

Table 3 – Required header keywords from the raw FITS image extensions

Keyword	**Type**	**Meaning**
INHERIT	bool	Extension inherits primary header (its value must be T).
EXTNAME	char	The name of the current image extension, as specified in Section 2.2.
CHIPNAME	char	The name of the detector chip.

GAIN	real	[electrons per ADU] Gain of the amplifier.
READNOIS	real	[electrons] The readnoise of the amplifier.
BIASSEC*n*[1-F,0]	char	The image section of the overscan or underscan strip where the bias can be estimated for quadrant *n*. There should be F of these.
TRIMSEC*n*[1-4]	char	The image section of the light sensitive part of quadrant *n*, where *n* is 1 to 4. This region will survive once all of the overscan and underscan regions are trimmed off after bias correction.

2.2.3 Fibre FITS Table Extension

Each exposure will have an associated FITS binary table that includes all the necessary information about each fibre, whether it has been used in the observation or not. Reference [2] has a description of the contents of this extension. In the table below, we duplicate this. It is important that if the value for a particular item is not known then it should be represented using a FITS NULL value (this null value needs to be defined for *integer* columns via their corresponding TNULL keywords). This is the only way to signal an unknown value completely unambiguously. Using a flag value (for example a magnitude of -99.99) is arbitrary and puts the onus on the pipeline to guess the user's meaning.

Table 4 – The fibre table included in raw FITS files

#	Column name (TTYPE)	Format (TFORM)	Unit (TUNIT)	Description
1	FIBREID	1I		The unique ID number of the fibre. Every fibre should have an entry in this table, whether it has been used, is broken, has been disabled or is simply parked.
2	CNAME	20A		An object name formed from the coordinates of the object (ICRS at epoch 2015.5). This is formed by splicing the RA in hours, minutes and seconds (to 2 decimal places) and the Dec in degrees, minutes and seconds (to 1 decimal place) together, including a sign for the declination. Thus an object at $3^h\,40^m\,21.^s767$ and -31° 20' 32.71" will have a unique CNAME of WVE_03402177-3120327.
3	FIBRERA	1D	deg	The RA of the position of the fibre in decimal degrees.

4	FIBREDEC	1D	deg	The Declination of the position of the fibre in decimal degrees.
5	XPOSITION	1E	mm	Fibre position in x.
6	YPOSITION	1E	mm	Fibre position in y.
7	STATUS	1A		The status of the current fibre: ‘P’ => Parked, ‘A’ => Allocated, ‘D’ => Disabled, ‘B’ => Broken.
8	TARGID	30A		The identifier of the target, as assigned by the survey, being covered by the current fibre.
9	TARGX	1E	mm	The computed x position on the plate of the target.
10	TARGY	1E	mm	The computed y position on the plate of the target.
11	TARGSRVY	15A		The name of the survey with which the target object is associated (e.g. ‘GA-LRHIGHLAT’). If it is associated with more than one survey, then this will be a list separated by commas.
12	ORIENTAT	1E	deg	Azimuthal orientation of the fibre.
13	RETRIES	1I		Number of times the fibre had to be repositioned to get within the position tolerance.
14	TARGNAME	30A		The target name (e.g. ‘Draco ET11’).
15	TARGRA	1D	deg	The catalogue RA of the object in decimal degrees, in the Gaia reference frame.
16	TARGDEC	1D	deg	The catalogue Declination of the object in decimal degrees, in the Gaia reference frame.
17	TARGEPOCH	1E	yr	The catalogue epoch of the object in decimal years.
18	TARGCAT	30A		Catalogue name and version (e.g. ‘GA-LRHIGHLAT_2020B2.fits’).
19	TARGPMRA	1E	mas/yr	Target proper motion in RA (this is the true angular motion on the sky, not a rate of change in RA).
20	TARGPMDEC	1E	mas/yr	Target proper motion in Dec.
21	TARGPARAL	1E	mas	Target parallax.
22	TARGUSE	1A		‘T’ => target, ‘S’ => sky, ‘G’ => guide, ‘C’ => calibration standard (usually white dwarfs to

				be used for flux and telluric calibration), ‘R’ => random.
23	TARGPROG	40A		Sub-programme name within survey. Delimitation with pipe (“|”) is permitted within the string.
24	TARGCLASS	12A		Input target classification.
25	TARGPRIO	1E		Target relative priority within a survey (1-10, where 1 is the lowest priority).
26	MAG_G	1E	mag	Magnitude estimate for the target in the *g* band (SDSS-like passband, AB magnitude system).
27	EMAG_G	1E	mag	The error in the magnitude estimate for the target in the *g* band.
28	MAG_R	1E	mag	Magnitude estimate for the target in the *r* band (SDSS-like passband, AB magnitude system).
29	EMAG_R	1E	mag	The error in the magnitude estimate for the target in the *r* band.
30	MAG_I	1E	mag	Magnitude estimate for the target in the *i* band (SDSS-like passband, AB magnitude system).
31	EMAG_I	1E	mag	The error in the magnitude estimate for the target in the *i* band.
32	MAG_GG	1E	mag	Magnitude estimate for the target in the Gaia G band (Vega magnitude system).
33	EMAG_GG	1E	mag	Error in the magnitude estimate for the target in the Gaia G band.
34	MAG_BP	1E	mag	Magnitude estimate for the target in the Gaia BP band (Vega magnitude system).
35	EMAG_BP	1E	mag	Error in the magnitude estimate for the target in the Gaia BP band.
36	MAG_RP	1E	mag	Magnitude estimate for the target in the Gaia RP band (Vega magnitude system).

37	EMAG_RP	1E	mag	Error in the magnitude estimate for the target in the Gaia RP band.

As this is a FITS table, there will be many header items that are required by the FITS standard. The only other header item that is required by the CPS is a value for EXTNAME, which should be 'FIBTABLE'.

2.3 CPS processing rules

The Operational Repository and L1 pipeline rely on the OBCLASS header entry to determine how to handle data from the instrument. We refer readers to [RD31] for a description and definition of the OBCLASS values.

Table 5 - QL/CPS processing capability based on OBCLASS

OBCLASS	QL processing	L1 processing	Example	Notes
science	🟩	🟩	Typical survey OB	
calibration	🟩	🟩	Sequence of 1x spatial binned bias frames	These OBs form the basis of the CASU calibration library
onskytest	🟨	🟨 Best effort	IFU focus tests for WEAVECOM-55 with short exposures*	Commissioning OBs will be similar to science OBs, but with non-standard instrumental setup
offskytest	🟨 Best effort	🟥	Sequence of 4x spatially-binned bias frames	TBD if CASU can ingest these into the Operational Repository
<missing>	🟨 Best effort	🟥	Engineering data with no XML provenance	The entire OB packet will be missing from these files

* whilst short exposures are not a problem, the test requires exposure times that are not defined within the PROGTEMP scheme, meaning that the XML is non-standard with respect to the WEAVE data model.

2.4 OCS end-of-night signalling to CPS

At the end of the night, OCS will transmit an "EON" (end of night) file. This is a simple text file that contains the standard ING observing report in addition to an entry that details any Observing Block retractions applied to the OB database since the previous EON file was issued. It is in the format:

```
<nightobs>.wht
```

The arrival of this file signals to CPS that no more data will be taken, and that processing tasks may commence on the night. Because this file is read by automated processes, it is important that the format does not change, except by agreement under a change request.

2.5 CPS data validation and ingestion

Once the EON file has been received, the Operational Repository will perform a series of tasks and checks to validate and ingest the data. These are:

1. FITS structure check (in the event the raw FITS file extensions were not written out in the correct order)
2. File and group permissions
3. Checks against NASA *fitsverify*
4. Checks against the WEAVE data model specification (in this instance [RD04])
5. Ingestion of the raw data into the Operational Repository database
6. Any required updates to the OB status arising from retracted OBs at OCS

If these validation stages pass, the OR will signal to the L1 pipeline that this night can be processed.

3 CPS → APS, WAS

3.1 Data Transfer

There are 8 products that CPS provides to the APS and WAS nodes:
1. Input FITS catalogues
2. L1 products
3. L1 superproducts
4. L1 calibration files
5. L2 analysis products
6. L2 Contributed Software (CS) products
7. aps_driver.json and aps_superdriver.json files
8. Open time survey specification files (only to WAS)

For cases 1–7, the data are pulled from CASU to the destination node. For each product, we use the same transfer mechanism: a dataflow state of "PENDING" (see Section 8.1.4) in the Operational Repository proc_state table. The output fields of a WDAP (Section 9.2.1) recipe are consistent across all products that CASU provides, an example of which is:

```
20221025,/data/apm63_/weave/L1/20221025/,2023-06-12 22:45:02.874588+01:00,6
```

This indicates that data are available for the night 25th October 2022, the directory hosting the data is */data/apm63_c/weave/L1/20221025*. The validation was timestamped at 22:45:02 on the 12th July 2023, and there are 6 files to transfer. *In both the "get" and the "set" recipes, a return value of 0 is expected for a successfully executed recipe.*

This (non-null) result indicates there is L1 data to transfer. Notification of the start of transfer involves using a WDAP recipe to set the L1 *wasate* (WAS) or *apstate* (APS) to "TRANSFERRING".

Authenticated connection to the CASU Operational Repository machine (apm63) is via RSA SSH key, with the above information (and defined data types within this document) providing the option to construct the required fields for an SCP transfer.

Both APS and WAS should poll for the presence of L1 data on a daily basis. Reactivity to retraction events might require more frequent polling.

Once the relevant files have been transferred to the target node, verification that these files have been received must be made. This is achieved by updating the *wastate* / *apstate* to VALIDATING, followed (for successful transfers) VALIDATED. The WDAP package provides inbuilt validation, but ultimate responsibility lies with the receiving node.

3.1.1 Input FITS Catalogues

Once survey teams have generated target catalogues for a given trimester, these are validated at CPS and unique CNAMEs are added. These catalogues shall be made available to the WAS and APS.

When validated catalogues are available, CPS will write them to a staging area based on the date the WASP submission window closed. The directories will be in the "nightobs" format of YYYYMMDD, and the contents will be a series of FITS files with a .fits extension. We refer users to [RD30] for the naming definition of these catalogues.

Both WAS and APS will be notified of availability (*wa/ap-state* of 'PENDING') of new catalogues via an entry in the proc_state table referring to the product type "CAT". For APS, transfer of L1 data via WDAP will also transfer (if not already done) the catalogues relevant to that night.

3.1.2 L1 Products

L1 products will be generated for each night within 24 hours. Once produced they are validated and QA-checked by the CPS team and subsequently released to the APS node via the proc_state table.

Due to the wide range of possible APS L1 inputs, and the resultant analysis files they generate (see Section 5.2.2) we apply an algorithmic approach to produce JSON-based driver files that identify which APS products should be generated, the L1 inputs that are used, and the provenance of these inputs if they are stacks or superproducts. The CPS generates two types of driver file –one for the typical "nightly" output of the CPS pipeline (*aps_driver.json*), and one for the superproducts (*aps_superdriver.json*) that are generated periodically by CPS. An example entry in these JSON files looks like:

```
"mWVE_01332545+3526393_01_BR_L1_P0000_APS.fits": {
"files": [
          "../L1/20200422/stackcubelet_0000001_01.fit",
          "../L1/20200422/stackcubelet_0000000_01.fit",
          ],
"prov": [
         "../L1/20200422/single_0000001.fit",
         "../L1/20200422/single_0000003.fit",
         "../L1/20200422/single_0000005.fit",
         "../L1/20200422/single_0000000.fit",
         "../L1/20200422/single_0000002.fit",
         "../L1/20200422/single_0000004.fit",
         ],
}
```

This indicates that APS should create a Blue & Red-arm Low Resolution 1x spectrally binned mIFU analysis file called "*mWVE_01332545+3526393_01_BR_L1_P0000_APS.fits*" from two stackcubelet files that used the 1st mIFU bundle. The individual files that went into these stacks are listed in the "prov" section. For the analysis of L1 supertargets, APS can specify, through the node exchange software, that all provenance data are delivered with the supertargets. The *aps_driver.json and aps_superdriver.json* files may also made available to WAS.

Depending on the archive release policy (as defined by the *was_release* entry in the proc_daemon table), L1 products will either be released when available (on a per-product basis; *was_release='product'*) or when all data products (L1, L2, CS) are available (on a per-night basis; *was_release='night'*).

3.1.3 L1 Superproducts

L1 superproducts will be generated on a ~monthly frequency, based on repeated observations of the same targets. These need to be provided to both APS and (on a longer timescale, phased with the release of L1 data, to WAS).

Superproducts will be placed in the **L1** nightobs directory on the date they were generated, meaning there should be a higher volume of entries in this subdirectory approximately monthly. In constructing these superproducts, we will ensure that the input L1 files used are also provided to APS, such that the L2 analysis pipeline is able to properly trace the provenance of these data products.

3.1.4 L1 Calibration Files

To facilitate the calculation of the line spread function, CASU will provide rebinned wavelength-calibrated arcs ("warcs") to the APS and to the WAS. In the former case, this serves both the APS pipeline as well as Contributed Software (CS) packages that might want to use these files. In the latter case, warcs are provided for use by Contributed Data Products (CDPs).

CASU will provide the "master" warc files generated on an approximately monthly basis (depending on instrument stability and L1 pipeline requirements). As such, these files will be placed in a nightobs corresponding to the date they were generated, and a signal provided to indicate availability. These will be released as data type "CAL" in the manner described above, by update to the proc_state table.

3.1.5 L2 Analysis Products

Section 5 deals with the transfer of files from the APS to CPS. This subsection describes transfer of L2 data files from the Operational Repository to WAS.

As with L1 data, once APS analysis files have been ingested and validated into the WEAVE Operational Repository, they are ready for transfer to the WAS. This involves an update to the L2 *wastate* entry in the proc_state table to "PENDING". Again, release of these data products is subject to the *was_release* entry in the proc_daemon table).

3.1.6 Contributed Software Products

Section 5 also deals with the transfer of Contributed Software products that have been run by the APS hardware. Once these files have been transferred to CPS and validated and ingested into the WEAVE Operational Repository, they are ready for onwards transfer to WAS. This involves an update to the CS *wastate* entry in the proc_state table to "PENDING". Because these data products are typically the last of all data products delivered for a night of observation, they will generally be released as soon as they are ingested, either on their own, or alongside the L1 and L2 counterpart data (subject to the *was_release* entry in the proc_daemon table).

The nightobs associated with these CS products will match their L1 equivalents in 3.1.3 and L2 equivalents in 3.1.5.

3.1.7 Open time survey specification files

The open time survey specification is a text file made available to WAS by CASU only following validation of the file by the WASP. The ING provide CASU with a description of what Open Time surveys must be created for the forthcoming trimester[1] of observation, including information about the time allocated, and under which TAC codes. This file is transferred by logging into the WASP, navigating to "My Survey" and downloading the relevant data file. WASP sends an email to the WAS with the direct link to the file. Every open time program will be treated by WAS as if it was a survey program belonging to one specific user. The files provided are in CSV format with the following data content in the order as shown in the table:

Table 6 – content of the Open Time survey specification file

Entry	Comment	Example
targsrvy	The name of the survey. This is adopted as the username for OR and WASP access, and is critical under WAS for determining user access rights. Encoded within this entry is the start trimester.	"WS2023A1-003"

[1] In reality, ING offer Open Time observations (in which we include ITP) across two trimesters (although ITP time can be awarded across *four* trimesters). We nevertheless tie survey definition to the initial trimester and permit a carryover to the following trimester(s) where directed to by the data in this file.

	Please refer to [RD30] for the definition of this field.	
trimester_end	This is the final trimester the targsrvy can observe under. If this is the same as that inferred from targsrvy, then this is not a "carryover" survey.	"2023A2"
surname	The surname of the PI awarded time	"Dalton"
forename	The forename of the PI awarded time	Gavin
email	The email address of the PI (used for account creation, notifications, etc)	"gavin.dalton@stfc.ac.uk"
proposal_id	The identifier of the proposal submitted to ING for WEAVE time	"SW2023b4"
wv_access	Boolean to describe whether the access to data from this programme should be granted to the WEAVE Survey users	True
tacalloc	The TAC allocation code. This encodes the time provided to the programme, and from what source. See [RD31] for the definition.	"WS2023A2I0020"
tac_ids	TAC identifiers used for award of time to this programme. This can be a comma-separated list with no spaces between the elements. See [RD31] for more details.	"ITP01,C07"

3.1.8 Data transfer to WAS

It is mandatory that the WAS receive the data of a specific night once the night is complete (i.e. CASU will make available the data of a specific night only when all L1, L2 and CS products are available). The WAS does not process nights where the data set is incomplete. Nevertheless, release of data products from CASU to WAS will be on the basis of the *was_release* entry in the proc_daemon table. Operationally this is set to "nightly" (i.e. with release of products only once all are received for a night).

3.2 L1 Output FITS Files: L1 Products and L1 Superproducts

3.2.1 Types of L1 Output FITS Files

Table 7 defines the different types of L1-output FITS files produced by CPS, that is, the L1 products and the L1 superproducts. The suffix for each file is the run number, as defined in [RD4] by the RUN value in the primary header. For L1 products that stack multiple exposures, by convention we use the first (i.e. lowest) run number as the suffix (noted as <RUN0>), but full provenance information is available in the header.

Table 7 – Types of L1 output FITS files

Type(s)	Filename	Comment
single	single_<RUN>.fit	A 2D image containing processed spectra from all objects in a single exposure.

		Each spectrum occupies a row in the image. All spectra are L1 processed with ancillary information in the same FITS container.
stack	stack_<RUN0>.fit	Stacked spectra of all objects from all exposures of a given instrumental setup within a given OB. Extensions are as for **single**.
superstack	superstack_<RUN0>.fit	As with stack, but across OBs duplicated (and observed with the same fiber configuration). Extensions are as for **single**.
stackcube	stackcube_<RUN0>.fit	All spectra (excluding sky bundles) from all exposures within an OB of LIFU observations resampled onto an equatorial coordinate grid to form a cube.
supercube	supercube_<RUN0>.fit	LIFU analogue of superstack, comprising exposures across multiple duplicate OBs (observed with the same initial pointing, dither pattern, etc.). Extensions are as for **stackcube**.
stackcubelet	stackcubelet_<RUN0>_<ID>.fit <ID> denotes the mIFU bundle, ranging from 01 to 20 (note that two digits are always used)	mIFU analogue of superstack, a series of cubes for each mIFU bundle observed within an OB. The ID denotes which bundle the cubelet refers to and corresponds to the equivalent CCNAMEn (n=1–20) in the primary header of the constituent exposures. Extensions as for **stackcube**.
supercubelet	supercubelet_<RUN0>_<ID>.fit	As with stackcubelet, but for each mIFU bundle observed across multiple duplicated OBs (observed with the same fiber configuration). Extensions are as for **stackcube**.
1-d supertarget LIFU supertarget mIFU supertarget	<OBSMODE><CNAME>_ <CFG>.fit <OBSMODE> comprises: • Blank [MOS] • L [LIFU] • m [mIFU] <CFG> comprises: • B/G[2]/R [Blue / Green / Red] • L/H [resolution] • 1/2/4 [spectral binning]	The most generalised case of spectral stacking, providing fibre configuration-independent stacking of target (MOS) or pointing (IFU) observations sharing common instrument configuration. Generation of IFU-based supertargets depends critically on the size of offsets between IFU exposures as the size of the (LIFU) output cubes

[2] The value 'G' for the camera is used to differentiate between the two high-resolution blue settings.

	Examples: • <CNAME>_BL1.fit • L<CCNAME0>_BL1.fit • m<CCNAMEn0>_BL1.fit	can become unwieldy [RD29]. Mosaics are not supported, so constraints are made on pointing tolerances for generating IFU supertargets.

3.2.1.1 Classification of the Types of L1 Output Files

As it has been described on the previous section, there are 10 types of L1 output files. To better understand this zoo of types of L1 output files and show their properties and relationships visually, as well as to give a name to some sets of these types, they have been classified in Table 8.

Table 8 – Classification the types of L1 output files

		Generic (MOS and IFU)	IFU (Only IFU)	
Products	**Exposure**	single	**LIFU**	**mIFU**
	OB (1 OB)	stack	stackcube	stackcubelet
Superproducts	**Intra-OB (duplicated OBs)**	superstack	supercube	supercubelet
	Inter-OB (different OBs)	1-d supertarget	LIFU supertarget	mIFU supertarget

3.2.1.2 When Is Each Type of L1 Output File Created?

The philosophy for creating an output file is to keep the total number of files to the minimum, that is, to avoid redundant files which will not contribute with new information.

For files with OBSTYPE equal to TARGET, this philosophy is translated in the following criteria:

1. Single files will be created for each raw file.
2. Stack files will be created depending on the FPMODE keyword:
   a. *MOS modes:* When an OB contains more than one target exposure.
   b. *IFU modes:* They will not be generated.
3. Stackcube/Stackcubelet files will be created for each OB containing LIFU/mIFU target exposures.
4. Superstack files will be created depending on the FPMODE keyword:
   a. *MOS modes:* When an OB has been duplicated.
   b. *IFU modes:* They will not be generated.

5. Supercube/Supercubelet files will be created for duplicated OBs containing LIFU/mIFU target exposures.
6. 1-d supertarget files will be created for objects with TARGUSE equal to T or C and which has been observed in a MOS mode in several non-duplicated OBs with the same instrumental setup.
7. LIFU/mIFU supertarget files will be created for IFU objects that have been observed in the LIFU/mIFU mode in several non-duplicated OBs with the same instrumental setup (see [RD30] for information on how OBs should be linked together).

Table 9 shows a summary reflecting the above criteria:

Table 9 – Creation of the L1 output files depending on mode (and dithering)

Mode [*and dithering*]	**L1 Products (exposure and OB level)**	**L1 Superproducts (> 1 OB level)**
MOS	single, stack	superstack, 1-d supertarget
LIFU [*dithered*]	single, stackcube	supercube, LIFU supertarget
LIFU [*un-dithered*]	single, stackcube	supercube, LIFU supertarget
mIFU [*dithered*]	single, stackcubelet	supercubelet, mIFU supertarget
mIFU [*un-dithered*]	single, stackcubelet	supercubelet, mIFU supertarget

In the case of files with OBSTYPE equal to FLUX_STD or RV_STD, only single files will be created.

3.2.2 Extensions of the L1 Output Files

Each type of L1 output file will contain several extensions. All the potential extensions for a file are described in Table 10, while Table 11 specifies for which types of files these extensions will be included. On both tables, when <arm> is used for referring to extension names, it means that its value can be either 'BLUE' or 'RED' depending on from which arm of the spectrograph the spectra originate.

Table 10 – Potential extensions of the L1 output files

Extension Name	**Description**
Primary	This is the primary header unit. There will be no data in this HDU. • *Singles:* The header will have all the general information about the observation inherited from the raw file. • *OB, intra-OB and supertarget files:* The header will have all the general information about the observation inherited from the raw file with the lowest run number.
<arm>_DATA	Final wavelength calibrated, sky subtracted and heliocentric corrected spectra. • *Generic files (unless 1-d supertargets):*

	Each spectrum is a row in the image and the number of rows is the number of fibres that were used to observe both objects and sky patches. All spectra are on a linear wavelength scale with exactly the same dispersion and wavelength range. • *IFU files:* Spectra are resampled onto a continuous spatial grid. The cube is oriented as (x, y, wavelength) where the WCS is fixed so that this maps into (α, δ, λ). The equatorial coordinates are expressed in decimal degrees. • *1-d supertarget:* The final 1D super-stacked spectrum of the object. The wavelength is in units of Angstrom. The spectral flux is in units of ADU.
<arm>_IVAR	The inverse variance of the spectra in <arm>_DATA extension (similar to a weight map). • *Generic files (unless 1-d supertargets):* Each row is the inverse variance of each spectrum. • *IFU files:* The inverse variance of the cube. • *1-d supertarget:* The inverse variance of the spectrum. Bad pixels will be flagged with an inverse variance of zero. The wavelength is in units of Angstrom. The spectral flux is in units of ADU^{-2}.
<arm>_DATA_NOSS	Final wavelength calibrated, non-sky-subtracted and heliocentric corrected spectra. These are only included as a means of assessing the quality of the sky correction. It also allows for an alternative treatment of the background subtraction by the user. • *Generic files (unless 1-d supertargets):* The same spectra as in the <arm>_DATA extension, but before sky subtraction. • *IFU files:* The same cube as in the <arm>_DATA extension extension, but before sky subtraction. • *1-d supertarget:* The final 1D super-stacked spectrum of the non-skysubtracted object. The wavelength is in units of Angstrom. The spectral flux is in units of ADU.
<arm>_IVAR_NOSS	The inverse variance of the <arm>_DATA_NOSS extension. Again, bad pixels will be flagged with an inverse variance of zero. The wavelength is in units of Angstrom. The spectral flux is in units of ADU^{-2}.

<arm>_SENSFUNC	The sensitivity function(s) for the spectra. • *Generic files (unless 1-d supertargets):* A 2D image where each row is the sensitivity function for each of the spectra. • *IFU files:* A single vector of flux versus wavelength. • *1-d supertarget:* A single vector of flux versus wavelength. They will be presented such that if they are multiplied by the spectra at <arm>_DATA extension, then fully flux calibrated spectra will result (for IFU files, the single vector can be multiplied into every spectrum in the cube). The wavelength is in units of Angstrom. The spectral flux is in units of erg/(s cm^2 Å ADU).
FIBTABLE	A binary FITS table with information about each fibre. • *Generic files (unless 1-d supertargets):* A fibre table as described in Table 4 and Table 16. • *1-d supertarget:* A pared down version of the fibre table described in Table 4 and Table 16. Much of the information relating to the individual observation will have been removed. A description of the fibre table is given in Table 19.
<arm>_DATA_COLLAPSE3	A 2D image representing the cube in <arm>_DATA extension when it has been collapsed along the spectral axis (it will only exist in the *IFU files*).
<arm>_ IVAR_COLLAPSE3	A 2D image representing the cube in <arm>_IVAR extension when it has been collapsed along the spectral axis (it will only exist in the *IFU files*).
PROVENANCE	A binary FITS table extension with information about each of the spectra that were combined to create the current spectrum. A description of this table is given in Table 20. Its columns will be slightly different depending on the kind of *supertarget*.

Table 11 – Types of L1 extensions and files which will contain those extensions

Extn Type	Ext Names	L1 Output Files
Basic extensions	Primary <arm>_DATA <arm>_IVAR <arm>_DATA_NOSS <arm>_IVAR_NOSS <arm>_SENSFUNC	All files: *single, stack, stackcube, stackcubelet, superstack, supercube, supercubelet, 1-d supertarget, LIFU supertarget and mIFU supertarget*
Fibinfo table extension	FIBTABLE	Generic files: *single, stack, superstack and 1-d supertarget*
Collapse extensions	<arm>_DATA_COLLAPSE3 <arm>_ IVAR_COLLAPSE3	IFU files: *stackcube, stackcubelet, supercube, supercubelet, LIFU supertarget and mIFU supertarget*
Provenance extension	PROVENANCE	Supertarget files: *1-d supertarget, LIFU supertarget and mIFU supertarget*

3.2.2.1 Contents of the header of each extension

3.2.2.1.1 Keywords contained in the primary header

The primary header of each L1 output file will contain all the keywords from the primary header of the file with the lowest run number used to build it. Furthermore, the keywords defined in Table 12 will be added to the primary header.

Table 12 – Keywords added to the primary headers of the L1 Output Files

Keyword(s)	Type	Description
CASUDATE	char	Data processing date (e.g. “2021-10-15T15:15:57”)
CASUVERS	char	Release version (e.g. “3.0”).
SOFTVERS	char	WEAVEDR software version (e.g. “1.2” or “02e21ca6”).
SOFTDATE	char	[yyyy-mm-dd] WEAVEDR software release.
SOFTAUTH	char	WEAVEDR author, including contact for bug reports (“CASU <casuhelp@ast.cam.ac.uk>”).
SOFTINST	char	CASU URL (“CASU <http://casu.ast.cam.ac.uk>”).

SRVY*n*	char	The name of the surveys with objects in the file. The number of these keywords depends on the number of surveys in each file.
CALDATE	int	[yyyymmdd] Master calibration library used
PROFILES	char	Profile PSF file used.
OBTRFILE	char	OB fibre trace file used.
WAVEFILE	char	File with wavelength solution.
OBWVFILE	char	OB wavelength solution file.
FFLATCOR	char	Normalised fibre flat used.
OBFBFLAT	char	OB fibre flat.
TELLCORR	char	Normalised telluric correction used.
SENSFILE	char	Pointer to master sensfunc file that was scaled.
WARCFILE	char	Warc file associated to this file.
NSAT	int	Total number of saturated pixels in the arm (i.e. from both detectors).
SKYSUB	bool	Spectra are sky subtracted.
PCASUB	bool	PCA correction applied.
PCANUM	int	Number of PCA eigenvectors used.
PCATOTAL	real	Percentage sky residual variation accounted for.
NCALIBRT	int	Number of calibrators used.
DELMAGZP	real	[mag] Mag offset applied as sensfunc scale factor.
DELZPGRA	real	Mag offset applied as sensfunc scale factor
EXTCORR	bool	Gaia DR3 extinction correction applied.
MINSNR	real	Minimum S/N ratio.
SPCMINSN	int	Spectrum with min S/N ratio.
FIBMINSN	int	Fibre with min S/N ratio.
MAXSNR	real	Maximum S/N ratio.
SPCMAXSN	int	Spectrum with max S/N ratio.
FIBMAXSN	int	Fibre with max S/N ratio.
MEDSNR	real	Median S/N ratio.
BUNDLEID	int	ID of the mIFU bundle used to build a mIFU stackcubelet or supercubelet. Its value will be 0 in other kind of files.
NCOMB	int	Number of combined images (value of 1 for single exposures).
PROV0000	char	Output file name.
PROV*n*[0001-NCOMB]	char	Filename and extension used for the *n*th file that was used in this stack (e.g. ‘single_2000073.fit[BLUE_DATA]’).

The values of some of inherited keywords (**DATAMVER**, **EXPTIME**, **CHECKSUM** and **DATASUM**) will be properly updated (if needed).

It is worth noting that EXPTIME keyword will be updated to reflect the total exposure time, but other keywords like EXPOSED, ELAPSED, DARKTIME or REQTIME will be simply inherited from the single file with the lowest run number.

The values for the keywords **CASUVERS**, **SOFTVERS**, **SOFTDATE**, **SOFTAUTH**, **SOFTINST, MINSNR, SPCMINSN, FIBMINSN, MAXSNR, SPCMAXSN, FIBMAXSN, MEDSNR, NCOMB, PROV0000** and **PROV*n*** will be updated in each L1 output file.

The values for the keywords **SRVY*n*, PROFILES, WAVEFILE, FFLATCOR, OBFBFLAT, TELLCORR, SENSFILE, NSAT, SKYSUB, PCASUB, PCANUM, PCATOTAL, NCALIBRT, DELMAGZP** and **EXTCORR** will be computed in the single files. The other L1 output files will inherit these keywords from the single file with the lowest run number.

3.2.2.1.2 Keywords contained in the image extensions

With the exception of <arm>_SENSFUNC, the extensions containing images:

- <arm>_DATA
- <arm>_IVAR
- <arm>_DATA_NOSS
- <arm>_IVAR_NOSS
- <arm>_DATA_COLLAPSE3
- <arm>_ IVAR_COLLAPSE3

will contain all the keywords from the extension <detector1>_DATA of the file with the lowest run number used to build it, except a few keywords which will be removed and which are listed below (for example, WCS or duplicated keywords). Furthermore, the keywords defined in Table 13 will be added to the header.

The values of some of inherited keywords (**BITPIX, NAXIS, NAXIS*n*, CRVAL*n*, CRPIX*n*, CUNIT*n*, CD*n*_*n*, EXTNAME**, **CHECKSUM** and **DATASUM**, where the index *n* ranges from 1 to NAXIS) will be properly updated. Obviously, when NAXIS takes the values 1/3 in the L1 output file, this will imply the removal/addition of keywords depending on the index *n*.

The removed keywords from the extension <detector1>_DATA are the following:

- BZERO and BSCALE (as these do not apply to the L1 products).
- PROJP1, PROJP3, PV1_1, PV1_2, PV2_1 and PV2_2 (as they are useless WCS keywords in the WEAVE context).
- DATE-OBS, UTSTART, EXPOSED, EXPTIME, ELAPSED and DARKTIME (as they are duplicated keywords which appear in the primary header).
- BIASSEC and TRIMSEC.

Table 13 – Keywords to be added to the image extensions of the L1 Output Files

Keyword(s)	Type	Description
CTYPE*n*[1-NAXIS]	char	Type for the intermediate-coordinate axis *n*.

BUNIT	char	Physical units of the quantities in the array.

Unlike the above extensions the <arm>_SENSFUNC HDU does not contain data derived from the original raw image extensions. The keywords within this header are shown below in Table 14.

Table 14 – Keywords in the SENSFUNC HDU

Keyword(s)	**Type**	**Description**
BITPIX	int	Describes number of bits per pixel (-32)
NAXIS	int	Number of data axes (2)
NAXIS*n*[1-NAXIS]	int	Length of data axis *n*
PCOUNT	int	0
GCOUNT	int	1
CTYPE*n*[1-NAXIS]	char	Type of data represented on axis *n* ('PIXEL','NSPEC')
CRPIX*n*[1-NAXIS]	real	Pixel zeropoint for axis *n*
CRVAL*n*[1-NAXIS]	real	Value of axis *n* at zeropoint
CD_n_m[1-NAXIS]	real	Coordinate translation matrix
EXTNAME	char	Name of the HDU ('<ARM_SENSFUNC>')
BUNIT	char	Physical units of the array ('erg/(s*cm^2*Angstrom*adu)')

3.2.2.1.3 Keywords contained in the table extensions

The keywords contained in the table extensions (i.e. FIBTABLE and PROVENANCE extensions) will be the following:

- Basic keywords to related with the contents of the table: XTENSION, BITPIX, NAXIS, NAXIS1, NAXIS2, PCOUNT, GCOUNT, TFIELDS, EXTNAME, CHECKSUM and DATASUM.
- Keywords with the definition of each column: TTYPE*n*, TFORM*n*, TUNIT*n*, and TNULL*n*.

The columns to be included in each table extension depends on the files and have been described in Section 3.2.3.

It is worth noting that TDIM*n* keywords would be present in the raw files, they would not be inherited in the L1 output files, because they would be simply redundant to the lenghts provided via TFORM*n* keywords.

3.2.3 Structure of Each Type of L1 Output FITS Files

The structure of the each one of the different output FITS files produced by CPS pipeline will be described on the following subsections.

3.2.3.1 Structure of the Single Files

The extensions of the single files are described in Table 15.

Table 15 – Extensions of the single, stack and superstack files from the CPS pipeline

Extn #	Extn Name
0	Primary
1	<arm>_DATA
2	<arm>_IVAR
3	<arm>_DATA_NOSS
4	<arm>_IVAR_NOSS
5	<arm>_SENSFUNC
6	FIBTABLE

The contents of the FIBTABLE extension are inherited from the raw files (described in Table 4 and [RD04]) plus a new set of columns added by the CPS pipeline and which are described in Table 16. The **Nspec** column has been highlighted because it will be added before the inherited columns, while the other ones are added afterwards in the same order as they are enumerated in the table.

Table 16 – New columns added to those from the raw fibinfo table in the single, stack and superstack files

Column name (TTYPE)	Format (TFORM)	Unit (TUNIT)	Null value (TNULL)	Description
Nspec	1I		-1	The number of the spectrum.
RMS_arc1	1D	Angstrom		The RMS of the wavelength solution for the first detector.
RMS_arc2	1D	Angstrom		The RMS of the wavelength solution for the second detector.
Resol	1E	Angstrom		The mean FWHM of the arc lines for this fibre.
Helio_cor	1E	km/s		The heliocentric correction.
Wave_cor1	1D	Angstrom		The wavelength offset applied to the standard arc solution for the first detector.
Wave_corrms1	1D	Angstrom		The RMS of the wavelength offset correction for the first detector.
Wave_cor2	1D	Angstrom		The wavelength offset applied to the standard arc

				solution for the second detector.
Wave_corrms2	1D	Angstrom		The RMS of the wavelength offset correction for the second detector.
Skyline_off1	1D	Angstrom		The offset between selected sky lines in the first spectrum and their known wavelengths.
Skyline_rms1	1D	Angstrom		The RMS of the above.
Skyline_off2	1D	Angstrom		The offset between selected sky lines in the second spectrum and their known wavelengths.
Skyline_rms2	1D	Angstrom		The RMS of the above.
Sky_shift	1E	pixel		An offset to align the mean sky wavelength with the current spectrum during the background correction phase.
Sky_scale	1E			The scaling factor applied to the mean sky before subtracting from the current spectrum.
Exptime	1E	s		The total exposure time for the object.
SNR	1E			The mean signal/noise ratio for the spectrum.
Meanflux_g	1E	adu		The mean flux in the SDSS-like g band.
Meanflux_r	1E	adu		The mean flux in the SDSS-like r band.
Meanflux_i	1E	adu		The mean flux in the SDSS-like i band.
Meanflux_gg	1E	adu		The mean flux in the Gaia G band.
Meanflux_bp	1E	adu		The mean flux in the Gaia BP band.
Meanflux_rp	1E	adu		The mean flux in the Gaia RP band.

3.2.3.2 Structure of the Stack and Superstack Files

Stack and superstack files contain the same extensions than single files (i.e. those described in Table 15).

The contents of the FIBTABLE extension are inherited from the singles, and therefore they contain the columns of the raw files (described in Table 4) plus the columns added by the CPS pipeline (described in Table 16). All the values contained in the FIBTABLE extension will be simply copied from the file with lowest run number, unless for the columns **Exptime, SNR, Meanflux_g, Meanflux_r, Meanflux_i, Meanflux_gg, Meanflux_bp** and **Meanflux_rp**, which will be properly updated.

3.2.3.3 Structure of the Stackcube, Stackcubelet, Supercube and Supercubelet Files

The extensions of the stackcube files are described in Table 17.

Table 17 – Extensions of the stackcube, stackcubelet, supercube and supercubelet files from the CPS pipeline

Extn #	Extn Name
0	Primary
1	<arm>_DATA
2	<arm>_IVAR
3	<arm>_DATA_NOSS
4	<arm>_IVAR_NOSS
5	<arm>_SENSFUNC
6	<arm>_DATA_COLLAPSE3
7	<arm>_IVAR_COLLAPSE3

It is worth noting that in this case the data stored in the extension <arm>_SENSFUNC are a single vector (instead of 2D image).

It is important to note that with cube products there will be no fibre table as the concept of spectra coming from an individual fibre is lost once the spectra are resampled onto the spatial grid.

The Primary header of these procducts are as described in [Table 10], in addition to the following keywords:

[table]

These are extracted from the source FIBINFO table – under the stacking scheme used to combine IFU observations, we require that exposures (or cubelets within these exposures) adopt the same [TARGNAME;xx]

3.2.3.4 Structure of the 1-d [MOS] Supertarget Files

The extensions of the 1-d MOS supertarget files are described in Table 18.

Table 18 – Extensions of the 1-d supertarget files from the CPS pipeline

Extn #	Extn Name
0	Primary
1	<arm>_DATA

2	<arm>_IVAR
3	<arm>_DATA_NOSS
4	<arm>_IVAR_NOSS
5	<arm> _SENSFUNC
6	FIBTABLE
7	PROVENANCE

It is worth noting that in this case the data stored in the image extensions are single vectors (instead of 2D images).

Its FIBTABLE extension will contain a pared down version (described in Table 19) and it will have only one row. All the values contained in the FIBTABLE extension will be simply copied from the file with lowest run number, except for the columns **Exptime, SNR, Meanflux_g, Meanflux_r, Meanflux_i, Meanflux_gg, Meanflux_bp** and **Meanflux_rp**, which will be properly updated.

Table 19 – The revised fibre table for 1-d supertarget output files. The descriptions of these columns can be found in Table 4 and Table 16

#	**Column name (TTYPE)**	**Format (TFORM)**	**Unit (TUNIT)**
1	CNAME	20A	
2	TARGID	30A	
3	TARGNAME	30A	
4	TARGRA	1D	deg
5	TARGDEC	1D	deg
6	TARGEPOCH	1E	yr
7	TARGCAT	30A	
8	TARGPMRA	1E	mas/yr
9	TARGPMDEC	1E	mas/yr
10	TARGPARAL	1E	mas
11	MAG_G	1E	mag
12	EMAG_G	1E	mag
13	MAG_R	1E	mag
14	EMAG_R	1E	mag
15	MAG_I	1E	mag
16	EMAG_I	1E	mag
17	MAG_GG	1E	mag
18	EMAG_GG	1E	mag
19	MAG_BP	1E	mag
20	EMAG_BP	1E	mag
21	MAG_RP	1E	mag
22	EMAG_RP	1E	mag
23	SNR	1E	
24	SNR_expected[3]	1E	

[3] This keyword is not described in the previous tables. It contains that expected SNR computed from the SNR of the spectra used to built the supertarget spectrum.

25	Meanflux_g	1E	adu
26	Meanflux_r	1E	adu
27	Meanflux_i	1E	adu
28	Meanflux_gg	1E	adu
29	Meanflux_bp	1E	adu
30	Meanflux_rp	1E	adu

The contents of its PROVENANCE table are described in Table 20.

Table 20 – Columns for the PROVENANCE table in the supertarget output files

Type of supertarget	**Column name (TTYPE)**	**Format (TFORM)**	**Unit (TUNIT)**	**Null value (TNULL)**	**Description**
All	ninput	1I		-1	An integer counter for the input spectra.
All	filename	36A			The file name of the individual exposure from where this input spectrum was taken.
1-d	nspec	1I		-1	The index of the spectrum from the original OB exposure file.
1-d	fibreid	1I		-1	The fibre number through which this spectrum was observed.
LIFU mIFU	ccname	20A			CNAME of the central fibre.
mIFU	mifuid	1I		-1	Identifier of the mIFU bundle.
All	obid	1J		-99	The OB id number for the exposure.
All	mjd_obs	1D	d		The MJD at the start of the exposure.
All	date_obs	10A			[yyyy-mm-dd] The date of the observation.
All	ut	12A			[hh:mm:ss.sss] The universal time at the start of the exposure.
All	st	12A			[hh:mm:ss.sss] The local sidereal time at the start of the exposure.
All	exptime	1E	s		The exposure time for this observation.
All	amstart	1E			The airmass at the start of the exposure.
All	amend	1E			The airmass at the end of the exposure.
All	agxsee	1E	arcsec		Average value of the mean FWHM of guide stars in the *x*-axis.
All	agysee	1E	arcsec		Average value of the mean FWHM of guide stars in the *y*-axis.

All	snr	1E			The estimated mean signal/noise ratio of the spectrum.
All	weight	1E			Weight used for this component of the stack.

3.2.3.5 Structure of the LIFU and mIFU Supertarget Files

LIFU and mIFU supertarget files will contain the extensions described in Table 21.

Table 21 – Extensions of the LIFU and mIFU supertarget files from the CPS pipeline

Extn #	Extn Name
0	Primary
1	<arm>_DATA
2	<arm>_IVAR
3	<arm>_DATA_NOSS
4	<arm>_IVAR_NOSS
5	<arm> _SENSFUNC
6	<arm>_DATA_COLLAPSE3
7	<arm>_IVAR_COLLAPSE3
8	PROVENANCE

Their PROVENANCE tables will be as described in Table 20. It is worth noting that they will be slightly different.

3.3 L1 Calibration Files

The only kind of L1 calibration files which will be shared with other SPA nodes by CPS are the warc files.

3.3.1 warc Files

warc files are rebinned and wavelength-calibrated files built from arc images. They could be used to study the line spread function. CASU will provide the master-calibration versions of these files as opposed to the OB-level warcs. The latter will have higher noise and should not be used to infer the LSF.

The convention to assign filenames for warc files is not still decided. Nevertheless, it will reflect the same information as in the following convention:

warc_<MONTH>_<FPMODE>-<CFG>.fit

where <FPMODE> comprises:

- A [MOS-A]
- B [MOS-B]
- L [LIFU]
- m [mIFU]

and <CFG> comprises:

- B/G/R [Blue / Green / Red]
- L/H [resolution]

- 1/2/4 [spectral binning]

The structure of the warc files is shown in Table 22:

Table 22 – Extensions of the warc files from the CPS pipeline

Extn #	Extn Name	Description
0	Primary	This is the primary header unit. It will have all the general information about the observation inherited from the raw file.
1	<detector1>_DATA	Wavelength calibrated and rebinned spectra for detector 1. Each spectrum is a row in the image. All spectra are on a linear wavelength scale with exactly the same dispersion and wavelength range. The wavelength is in units of Angstrom. The spectral flux is in units of ADU.
2	<detector1>_DATA_var	The variance of the spectra in <detector1>_DATA. Each row is the inverse variance of each spectrum. The wavelength is in units of Angstrom. The spectral flux is in units of ADU^2.
3	<detector2>_DATA	As <detector1>_DATA, but for detector 2.
4	<detector3>_DATA_var	As <detector1>_DATA_var, but for detector 2.
5	FIBTABLE	A binary FITS table with information about each fibre inherited from the raw file.

The primary header of the warc files will contain the keywords inherited from the raw files plus **CASUVERS**, **SOFTVERS**, **SOFTDATE**, **SOFTAUTH**, **SOFTINST**, **PROFILES**, **WAVEFILE**, **FFLATCOR** and **OBFBFLAT**. The definitions of these keywords are available in Table 12.

The image extensions (<detector1>_DATA, <detector1>_DATA_var, <detector2>_DATA and <detector2>_DATA_var) will contain the keywords inherited from the raw files plus the keywords listed in Table 13.

The fibre table extension (FIBTABLE) will be simply inherited from the raw file.

Also, the following inherited keywords will be properly updated (if needed): **DATAMVER**, **CHECKSUM** and **DATASUM**.

4 QUICK-LOOK

The quick-look facility will consist of three separate parts:

- A quick-look processing program that will scan the raw data directory for new data files. When it finds something, it will do a quick-look reduction of the file and place the resulting processed image in the processed data directory.
- A QC logging program that will scan the processed data directory for newly processed data files. When a new file appears, it will summarise the QC information in the file's headers and fibre table and will append this information to a database FITS file described in section 4.1. This will ultimately be ingested into the OCS QC database.
- An interactive display program for observers that will scan the reduced data directory for processed data files. The list of processed data files will be presented in a graphical user interface. The user will be able to choose a particular file to examine. The GUI will allow the user to display an image, plot spectra and retrieve information from the chosen file's fibre table regarding individual fibres and the objects they covered. Summary QC information will also be accessible. An example of what the GUI will resemble is given in Section 4.2.

The quick-look processing software in no way interacts with either the QC logging or the interactive display. The latter two only react to the presence of files containing processed data.

4.1 QL → OCS

A database consisting of a summary of results from the quick-look analysis will be written as the software is running. This will consist of a single FITS file per night, with different binary table extensions for each data type that has been generated and examined by the quick-look pipeline. The primary header will contain information about the date the observations and the analysis were done.

Table 23 below contains the proposed structure and contents of this quick-look database file. Table 24 details the definition of the columns included in this file. Note that the entries will be done on a per-detector basis, hence provided both detectors are alive, there will be two entries in the relevant table for each run number.

It is worth noting that there are not specific extensions for images with OBSTYPEs FLUX_STD and RV_STD. If these kinds of images would be processed, they would be included in the extension TARGET.

Table 23 – Columns contained in each binary table extension of the quick-look database FITS files. There will be a table extension for each observation type.

	Extension names								
Columns	BIAS	DUMMY_BIAS	DARK	DETECTOR_FLAT	SALSA_ARC	SALSA_FLAT	ARC	FIBRE_FLAT	TARGET
runno	✓	✓	✓	✓	✓	✓	✓	✓	✓
detno	✓	✓	✓	✓	✓	✓	✓	✓	✓
detname	✓	✓	✓	✓	✓	✓	✓	✓	✓
mjd_obs	✓	✓	✓	✓	✓	✓	✓	✓	✓
date_obs	✓	✓	✓	✓	✓	✓	✓	✓	✓
ut	✓	✓	✓	✓	✓	✓	✓	✓	✓
exptime			✓	✓	✓	✓	✓	✓	✓
arm	✓	✓	✓	✓	✓	✓	✓	✓	✓
ccdxbin	✓	✓	✓	✓	✓	✓	✓	✓	✓
ccdybin	✓	✓	✓	✓	✓	✓	✓	✓	✓
grating					✓	✓	✓	✓	✓
plate					✓	✓	✓	✓	✓
fpmode					✓	✓	✓	✓	✓
salsa_i					✓	✓			
nsalsa					✓	✓			
flat_source				✓		✓		✓	
arc_source					✓		✓		
obtitle	✓	✓	✓	✓	✓	✓	✓	✓	✓
obid	✓	✓	✓	✓	✓	✓	✓	✓	✓
cat_name									✓
cat_ra									✓
cat_dec									✓
airmass									✓
nsat	✓	✓	✓	✓	✓	✓	✓	✓	✓
medos_1	✓	✓	✓	✓	✓	✓	✓	✓	✓
rmsos_1	✓	✓	✓	✓	✓	✓	✓	✓	✓
medos_2	✓	✓	✓	✓	✓	✓	✓	✓	✓
rmsos_2	✓	✓	✓	✓	✓	✓	✓	✓	✓
medos_3	✓	✓	✓	✓	✓	✓	✓	✓	✓

rmsos_3	✓	✓	✓	✓	✓	✓	✓	✓	✓
medos_4	✓	✓	✓	✓	✓	✓	✓	✓	✓
rmsos_4	✓	✓	✓	✓	✓	✓	✓	✓	✓
medbias_1	✓	✓							
rmsbias_1	✓	✓							
medbias_2	✓	✓							
rmsbias_2	✓	✓							
rmsbias_3	✓	✓							
medbias_4	✓	✓							
rmsbias_4	✓	✓							
meddark			✓						
rmsdark			✓						
medscatter						✓		✓	✓
nspectra						✓		✓	✓
nmissing						✓		✓	✓
ntarget									✓
nsky									✓
ncalib									✓
nlines_fwhm					✓		✓		✓
fwhm_spa					✓	✓		✓	
rms_spa					✓	✓		✓	
fwhm_spec					✓		✓		✓
rms_spec					✓		✓		
testloc						✓			
nlines_median							✓		
nlines_min							✓		
spec_nlines_min							✓		
nlines_max							✓		
spec_nlines_max							✓		
fit_rms_median							✓		
fit_rms_min							✓		
spec_rms_min							✓		
fit_rms_max							✓		
spec_rms_max							✓		
medflux				✓				✓	
rmsflux				✓				✓	
minflux								✓	
fib_minflux								✓	
spec_minflux								✓	
maxflux								✓	

fib_maxflux							✓	
spec_maxflux							✓	
snrmed								✓
snrmin								✓
fib_snrmin								✓
spec_snrmin								✓
snrmax								✓
fib_snrmax								✓
spec_snrmax								✓
skyflx_g								✓
skyflx_r								✓
skyflx_i								✓

Table 24 – Definition of columns contained in the quick-look database FITS files

TTYPE	TFORM	TUNIT	TNULL	Description
runno	J		-1	The run number of the observation.
detno	I		-1	The number of the detector (1 or 2).
detname	13A			The name of the detector.
mjd_obs	D	d		The MJD of the observation.
date_obs	10A			The observation date.
ut	12A			The UT of the observation.
exptime	E	s		The exposure time of the current run.
arm	4A			The spectrograph arm ('BLUE' or 'RED').
ccdxbin	I		-1	The binning factor in the x direction.
ccdybin	I		-1	The binning factor in the y direction.
grating	8A			The grating used ('N/A', 'LowRes', 'HighRes1' or 'HighRes2').
plate	7A			The illuminated plate ('PLATE_A', 'PLATE_B' or 'LIFU').
fpmode	5A			The module being used ('MOS-A', 'MOS-B', 'mIFU' or 'LIFU').
salsa_i	I		-1	The number of the current exposure in the salsa sequence.

nsalsa	I		-1	The number of exposures in the salsa sequence.
flat_source	11A			The name of the light source for the flat.
arc_source	11A			The name of the arc lamp.
obtitle	68A			The name of the OB.
obid	J		-99	The ID number of the current OB.
cat_name	68A			The catalogue name for exposure.
cat_ra	12A			The central RA for exposure.
cat_dec	12A			The central Dec for exposure.
airmass	E			The airmass of the observation.
nsat	J		-1	Number of saturated pixels in image.
medos_1	E	adu		Median bias in overscan region 1.
rmsos_1	E	adu		RMS bias in overscan region 1.
medos_2	E	adu		Median bias in overscan region 2.
rmsos_2	E	adu		RMS bias in overscan region 2.
medos_3	E	adu		Median bias in overscan region 3.
rmsos_3	E	adu		RMS bias in overscan region 3.
medos_4	E	adu		Median bias in overscan region 4.
rmsos_4	E	adu		RMS bias in overscan region 4.
medbias_1	E	adu		Median bias in quadrant 1.
rmsbias_1	E	adu		RMS bias in quadrant 1.
medbias_2	E	adu		Median bias in quadrant 2.
rmsbias_2	E	adu		RMS bias in quadrant 2.
medbias_3	E	adu		Median bias in quadrant 3.
rmsbias_3	E	adu		RMS bias in quadrant 3.
medbias_4	E	adu		Median bias in quadrant 4.
rmsbias_4	E	adu		RMS bias in quadrant 4.
meddark	E	adu/(pixel*s)		The median dark current.
rmsdark	E	adu/(pixel*s)		The RMS dark current.
medscatter	E	adu		The median level of scattered light removed.

nspectra	I		-1	Number of detected spectra.
nmissing	I		-1	Number of spectra flagged as missing in fibre table.
ntarget	I		-1	Number of science target spectra.
nsky	I		-1	Number of sky monitor fibres.
ncalib	I		-1	Number of calibration standard fibres.
nlines_fwhm	J		-1	Number of lines analysed in each spectrum for the FWHM measurements.
fwhm_spa	E	pixel		The median FWHM in the spatial axis of the spectra.
rms_spa	E	pixel		The RMS of the spatial FWHM.
fwhm_spec	E	Angstrom		The median spectral FHWM of the lines.
rms_spec	E	Angstrom		The RMS of the spectral FWHM of the lines.
testloc	I		-1	The position in the spectral direction of where the spatial FWHM measurement is taken.
nlines_median	J		-1	The median number of lines used in the wavelength solutions.
nlines_min	J		-1	The minimum number of lines used in a wavelength solution.
spec_nlines_min	I		-1	Number of first spectrum with *nlines_min* lines used in fit.
nlines_max	J		-1	The maximum number of lines used in a wavelength solution.
spec_nlines_max	I		-1	Number of first spectrum with *nlines_max* lines used in fit.
fit_rms_median	E	Angstrom		The median wavelength solution RMS.
fit_rms_min	E	Angstrom		The minimum wavelength solution RMS.
spec_rms_min	I		-1	The number of the spectrum with the

				minimum wavelength solution RMS.
fit_rms_max	E	Angstrom		The maximum wavelength solution RMS.
spec_rms_max	I		-1	The number of the spectrum with the maximum wavelength solution RMS.
medflux	E	adu		The median flux.
rmsflux	E	adu		The RMS flux.
minflux	E	adu		The minimum average flux in a spectrum.
fib_minflux	I		-1	The fibre number of the spectrum with the smallest average flux.
spec_minflux	I		-1	The spectrum number of the spectrum with the smallest average flux.
maxflux	E	adu		The maximum average flux in a spectrum.
fib_maxflux	I		-1	The fibre number of the spectrum with the largest average flux.
spec_maxflux	I		-1	The spectrum number of the spectrum with the largest average flux.
snrmed	E			The median s/n ratio over all target spectra.
snrmin	E			The minimum mean s/n ratio in a target spectrum.
fib_snrmin	I		-1	The fibre with the minimum mean s/n ratio.
spec_snrmin	I		-1	The spectrum number with the minimum mean s/n ratio.
snrmax	E			The maximum mean s/n ratio in a target spectrum.
fib_snrmax	I		-1	The fibre with the maximum mean s/n ratio.
spec_snrmax	I		-1	The spectrum number with the maximum mean s/n ratio.
skyflx_g	E	adu		The median of the mean flux in the sky fibres in the g band.

skyflx_r	E	adu		The median of the mean flux in the sky fibres in the r band.
skyflx_i	E	adu		The median of the mean flux in the sky fibres in the i band.

4.1.1 Data Transfer

When a night of data starts to be processed by the quick-look facility (at >~ 10:00 UTC, just after the processing of the previous night has finished), a new empty quick-look database FITS file is created into a staging area with nightly subdirectories whose names will be in the format YYYYMMDD. In particular, these files will be located at "weaveQL:/data/weaveql/procdata/{YYYYMMDD}/QL_results.fit".

Each time that the QC logging program will detect a new reduced image by the quick-look processing program, the contents of the quick-look database FITS file of the current night will be updated.

The presence of a file called "*.harvest.eon*" in a given nightly directory will flag that the night is processed and its quick-look database FITS file will not be updated again.

The contents of the staging area will be periodically removed, but the quick-look database FITS files of the current and the previous night will not be never deleted.

The quick-look outputs described above are written to a disk that has been NFS-exported at ING, so in principle can be read-only mounted by the machines reported by OCS.

This way of sharing the data will allow polling the quick-look database FITS files in real-time, or at the end of the night, as preferred by the OCS.

4.2 QL → Observers

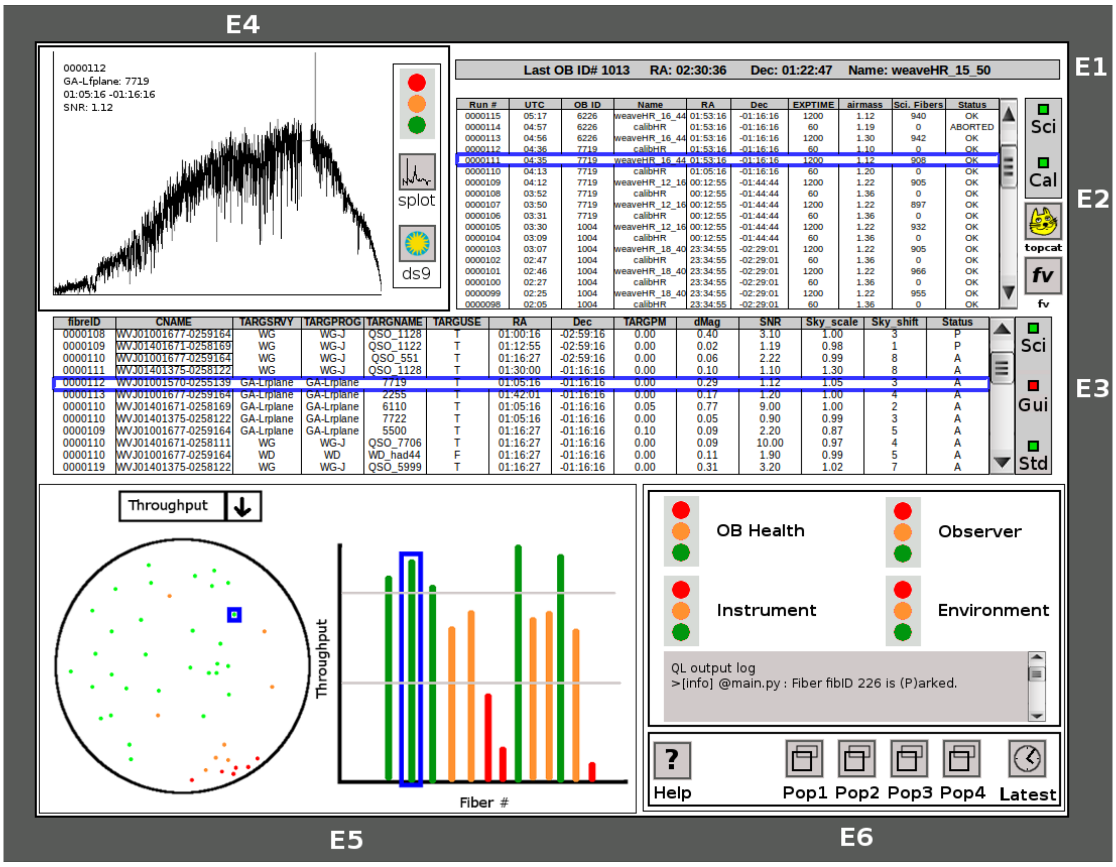
Figure 1 shows a preliminary design layout for the WEAVE QL user interface for use on-summit. Each element of the UI is identified in white and described below:

1. Provides an at-a-glance identification of the most recently observed exposure, including its ID and position.
2. Lists the executed exposures (most recent on top). This table provides useful top-level information about the exposure and can filter out [Sci]ence or [Cal]ibration frames. There are options to launch external tools (e.g. "topcat") to analyse the relevant fibre information tables. On clicking an entry in this table, the fibre table is displayed in element 3.
3. This shows the per-fibre information available for given exposure. This can be filtered to exclude [Sci]ence, [Gui]de or Flux Standard [Std] fibres. The table provides a mixture of information provided before CPS processing (e.g. CNAME) and quantities measured by the QL pipeline (e.g. dMag – the difference between photometric and spectrally-measured broadband

magnitudes). Selecting an entry in this table will trigger a quick-look plot of the relevant 1D spectrum.

4. This displays the spectrum that has been selected in element 3, along with information distilled from the fibre table. This spectrum can be inspected in greater detail by launching "splot" or "ds9". The overall "health" of the spectrum, specifically how well the scientific requirements are met, are encapsulated in the "traffic light" indicator. This indicator will highlight when a spectrum is non-optimal and tool-tips will indicate which requirements are not met.
5. This shows the spatial position of the selected fibre in the WEAVE field of view. The fibres are colour-coded according to how well the spectra meet the defined metric (in this case, the throughput of the flux compared to expectations from photometry). This map permits an at-a-glance view of issues such as cloud cover or vignetting. The selected metric is also dynamically plotted in graph form. Selection of a fibre within the field of view or within the graph will update elements 3 and 4 appropriately.
6. This displays the broad status of the instrument, as encapsulated in the "traffic light" concept described above for element 4. Each indicator will adopt thresholds defined via and editable configures file that describes acceptable parameter ranges and tolerances. This element also displays the real-time output log of the QL pipeline. Beneath this panel are a series of buttons that provide additional functionality via pop-up windows. These windows are not yet defined, but would include (for example) flux-magnitude plots, sky analysis and time-series data for monitoring trends across exposures during the night. The "Latest" button returns the user to the last processed exposure, which provides a shortcut to the latest data should they no longer wish to explore past observations.

Figure 1 – A preliminary design layout for the quick-look user interface



Last OB ID# 1013 RA: 02:30:36 Dec: 01:22:47 Name: weaveHR_15_50

Run #	UTC	OB ID	Name	RA	Dec	EXPTIME	airmass	Sci. Fibers	Status
0000115	05:17	6226	weaveHR_16_44	01:53:16	-01:16:16	1200	1.12	940	OK
0000114	04:57	6226	calibHR	01:53:16	-01:16:16	60	1.19	0	ABORTED
0000113	04:56	6226	weaveHR_16_44	01:53:16	-01:16:16	1200	1.30	942	OK
0000112	04:36	7719	calibHR	01:53:16	-01:16:16	60	1.10	0	OK
0000111	04:35	7719	weaveHR_16_44	01:53:16	-01:16:16	1200	1.12	908	OK
0000110	04:13	7719	calibHR	01:05:16	-01:16:16	60	1.20	0	OK
0000109	04:12	7719	weaveHR_12_16	00:12:55	-01:44:44	1200	1.22	905	OK
0000108	03:52	7719	calibHR	00:12:55	-01:44:44	60	1.36	0	OK
0000107	03:50	7719	weaveHR_12_16	00:12:55	-01:44:44	1200	1.22	897	OK
0000106	03:31	7719	calibHR	00:12:55	-01:44:44	60	1.36	0	OK
0000105	03:30	1004	weaveHR_12_16	00:12:55	-01:44:44	1200	1.22	932	OK
0000104	03:09	1004	calibHR	00:12:55	-01:44:44	60	1.36	0	OK
0000103	03:07	1004	weaveHR_18_40	23:34:55	-02:29:01	1200	1.22	905	OK
0000102	02:47	1004	calibHR	23:34:55	-02:29:01	60	1.36	0	OK
0000101	02:46	1004	weaveHR_18_40	23:34:55	-02:29:01	1200	1.22	966	OK
0000100	02:27	1004	calibHR	23:34:55	-02:29:01	60	1.36	0	OK
0000099	02:25	1004	weaveHR_18_40	23:34:55	-02:29:01	1200	1.22	955	OK
0000098	02:05	1004	calibHR	23:34:55	-02:29:01	60	1.36	0	OK

fibreID	CNAME	TARGSRVY	TARGPROG	TARGNAME	TARGUSE	RA	Dec	TARGPM	dMag	SNR	Sky_scale	Sky_shift	Status
0000108	WVJ01001677-0259164	WG	WG-J	QSO_1128	T	01:00:16	-02:59:16	0.00	0.40	3.10	1.00	3	P
0000109	WVJ01401671-0258169	WG	WG-J	QSO_1122	T	01:12:55	-02:59:16	0.00	0.02	1.19	0.98	1	P
0000110	WVJ01001677-0259164	WG	WG-J	QSO_551	T	01:16:27	-02:59:16	0.00	0.06	2.22	0.99	8	A
0000111	WVJ01401375-0258122	WG	WG-J	QSO_1128	T	01:30:00	-01:16:16	0.00	0.10	1.10	1.30	8	A
0000112	WVJ01001570-0255139	GA-Lrplane	GA-Lrplane	7719	T	01:05:16	-01:16:16	0.00	0.29	1.12	1.05	3	A
0000113	WVJ01001677-0259164	GA-Lrplane	GA-Lrplane	2255	T	01:42:01	-01:16:16	0.00	0.17	1.20	1.00	4	A
0000110	WVJ01401671-0258169	GA-Lrplane	GA-Lrplane	6110	T	01:05:16	-01:16:16	0.05	0.77	9.00	1.00	2	A
0000110	WVJ01401375-0258122	GA-Lrplane	GA-Lrplane	7722	T	01:05:16	-01:16:16	0.00	0.05	0.90	0.99	3	A
0000109	WVJ01001677-0259164	GA-Lrplane	GA-Lrplane	5500	T	01:16:27	-01:16:16	0.10	0.09	2.20	0.87	5	A
0000110	WVJ01401671-0258111	WG	WG-J	QSO_7706	T	01:16:27	-01:16:16	0.00	0.09	10.00	0.97	4	A
0000110	WVJ01001677-0259164	WD	WD	WD_had44	F	01:16:27	-01:16:16	0.00	0.11	1.90	0.99	5	A
0000119	WVJ01401375-0258122	WG	WG-J	QSO_5999	T	01:16:27	-01:16:16	0.00	0.31	3.20	1.02	7	A

```
QL output log
>[info] @main.py : Fiber fibID 226 is (P)arked.
```

5 APS → CPS AND APS → WAS

5.1 Data Transfer

An overview of the data and information flow for WEAVE is shown in Section 8. APS requires interfaces for receiving L1 data products (including superproducts) from the Core Processing System (CPS). We detail this mechanism in 3.1. Similarly, APS shall transfer L2 analysis of L1 data to the WEAVE Operational Repository. This is achieved in the same way CPS transfers data to APS – we outline this briefly here.

APS provides two types of products:

1. L2 analysis products - these data products are the night-by-night analysis of L1 processed raw files that CPS produces, as well as superproducts generated periodically by the CPS pipeline. These are deeper L1 stacks of targets that have been observed across multiple OBs (and indeed trimesters). APS generates L2 analysis of these deeper stacks for onwards ingestion into the WAS.
2. Contributed Software products – these are products generated by non-APS software running on the APS cluster. This ICD does not detail the contents and structure of these types of data product as it not generated by the SPA, but for reference we describe how they are transferred within the SPA dataflow system.

Both products are grouped into nightobs directories (but in separate locations) of format YYYYMMDD. Once a nightobs has been analysed and L2 files have been stored in the equivalent nightobs directory, the APS user updates the *flowstate* value in the Operational Repository proc_state table to "PENDING" for the product="L2" row for that nightobs. This update triggers a response at the Operational Repository (CASU) to transfer the data from APS. This is done via a "get" recipe from the CASU end, returning an entry of (by direct analogy to the example presented in Section 3.1)

```
20200503,/aps/weave/20200503 2020-06-13 10:12:02.874588+01:00,3
```

This example indicates that L2 analysis data from the 3rd May 2020 have been generated and is stored within the nightobs directory 20200503 located at /aps/weave/L2/20200503. This analysis was validated at 10:12 on the 13th June 2020, and comprises 3 files (the APS provides analysis of L1 products with the red and blue arms merged).

Transfer of the L2 data to CPS is via a pull of the data from the CPS end using built-in transfer tools provided with WDAP (in effect an authenticated scp connection).

Once these files have been downloaded and validated by the WEAVE Operational Repository, the proc_state *flowstate* value is updated to "VALIDATED". This signals that the transfer was successful. For onwards transfer of these data products to the WAS (using CPS as a proxy), we refer readers to 3.1.

We note that for the transfer of CS data from APS to CPS, the procedure is identical to the above, with the exception that the product entry in proc_state is "CS". The output files, regardless of the CS code run, shall all be stored in the *same* directory, with no subdirectories. The filenames of the CS products transferred are described in the ICDs of each CS package.

5.2 L2 Output FITS Files

For the purposes of this section, we exclude Contributed Software products generated at the APS and focus entirely on output generated from the L2 analysis pipeline. We refer readers respectively to [RD32] and references therein for a full list of CS codes and descriptions of their data products.

The APS operates under the general principle of analysing:

1. The deepest possible data products.
2. The greatest possible wavelength coverage.

This means that all L1 LR data products will be analysed with the Red and Blue spectral data stitched together. Moreover, for HR data, this also results in analysis of Red + Green + Blue stitched data where possible.

5.2.1 Types of L2 Output FITS Files

Table 25 defines the fundamental set of L2 output FITS files generated by the APS.

Table 25 – Main categories of L2 output FITS files

Type	Filename	Comment
single	single_r0000001__single_r0000000_APS.fits	APS analysis of L1 single MOS / IFU exposures
stack	stack_r0000001__stack_r0000000_APS.fits	APS analysis of L1 OB MOS stacks
superstack	superstack_r0000001__superstack_r0000000_APS.fits	APS analysis of L1 multi-OB MOS superstacks
MOS supertarget	WVE_00000000+0000000_ID_ARM_RESBIN_APS.fits	APS analysis of L1 MOS supertargets
LIFU cube	LWVE_00000000+0000000_ID_ARM_RESBIN_PNNNN_APS.fits	APS analysis of L1 LIFU OB stackcubes, multi-OB supercubes or supertargets
mIFU cubelets	mWVE_00000000+0000000_ID_ARM_RESBIN_PNNNN_APS.fits	APS analysis of L1 mIFU OB stackcubelets, multi-OB supercubelets or supertargets

Because the L2 analysis pipeline only analyses arm-joined data, “MOS-type OB-level” files (ie. files containing multiple spectra) have filenames taking the form:

type1_run1__type2_run2_APS.fits

However, for HR data, it is in principle possible to observe a target (under two or more OBs) using the Blue **and** Green gratings. This means arm-joined products may also take the form:

type1_run1__type2_run2__type3_run3_APS.fits

In all cases, the run numbers will correspond to observations made from bluest through to reddest arm (meaning that run1, 2 and 3 correspond to the HR Blue, Green and Red arms respectively).

Under certain circumstances, this also implies that type1, 2 and 3 do not strictly match. We highlight the example of two HR OBs observing the same fibre configuration, one using HR Green, one using HR Blue. Under this scenario, the L1 pipeline will generate a Red L1 superstack (see Section 3.2.1.1) a Green L1 OB stack and a Blue L1 OB stack. These will be analysed under APS by joining the three arms:

stack_run1__stack_run2__superstack_run3_APS.fits

For MOS supertargets (i.e. files containing just one MOS spectrum) and IFU data, the filename convention switches to the CNAME system:

WVE_00000000+0000000_ID_ARM_RESBIN_APS.fits

In the case of IFU data, the "WVE_" CNAME indicates the Central CNAME (CCNAME) at the centre of the IFU array. "ID" indicates a night-specific running index, "ARM" indicates which arms are joined within this file – again following the blue-to-red convention above, meaning "ARM" can be one of "BR", "GR", "BGR". The "RESBIN" component describes the instrument resolution (L, H) and spectral binning (1,2,4).

For IFU data, the APS analysis pipeline segments the data into *patches* [cite]:

LWVE_00000000+0000000_ID_ARM_RESBIN_PNNNN_APS.fits

The PNNNN indicates the patch number of this segmentation analysis. If the analysis returns no patches, this value is "P0000".

The "ID" component is required due to cases where, for example, cloned OBs observed in the same night would produce the same CNAME-based filename without a running index to differentiate them. We reserve the ID = "00" value for **supertarget** L1 inputs only. For **stacktarget** L1 inputs, the ID starts at "01", and iterates each time a would-be duplication occurs. This ensures no L2 outputs are overwritten - an example of this is shown in .

Because of the way CPS generates supertargets, it is impossible for two supertarget products to map to the same APS output in the same night: each time the superproduct generation is run at CPS, it would only ever produce a single deepest possible supertarget product for a given target.

We detail the scenarios under which APS analyses L1 data in the following section.

5.2.2 When is each type of L2 output file created?

Figure 2 and Figure 3 show the permitted analysis modes of the APS pipeline based on the full range of L1 data products defined in Table 25. The pathways trace out the modes under which APS will operate, based on whether the data is LR or HR, will be arm-joined or not, and whether the observations are (relevant for IFU modes) dithered. Combinations that will not be analysed are indicated by the red cross, whilst putative output filenames that would arise from these forbidden pathways are also indicated in some instances. Additional information on how files are formed is provided in boxes marked with a yellow warning triangle. The examples in the lower panel of Figure 3 indicate how the "ID" system operates for CNAME-based filename outputs.

Figure 2 – APS data I/O pathways for L1 MOS data

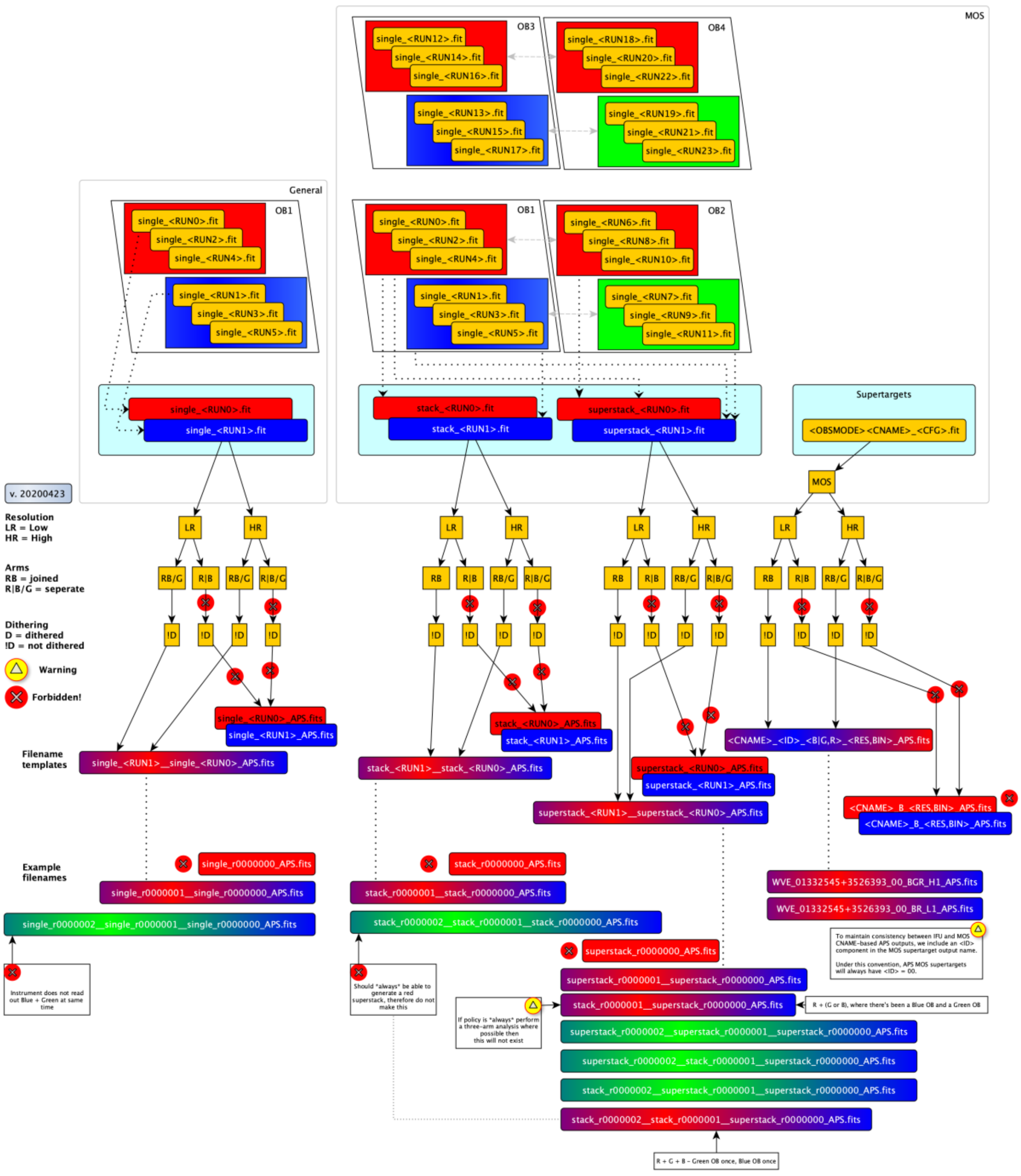
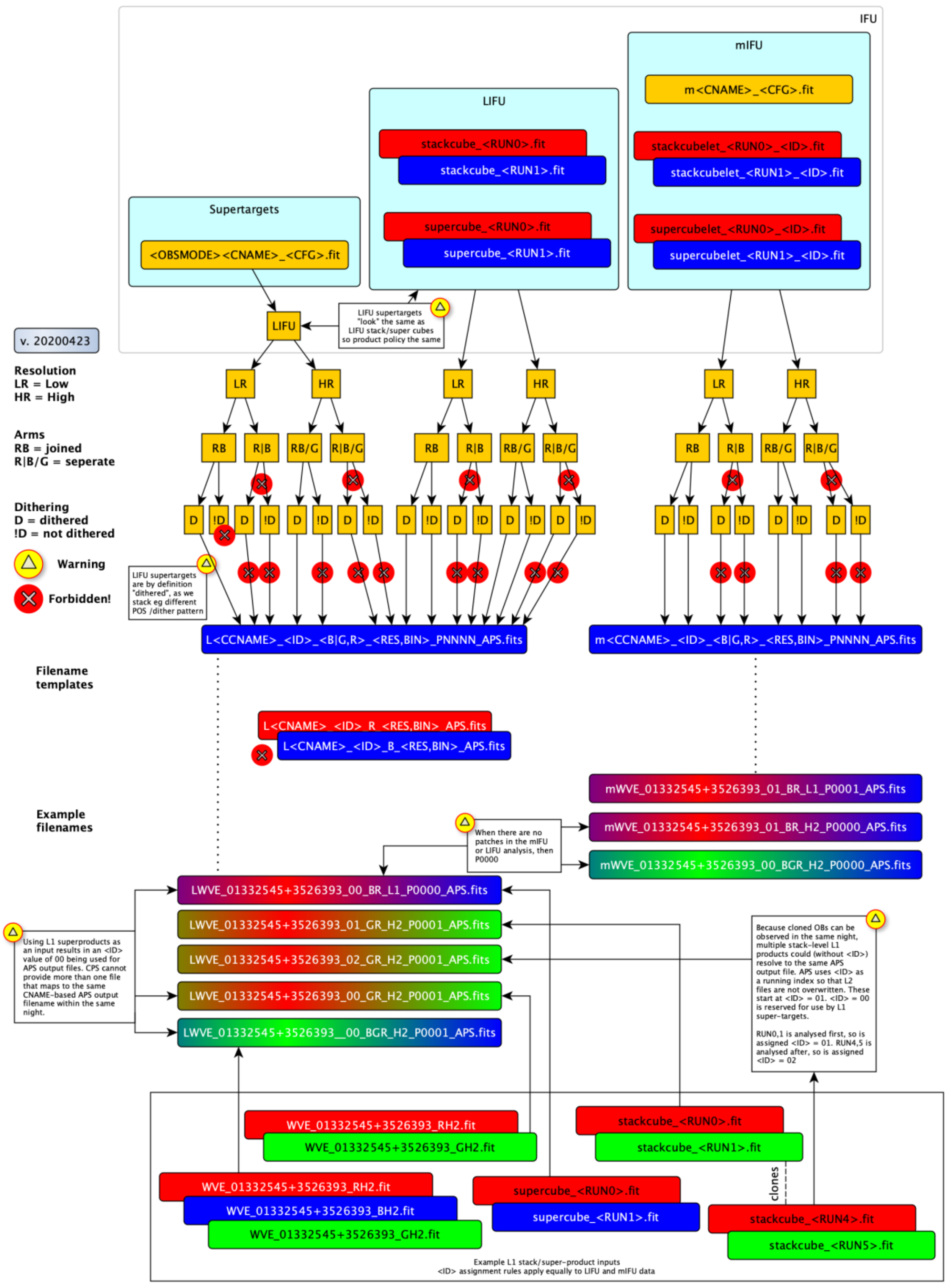

MOS
OB3
single_<RUN12>.fit
single_<RUN14>.fit
single_<RUN16>.fit
single_<RUN13>.fit
single_<RUN15>.fit
single_<RUN17>.fit
OB4
single_<RUN18>.fit
single_<RUN20>.fit
single_<RUN22>.fit
single_<RUN19>.fit
single_<RUN21>.fit
single_<RUN23>.fit
General
OB1
single_<RUN0>.fit
single_<RUN2>.fit
single_<RUN4>.fit
single_<RUN1>.fit
single_<RUN3>.fit
single_<RUN5>.fit
OB2
single_<RUN6>.fit
single_<RUN8>.fit
single_<RUN10>.fit
single_<RUN7>.fit
single_<RUN9>.fit
single_<RUN11>.fit
stack_<RUN0>.fit
stack_<RUN1>.fit
superstack_<RUN0>.fit
superstack_<RUN1>.fit
Supertargets
<OBSMODE><CNAME>_<CFG>.fit
MOS
v. 20200423
Resolution
LR = Low
HR = High
Arms
RB = joined
R|B/G = seperate
Dithering
D = dithered
!D = not dithered
Warning
Forbidden!
LR
HR
RB/G
R|B
R|B/G
RB
!D
single_<RUN0>_APS.fits
single_<RUN1>_APS.fits
single_<RUN1>__single_<RUN0>_APS.fits
stack_<RUN0>_APS.fits
stack_<RUN1>_APS.fits
stack_<RUN1>__stack_<RUN0>_APS.fits
superstack_<RUN0>_APS.fits
superstack_<RUN1>_APS.fits
superstack_<RUN1>__superstack_<RUN0>_APS.fits
<CNAME>_<ID>_<B|G,R>_<RES,BIN>_APS.fits
<CNAME>_B_<RES,BIN>_APS.fits
<CNAME>_B_<RES,BIN>_APS.fits
Filename templates
Example filenames
single_r0000000_APS.fits
single_r0000001__single_r0000000_APS.fits
single_r0000002__single_r0000001__single_r0000000_APS.fits
stack_r0000000_APS.fits
stack_r0000001__stack_r0000000_APS.fits
stack_r0000002__stack_r0000001__stack_r0000000_APS.fits
superstack_r0000000_APS.fits
superstack_r0000001__superstack_r0000000_APS.fits
stack_r0000001__superstack_r0000000_APS.fits
superstack_r0000002__superstack_r0000001__superstack_r0000000_APS.fits
superstack_r0000002__stack_r0000001__superstack_r0000000_APS.fits
stack_r0000002__superstack_r0000001__superstack_r0000000_APS.fits
stack_r0000002__stack_r0000001__superstack_r0000000_APS.fits
WVE_01332545+3526393_00_BGR_H1_APS.fits
WVE_01332545+3526393_00_BR_L1_APS.fits
To maintain consistency between IFU and MOS CNAME-based APS outputs, we include an <ID> component in the MOS supertarget output name.
Under this convention, APS MOS supertargets will always have <ID> = 00.
R + (G or B), where there's been a Blue OB and a Green OB
Instrument does not read out Blue + Green at same time
Should *always* be able to generate a red superstack, therefore do not make this
If policy is *always* perform a three-arm analysis where possible then this will not exist
R + G + B - Green OB once, Blue OB once

Figure 3 – APS data I/O pathways for L1 IFU data


IFU
mIFU
m<CNAME>_<CFG>.fit
stackcubelet_<RUN0>_<ID>.fit
stackcubelet_<RUN1>_<ID>.fit
supercubelet_<RUN0>_<ID>.fit
supercubelet_<RUN1>_<ID>.fit
LIFU
stackcube_<RUN0>.fit
stackcube_<RUN1>.fit
supercube_<RUN0>.fit
supercube_<RUN1>.fit
Supertargets
<OBSMODE><CNAME>_<CFG>.fit
LIFU
LIFU supertargets "look" the same as LIFU stack/super cubes so product policy the same
v. 20200423
Resolution
LR = Low
HR = High
Arms
RB = joined
R|B/G = seperate
Dithering
D = dithered
!D = not dithered
Warning
Forbidden!
LR
HR
RB
R|B
RB/G
R|B/G
D
!D
LIFU supertargets are by definition "dithered", as we stack eg different POS /dither pattern
L<CCNAME>_<ID>_<B|G,R>_<RES,BIN>_PNNNN_APS.fits
m<CCNAME>_<ID>_<B|G,R>_<RES,BIN>_PNNNN_APS.fits
Filename templates
L<CNAME>_<ID>_R_<RES,BIN>_APS.fits
L<CNAME>_<ID>_B_<RES,BIN>_APS.fits
Example filenames
mWVE_01332545+3526393_01_BR_L1_P0001_APS.fits
mWVE_01332545+3526393_01_BR_H2_P0000_APS.fits
mWVE_01332545+3526393_00_BGR_H2_P0000_APS.fits
When there are no patches in the mIFU or LIFU analysis, then P0000
LWVE_01332545+3526393_00_BR_L1_P0000_APS.fits
LWVE_01332545+3526393_01_GR_H2_P0001_APS.fits
LWVE_01332545+3526393_02_GR_H2_P0001_APS.fits
LWVE_01332545+3526393_00_GR_H2_P0001_APS.fits
LWVE_01332545+3526393__00_BGR_H2_P0001_APS.fits
Using L1 superproducts as an input results in an <ID> value of 00 being used for APS output files. CPS cannot provide more than one file that maps to the same CNAME-based APS output filename within the same night.
Because cloned OBs can be observed in the same night, multiple stack-level L1 products could (without <ID>) resolve to the same APS output file. APS uses <ID> as a running index so that L2 files are not overwritten. These start at <ID> = 01. <ID> = 00 is reserved for use by L1 super-targets.
RUN0,1 is analysed first, so is assigned <ID> = 01. RUN4,5 is analysed after, so is assigned <ID> = 02
WVE_01332545+3526393_RH2.fit
WVE_01332545+3526393_GH2.fit
stackcube_<RUN0>.fit
stackcube_<RUN1>.fit
clones
WVE_01332545+3526393_RH2.fit
WVE_01332545+3526393_BH2.fit
WVE_01332545+3526393_GH2.fit
supercube_<RUN0>.fit
supercube_<RUN1>.fit
stackcube_<RUN4>.fit
stackcube_<RUN5>.fit
Example L1 stack/super-product inputs
<ID> assignment rules apply equally to LIFU and mIFU data

5.2.3 Structure of L2 Outputs FITS Files

5.2.3.1 Science Data Products in MOS mode

The main data products of the APS in MOS mode are the FITS files containing the different parameters retrieved from the science analysis. These outputs will be stored in different HDUs, as the table below shows. The information on classification and redshift is common to all FITS, but in the case of galaxies, stellar fields (HDUs) will remain empty and vice versa.

Table 18 – The APS MOS product data model

Extn	Type	Extn Name	Description
HDU#0	Null byte image	PRIMARY	The primary header unit.
HDU#1	Binary table	CLASS_TABLE	The object classification and redshifts data table (see Table 19)
HDU#2	Binary table	STAR_TABLE	The star analysis data table, based on FERRE code (see Table 20).
HDU#3	Binary table	GALAXY_TABLE	The galaxy analysis data table (see Table 21).
HDU#4	Binary table	CLASS_SPEC	A binary fits table with information about the spectra analysed by the APS classification module, including the best fitting spectra (see Table 22).
HDU#5	Binary table	STAR_SPEC	A binary fits table with information about the spectra analysed by the APS stellar module, including the best fitting spectra (see Table 23).
HDU#6	Binary table	GALAXY_SPEC	A binary fits table with information about the spectra analysed by the APS galaxy modules, including the best fitting spectra, coming out from different modules (see Table 24).

Below we describe the fields of the different data tables:

Table 19 – HDU#1: The Object Classification and Redshifts data table

Column	Description
APS_ID	The unique ID number assigned by APS for every target in this field. In case of MOS data, this is equivalent to the FIBREID.
TRAGID	The identifier of the target, as assigned by the survey, being covered by the current fibre.
CNAME	A unique object name formed from the coordinates of the object, as explained in Table 4

Z	Best fit Barycentric Redshift.
ZERR	Uncertainty on best fit redshift based upon fit to $\chi 2$ minimum; NaN for invalid fit.
ZWARN	Warning flags (BIT-MASK) as explained in Table 35 (0 is good) (Hutchinson et al. 2016)
CLASS	Spectral classification. This field can be STAR, GALAXY or QSO_HIZ, QSO_LOZ
SUBCLASS	Spectral sub-classification. For galaxies this field can be AGN, STARFORMING, STARBURST or BROADLINE. For stars this field can be O, OB, B6, B9, A0, A0p, F2, F5, F9, G0, G2, G5, K1, K3, K5, K7, M0V, M2V,M1, M2, M3, M4, M5, M6, M7, M8, L0, L1, L2, L3, L4, L5, L5.5, L9, T2, Carbon, Carbon lines, CarbonWD or CV.
TARGSRVY	The name of the survey with which the target object is associated (e.g. GA-LRhalo). If it is associated with more than one survey, then this will be a list separated by commas
SNR	Signal to Noise ratio, evaluated by REDROCK
CHI2	Best fit chi2
DELTACHI2	Delta(chi2) to next best fit
NCOEFF	Number of Redrock template coefficients, used to generate the best fit
COEFF	Redrock template coefficients
NPIXELS	Number of pixels used by REDROCK
SRVY_CLASS	Spectral classification, based on the input TARGCLASS parameter. If no TARGCLASS is provided it will be based on the TARGSRVY parameter.
CZZ_X	Redshift basis used by REDROCK to generate Coarse grids Chi^2(z) distributions for each of 10 templates (X), where X can be GALAXY, QSO_HIZ, QSO_LOZ, STAR_A, STAR_B, STAR_CV, STAR_F, STAR_G, STAR_K, STAR_M and STAR_WD
CZZ_CHI2_X	Coarse grids Chi^2(z) distributions for the template X used by REDROCK, where X can be GALAXY, QSO_HIZ, QSO_LOZ, STAR_A, STAR_B, STAR_CV, STAR_F, STAR_G, STAR_K, STAR_M and STAR_WD

Table 20 – HDU#2: The Stellar Analysis data table

Column	**Description**
APS_ID	The unique ID number assigned by APS for every target in this field. In case of MOS data, this is equivalent to the FIBREID.
TARGID	The identifier of the target, as assigned by the survey, being covered by the current fibre.

CNAME	Unique object name formed from the coordinates of the object, as explained in Table 4
VRAD	Barycentric radial velocity, measured by RVSPECFIT
VRAD_ERR	Radial velocity error, measured by RVSPECFIT
SKEWNESS_RVS	Skewness, measured by RVSPECFIT
KURTOSIS_RVS	Kurtosis, measured by RVSPECFIT
LOGG_RVS	Surface gravity (g in cm/s**2), measured by RVSPECFIT
TEFF_RVS	Effective temperature, measured by RVSPECFIT
VSINI_RVS	Rotational velocity, measured by RVSPECFIT
FEH_RVS	Metallicity [Fe/H]=log10(N(Fe)/N(H)) - log10(N(Fe)/N(H))sun, measured by RVSPECFIT
ALPHA_RVS	Alpha-to-iron ratio [alpha/Fe], measured by RVSPECFIT
LOGG_ERR_RVS	Uncertainties in Surface gravity (LOGG), measured by RVSPECFIT
TEFF_ERR_RVS	Uncertainties in Effective temperature (TEFF), measured by RVSPECFIT
FEH_ERR_RVS	Uncertainties in Metallicity [Fe/H], measured by RVSPECFIT
ALPHA_ERR_RVS	Uncertainties in Alpha-to-iron ratio [alpha/Fe], measured by RVSPECFIT
SNR_RVS	Signal to Noise ratio, evaluated by RVSPECFIT
CHISQ_TOT_RVS	Total chi**2, evaluated by RVSPECFIT
TEFF	Effective temperature
TEFF_ERR	Uncertainties in Effective temperature (TEFF)
LOGG	Surface gravity (g in cm/s**2)
LOGG_ERR	Uncertainties in Surface gravity (LOGG)
FEH	Metallicity [Fe/H]=log10(N(Fe)/N(H)) - log10(N(Fe)/N(H))sun
FEH_ERR	Uncertainties in Metallicity [Fe/H]
ALPHA	Alpha-to-iron ratio [alpha/Fe]
ALPHA_ERR	Uncertainties in Alpha-to-iron ratio [alpha/Fe]
MICRO	Microturbulence
MICRO_ERR	Uncertainties in Microturbulence
COVAR	Covariance matrix for ([Fe/H], [a/Fe], logmicro, Teff,logg)
ELEM	Elemental abundance ratios to iron [elem/Fe]
ELEM_ERR	Uncertainties in the elemental abundance ratios to iron
SNR_FR	Signal to Noise ratio, evaluated by FERRE
CHISQ_TOT_FR	Total chi**2, evaluated by FERRE
FLAG_FR	Bit indicating whether the code has likely produced useful results (1: yes, 0: No)

Table 21 – HDU#3: Galaxy Analysis data table

Column	Description
APS_ID	The unique ID number assigned by APS for every target in this field. In case of MOS data, this is equivalent to the FIBREID.
TARGID	The identifier of the target, as assigned by the survey, being covered by the current fibre.
CNAME	Unique object name formed from the coordinates of the object, as explained in Table 4
ZCORR	Redshift, corrected for the best fit velocity of the stellar component (The initial guess for redshift was based on CLASS module: REDROCK)
V	Systemic velocity of the stellar component
SIGMA	Velocity dispersion of the stellar component
H3	Gauss-Hermite moment h_3 of the stellar component
H4	Gauss-Hermite moment h_4 of the stellar component
H5	Gauss-Hermite moment h_5 of the stellar component
H6	Gauss-Hermite moment h_6 of the stellar component
ERR_ZCORR	Uncertainty in the corrected Redshift, taking into account the uncertainty of the initial guess for redshift was based on CLASS module (REDROCK) and the error in the Systemic velocity of the stellar component, obtained through Monte-Carlo simulations by pPXF
ERR_V	Uncertainty in Systemic velocity of the stellar component, obtained through Monte-Carlo simulations by pPXF (parameter: MC_PPXF)
ERR_SIGMA	Uncertainty in velocity dispersion of the stellar component, obtained through Monte-Carlo simulations by pPXF
ERR_H3	Uncertainty in H3, obtained through Monte-Carlo simulations by pPXF
ERR_H4	Uncertainty in H4, obtained through Monte-Carlo simulations by pPXF
ERR_H5	Uncertainty in H5, obtained through Monte-Carlo simulations by pPXF
ERR_H6	Uncertainty in H6, obtained through Monte-Carlo simulations by pPXF
FORM_ERR_ZCORR	Uncertainty in the corrected Redshift, taking into account the uncertainty of the initial guess for redshift was based on CLASS module (REDROCK) and the formal error on the initial fit by PPXF on the extracted Systemic velocity of the stellar component
FORM_ERR_V	The formal error on the initial fit by pPXF on the extracted Systemic velocity of the stellar component
FORM_ERR_SIGMA	The formal error on the initial fit by pPXF on the extracted velocity dispersion of the stellar component

FORM_ERR_H3	The formal error on the initial fit by pPXF on the extracted Gauss-Hermite moment h_3 of the stellar component
FORM_ERR_H4	The formal error on the initial fit by pPXF on the extracted Gauss-Hermite moment h_4 of the stellar component
FORM_ERR_H5	The formal error on the initial fit by pPXF on the extracted Gauss-Hermite moment h_5 of the stellar component
FORM_ERR_H6	The formal error on the initial fit by pPXF on the extracted Gauss-Hermite moment h_6 of the stellar component
FLUX_**EL**_*W*	Flux of the emission line (**EL**) at the rest-frame wavelength (*W*), measured by GANDALF Refer to Table 37 for the list of available emission lines
AMPL_**EL**_*W*	Amplitude of the emission line (**EL**) at the rest-frame wavelength (*W*), measured by GANDALF
Z_**EL**_*W*	Redshift of the emission line (**EL**) at the rest-frame wavelength (*W*), measured by GANDALF
SIGMA_**EL**_*W*	Gaussian width of the emission line (**EL**) at the rest-frame wavelength (*W*), measured by GANDALF
AON_**EL**_*W*	Amplitude-over-noise threshold for emission-line subtraction, applied to the the emission line (**EL**) at the rest-frame wavelength (*W*)
EBMV_**EL**_*W*	De-redden the spectra for the Galactic extinction in the direction of the target, applied to the the emission line (**EL**) at the rest-frame wavelength (*W*)
ERR_FLUX_**EL**_*W*	Error in FLUX_**EL**_*W*
ERR_AMPL_**EL**_*W*	Error in AMPL_**EL**_*W*
ERR_Z_**EL**_*W*	Error in Z_**EL**_*W*
ERR_SIGMA_**EL**_*W*	Error in SIGMA_**EL**_*W*
ERR_EBMV_**EL**_*W*	Error in EBMV_**EL**_*W*
INDX	Spectral index **INDX** (Refer to Table 36 for the list of available indices)
ERR_**INDX**	Error in Spectral index **INDX**
FWHM_FLAG'	A flag indicates if the total intrinsic dispersion is larger than the LIS measurement resolution (1: yes, 0:no)

Table 22 – HDU#4: The classification spectra data table

Column	Description
APS_ID	The unique ID number assigned by APS for every target in this field. In case of MOS data, this is equivalent to the FIBREID.

TARGID	The identifier of the target, as assigned by the survey, being covered by the current fibre.
CNAME	Unique object name formed from the coordinates of the object, as explained in Table 4
LAMBDA_RR_**ARM**	The wavelength array (converted to air), used by the classifier module for the **ARM** band (B: Blue, G:Green, R: Red, C: stitched arms)
FLUX_RR_**ARM**	The actual spectra analysed by the APS classification module for the **ARM** band (B: Blue, G:Green, R: Red, C: stitched arms), corrected by the sensitivity function
IVAR_RR_**ARM**	The inverse variance of the former spectra.
MODEL_RR_**ARM**	Best fit spectra reconstructed by the classification for the **ARM** band (B: Blue, G:Green, R: Red, C: stitched arms)

Table 23 – HDU#5: The stellar spectra data table

Column	Description
APS_ID	The unique ID number assigned by APS for every target in this field. In case of MOS data, this is equivalent to the FIBREID.
TARGID	The identifier of the target, as assigned by the survey, being covered by the current fibre.
CNAME	Unique object name formed from the coordinates of the object, as explained in Table 4
LAMBDA_RVS_**ARM**	The wavelength array, used by RVSPECFIT module (Radial velocity measurements and atmospheric parameters) for the **ARM** band (B: Blue, G:Green, R: Red, C: stitched arms)
FLUX_RVS_**ARM**	The actual spectra analysed by RVSPECFIT module for the **ARM** band (B: Blue, G:Green, R: Red, C: stitched arms), corrected by the sensitivity function
ERROR_RVS_**ARM**	Error of the former spectra.
MODEL_RVS_**ARM**	Best fit spectra obtained from RVSPECFIT for the **ARM** band (B: Blue, G:Green, R: Red, C: stitched arms)
LAMBDA_FR_**ARM**	The wavelength array, used by FERRE module for the **ARM** band (B: Blue, G:Green, R: Red, C: stitched arms)
FLUX_FR_**ARM**	The normalised spectra analysed by FERRE module for the **ARM** band (B: Blue, G:Green, R: Red, C: stitched arms), corrected by the sensitivity function
ERROR_FR_**ARM**	Error of the former spectra.
MODEL_FR_**ARM**	Best fit spectra obtained from FERRE module for the **ARM** band (B: Blue, G:Green, R: Red, C: stitched arms)

Table 24 – HDU#6: The galaxy spectra data table

Column	Description
APS_ID	The unique ID number assigned by APS for every target in this field. In case of MOS data, this is equivalent to the FIBREID.
TARGID	The identifier of the target, as assigned by the survey, being covered by the current fibre.
CNAME	Unique object name formed from the coordinates of the object, as explained in Table 4
LOGLAM_PPXF	The rest-frame wavelength array (redshift taken from the classifier module, but not corrected for the velocity, evaluated by pPXF itself), equally spaced in velocity domain (logarithmically binned), used by pPXF
FLUX_PPXF	logarithmically binned spectra used by pPXF
ERROR_PPXF	The error array of the former spectra.
MODEL_PPXF	The best fit to the spectrum by pPXF
GOODPIX_PPXF	Array of not masked spectral pixels, used by pPXF
LOGLAM_GAND	The wavelength array, shifted to the rest-frame, (according to the initial guess of redshift by the classifier module). equally spaced in velocity domain (logarithmically binned), used by GANDALF
FLUX_GAND	logarithmically binned spectra used by pPXF
ERROR_GAND	The error array of the former spectra.
MODEL_GAND	The best fit to the spectrum, including continuum and emission lines
EMISSION_GAND	Emission lines spectra
FLUX_CLEAN_GAND	Emission line subtracted spectra
MODEL_CLEAN_GAND	The best fit to the spectrum, excluding the emission lines
GOODPIX_GAND	Array of not masked spectral pixels, used by GANDALF

5.2.3.2 Science Data Products in IFU mode (LIFU, mIFU)

For each detected target in m/LIFU fields, APS defines a patch, could be considered as a sub-cube. The output of the IFU Analysis Software for each path (sub-cube) will be stored as a separate FITS file. The proposed format of this FITS file (dedicated to each patch) for Galactic (aka, STAR) and Extra-Galactic sources (aka, GALAXY) is the following:

Table 25 – The APS IFU product Data Model (Extragalactic sources)

Extension	Type	Extension Name	Description
HDU#0	Null byte image	PRIMARY	The primary header unit.

HDU#1	Binary table	PATCH_ALLSPEC	A binary FITS table, including the logarithmically binned original spectra (equally spaced spectra in velocity domain) for each spaxel in that specific path (see Table 26)
HDU#2	Binary table	PATCH_TABLE	Data table containing all necessary information to reconstruct the Voronoi binning, map bins to spaxels and vice versa. In particular, this table contains one line per spaxel and states the BIN_ID as well as spatial coordinates of every spaxel (see Table 27)
HDU#3	Binary table	PATCH_BINSPEC	Data table containing the logarithmically binned spectra for each bin (adaptive Voronoi spatial bin) or spaxel (if no binning is applied) one row per bin/spaxel. The BIN_ID in this table links results to the original spaxel through the PATCH_TABLE (HDU#2).
HDU#4	Binary table	GALAXY_TABLE	The galaxy analysis data table, one row per bin/spaxel (see Table 29) for each Patch .
HDU#4	Binary table	GALAXY_SPEC	A binary fits table (one row per bin/spaxel) with information about the spectra analysed by the APS galaxy modules, including the best fitting spectra, coming out from different galaxy analysis modules (see Table 30).

Below we describe the PATCH_ALLSPEC table (Extragalactic sources):

Table 26 – HDU#1: The PATCH_ALLSPEC table (Extragalactic sources)

Column	**Description**
APS_ID	A sequence number assigned to each spaxel by APS pipeline on the original IFU field (before defining patches).
TARGID	The identifier of the central target in the original IFU field, as assigned by the survey
CNAME	Unique object name formed from the coordinates of the original IFU field centre (CCNAME)
LOGLAM	The rest-frame wavelength array for each spaxel in this PATCH (initial guess for redshift taken from the

	classifier module), equally spaced in velocity domain (logarithmically binned)
SPEC	logarithmically binned spectra, one line per spaxel
ESPEC	The error array of the former spectra, one line per spaxel.

Table 27 – HDU#2: The PATCH_TABLE table (Extragalactic sources)

Column	**Description**
ID	A sequence index assigned to each spaxel in the PATCH
BIN_ID	The bin number assigned by the Voronoi-binning code to each input spaxel. If no binning is applied, it will be a sequence index assigned to each good spaxel, not necessarily identical to the ID.
APS_ID	A sequence number assigned to each spaxel by APS pipeline on the original IFU field (before defining patches).
TARGID	The identifier of the central target in the original IFU field, as assigned by the survey
CNAME	Unique object name formed from the coordinates of the original IFU field centre (CCNAME)
X	Relative coordinates in arcsec (along RA direction) with respect to the centre of the Field
Y	Relative coordinates in arcsec (along DEC direction) with respect to the centre of the Field
Z	Initial guess for redshift of the target in this specific PATH
ZERR	Error in Z
HEALPIX_ID	HEALPix index (Resolution = 10, NSide=1024) of the bin centroid/spaxel coordinates
X_0	The RA of the position of the centre of the field (before defining Patches) in decimal degrees.
Y_0	The Declination of the position of the centre of the field (before defining Patches) in decimal degrees.
FLUX	The mean FLUX of each spaxel
SNR	The Signal to Noise ratio of each spaxel
XBIN	The X coordinates of the bin generators
YBIN	The Y coordinates of the bin generators
SNRBIN	The final SN of each bin after Voronoi tessellation process
NSPAX	The number of spaxels contributed to generate each bin

Table 28 – HDU#3: The PATCH_BINSPEC table (Extragalactic sources)

Column	Description
BIN_ID	The bin number assigned to each bin for each PATCH, generated through the Voronoi tessellation process. If no binning is applied, it will be a sequence index, defined in the PATCH_TABLE table, assigned to each good spaxel in this specific PATCH. This ID could be used to link each BIN to its generators (spaxels) through the PATACH_TABLE table.
LOGLAM	The rest-frame wavelength array for each BIN/spaxel in this PATCH (initial guess for redshift taken from the classifier module), equally spaced in velocity domain (logarithmically binned)
SPEC	Logarithmically binned spectra, one line per BIN/spaxel (in case no binning applied)
ESPEC	The error array of the former spectra, one line per BIN/spaxel (in case no binning applied).

Below we describe the GALAXY_TABLE columns, presenting the results of analysis on each BIN of a specific PATH. BIN_ID could be used to link results for each bin to its generators (spaxels) through the PATCH_TABLE table.

Table 29 – HDU#4: Galaxy_TABLE table (Extragalactic sources)

Column	Description
BIN_ID	The bin number assigned to each bin for each PATCH, generated through the Voronoi tessellation process. If no binning is applied, it will be a sequence index, defined in the PATCH_TABLE table, assigned to each good spaxel in this specific PATCH. This ID could be used to link each BIN to its generators (spaxels) through the PATACH_TABLE table.
V	velocity of the stellar component, with respect to the systemic velocity, evaluated from the redshift Z (see PATCH_TABLE table)
SIGMA	Velocity dispersion of the stellar component
H3	Gauss-Hermite moment h_3 of the stellar component
H4	Gauss-Hermite moment h_4 of the stellar component
H5	Gauss-Hermite moment h_5 of the stellar component
H6	Gauss-Hermite moment h_6 of the stellar component
ERR_V	Uncertainty in velocity of the stellar component of each BIN/spaxel, obtained through Monte-Carlo simulations by pPXF (parameter: MC_PPXF)
ERR_SIGMA	Uncertainty in velocity dispersion of the stellar component, obtained through Monte-Carlo simulations by pPXF

ERR_H3	Uncertainty in H3, obtained through Monte-Carlo simulations by pPXF
ERR_H4	Uncertainty in H4, obtained through Monte-Carlo simulations by pPXF
ERR_H5	Uncertainty in H5, obtained through Monte-Carlo simulations by pPXF
ERR_H6	Uncertainty in H6, obtained through Monte-Carlo simulations by pPXF
FORM_ERR_ZCORR	Uncertainty in the corrected redshift, taking into account the uncertainty of the initial guess for redshift was based on CLASS module (REDROCK) and the formal error on the initial fit by PPXF on the extracted Systemic velocity of the stellar component
FORM_ERR_V	The formal error on the initial fit by pPXF on the extracted velocity of the stellar component
FORM_ERR_SIGMA	The formal error on the initial fit by pPXF on the extracted velocity dispersion of the stellar component
FORM_ERR_H3	The formal error on the initial fit by pPXF on the extracted Gauss-Hermite moment h_3 of the stellar component
FORM_ERR_H4	The formal error on the initial fit by pPXF on the extracted Gauss-Hermite moment h_4 of the stellar component
FORM_ERR_H5	The formal error on the initial fit by pPXF on the extracted Gauss-Hermite moment h_5 of the stellar component
FORM_ERR_H6	The formal error on the initial fit by pPXF on the extracted Gauss-Hermite moment h_6 of the stellar component
FLUX_**EL**_*W*	Flux of the emission line (**EL**) at the rest-frame wavelength (*W*), measured by GANDALF Refer to Table 37 for the list of available emission lines
AMPL_**EL**_*W*	Amplitude of the emission line (**EL**) at the rest-frame wavelength (*W*), measured by GANDALF
Z_**EL**_*W*	Redshift of the emission line (**EL**) at the rest-frame wavelength (*W*), measured by GANDALF
SIGMA_**EL**_*W*	Gaussian width of the emission line (**EL**) at the rest-frame wavelength (*W*), measured by GANDALF
AON_**EL**_*W*	Amplitude-over-noise threshold for emission-line subtraction, applied to the the emission line (**EL**) at the rest-frame wavelength (*W*)
EBMV_**EL**_*W*	De-redden the spectra for the Galactic extinction in the direction of the target, applied to the the emission line (**EL**) at the rest-frame wavelength (*W*)
ERR_FLUX_**EL**_*W*	Error in FLUX_**EL**_*W*
ERR_AMPL_**EL**_*W*	Error in AMPL_**EL**_*W*
ERR_Z_**EL**_*W*	Error in Z_**EL**_*W*
ERR_SIGMA_**EL**_*W*	Error in SIGMA_**EL**_*W*

ERR_EBMV_**EL**_*W*	Error in EBMV_**EL**_*W*
INDX	Spectral index **INDX** (refer to Table 36 for the list of available indices)
ERR_**INDX**	Error in Spectral index **INDX**
FWHM_FLAG'	A flag indicates if the total intrinsic dispersion is larger than the LIS measurement resolution (1: yes, 0:no)

Table 30 – HDU#5: The GALAXY_SPEC table (Extragalactic sources)

Column	**Description**
BIN_ID	The bin number assigned to each bin for each PATCH, generated through the Voronoi tessellation process. If no binning is applied, it will be a sequence index, defined in the PATCH_TABLE table, assigned to each good spaxel in this specific PATCH. This ID could be used to link each BIN to its generators (spaxels) through the PATACH_TABLE table.
LOGLAM_PPXF	The rest-frame wavelength array (redshift taken from the classifier module, but not corrected for the velocity, evaluated by pPXF itself), equally spaced in velocity domain (logarithmically binned), used by pPXF
FLUX_PPXF	logarithmically binned spectra used by pPXF
ERROR_PPXF	The error array of the former spectra.
MODEL_PPXF	The best fit to the spectrum by pPXF
GOODPIX_PPXF	Array of not masked spectral pixels, used by pPXF
LOGLAM_GAND	The wavelength array, shifted to the rest-frame, (according to the initial guess of redshift by the classifier module). equally spaced in velocity domain (logarithmically binned), used by GANDALF
FLUX_GAND	logarithmically binned spectra used by pPXF
ERROR_GAND	The error array of the former spectra.
MODEL_GAND	The best fit to the spectrum, including continuum and emission lines
EMISSION_GAND	Emission lines spectra
FLUX_CLEAN_GAND	Emission line subtracted spectra
MODEL_CLEAN_GAND	The best fit to the spectrum, excluding the emission lines
GOODPIX_GAND	Array of not masked spectral pixels, used by GANDALF

Table 31 – The APS IFU product Data Model (Galactic sources)

Extn	**Type**	**Extn Name**	**Description**
HDU#0	Null byte image	PRIMARY	The primary header unit.
HDU#1	Binary table	PATCH_TABLE	Data table, containing the classification, spatial information and some basic statistics of each pixel in the patch. In particular, this table contains one line per resampled pixel in and states where The BIN_ID in this table links parameters in this table to the stellar parameters presented in the other extension of this FITS file (see Table 32 for more details)
HDU#2	Binary table	STAR_TABLE	The STAR analysis data table, one row per spaxel for each Patch. BIN_ID can be used to link parameters in this table to the results presented in other extensions. (See Table 33)
HDU#3	Binary table	STAR _SPEC	A binary fits table (one row per spaxel) with information about the spectra analysed by the APS STAR modules, including the best fitting spectra, coming out from the RVS and FERRE modules BIN_ID can be used to link parameters in this table to the results presented in other extensions. (See Table 34)

Table 32 – HDU#1: The IFU PATCH_TABLE table (Galactic sources)

Column	**Description**
ID	A sequence index assigned to each spaxel in the PATCH
BIN_ID	An ID assigned by PyAPS to each spaxel in this PATCH. it will be a sequence index, defined in the PATCH_TABLE table. This ID could be used to link each row in this table to the other extensions of this FITS file.
APS_ID	Sequence number assigned to each spaxel by APS pipeline on the original IFU field (before cutting to patches).

TARGID	The identifier of the central target in the original IFU field, as assigned by the survey
CNAME	Unique object name formed from the coordinates of the original IFU field centre (CCNAME)
X	Relative coordinates in arcsec (along RA direction) with respect to the centre of the Field
Y	Relative coordinates in arcsec (along DEC direction) with respect to the centre of the Field
Z	Initial guess for redshift of the target in this specific PATH
ZERR	Error in Z
HEALPIX_ID	HEALPix index (Resolution = 10, NSide=1024) of the bin centroid/spaxel coordinates
X_0	The RA of the position of the centre of the field (before defining Patches) in decimal degrees.
Y_0	The Declination of the position of the centre of the field (before defining Patches) in decimal degrees.
FLUX	The mean FLUX of each spaxel
SNR	The Signal to Noise ratio of each spaxel
XBIN	The X coordinates of the bin generators
YBIN	the Y coordinates of the bin generators
SNRBIN	the final SN of each bin after Voronoi tessellation process
NSPAX	the number of spaxels contributed to generate each bin

Table 33 – HDU#2: The IFU Stellar Analysis data table (Galactic sources)

Column	**Description**
BIN_ID	An ID assigned to each spaxel in this PATCH. it will be a sequence index, defined in the PATCH_TABLE table. This ID could be used to link each row in this table to the other extensions of this FITS file.
VRAD	Barycentric radial velocity, measured by RVSPECFIT
VRAD_ERR	Radial velocity error, measured by RVSPECFIT
SKEWNESS_RVS	Skewness, measured by RVSPECFIT
KURTOSIS_RVS	Kurtosis, measured by RVSPECFIT
LOGG_RVS	Surface gravity (g in cm/s**2), measured by RVSPECFIT
TEFF_RVS	Effective temperature, measured by RVSPECFIT

VSINI_RVS	Rotational velocity, measured by RVSPECFIT
FEH_RVS	Metallicity [Fe/H] = log10(N(Fe)/N(H)) - log10(N(Fe)/N(H))sun, measured by RVSPECFIT
ALPHA_RVS	Alpha-to-iron ratio [alpha/Fe], measured by RVSPECFIT
LOGG_ERR_RVS	Uncertainties in Surface gravity (LOGG), measured by RVSPECFIT
TEFF_ERR_RVS	Uncertainties in Effective temperature (TEFF), measured by RVSPECFIT
FEH_ERR_RVS	Uncertainties in Metallicity [Fe/H], measured by RVSPECFIT
ALPHA_ERR_RVS	Uncertainties in Alpha-to-iron ratio [alpha/Fe], measured by RVSPECFIT
SNR_RVS	Signal to Noise ratio, evaluated by RVSPECFIT
CHISQ_TOT_RVS	Total chi**2, evaluated by RVSPECFIT
TEFF	Effective temperature
TEFF_ERR	Uncertainties in Effective temperature (TEFF)
LOGG	Surface gravity (g in cm/s**2)
LOGG_ERR	Uncertainties in Surface gravity (LOGG)
FEH	Metallicity [Fe/H] = log10(N(Fe)/N(H)) - log10(N(Fe)/N(H))sun
FEH_ERR	Uncertainties in Metallicity [Fe/H]
ALPHA	Alpha-to-iron ratio [alpha/Fe]
ALPHA_ERR	Uncertainties in Alpha-to-iron ratio [alpha/Fe]
MICRO	Microturbulence
MICRO_ERR	Uncertainties in Microturbulence
COVAR	Covariance matrix for ([Fe/H], [a/Fe], logmicro, Teff,logg)
ELEM	Elemental abundance ratios to iron [elem/Fe]
ELEM_ERR	Uncertainties in the elemental abundance ratios to iron
SNR_FR	Signal to Noise ratio, evaluated by FERRE
CHISQ_TOT_FR	Total chi**2, evaluated by FERRE
FLAG_FR	Bit indicating whether the code has likely produced useful results (1: yes, 0: No)

Table 34 – HDU#3: The IFU stellar spectra data table (Galactic sources)

Column	**Description**
BIN_ID	An ID assigned to each spaxel in this PATCH. it will be a sequence index, defined in the PATCH_TABLE table. This ID could be used to link each row in this table to the other extensions of this FITS file.
LAMBDA_RVS_**ARM**	The wavelength array, used by RVSPECFIT module (Radial velocity measurements and atmospheric parameters) for the **ARM** band (B: Blue, G:Green, R: Red, C: stitched arms)

FLUX_RVS_**ARM**	The actual spectra analysed by RVSPECFIT module for the **ARM** band (B: Blue, G:Green, R: Red, C: stitched arms), corrected by the sensitivity function
ERROR_RVS_**ARM**	Error of the former spectra.
MODEL_RVS_**ARM**	Best fit spectra obtained from RVSPECFIT for the **ARM** band (B: Blue, G:Green, R: Red, C: stitched arms)
LAMBDA_FR_**ARM**	The wavelength array, used by FERRE module for the **ARM** band (B: Blue, G:Green, R: Red, C: stitched arms)
FLUX_FR_**ARM**	The normalised spectra analysed by FERRE module for the **ARM** band (B: Blue, G:Green, R: Red, C: stitched arms), corrected by the sensitivity function
ERROR_FR_**ARM**	Error of the former spectra.
MODEL_FR_**ARM**	Best fit spectra obtained from FERRE module for the **ARM** band (B: Blue, G:Green, R: Red, C: stitched arms)

Table 35 – ZWARN bit-mask definitions (Hutchinson et al. 2016)

Bit	**Name**	**Definition**
0	SKY	Sky fiber
1	LITTLE COVERAGE	Insufficient wavelength coverage
2	SMALL DELTA CHI2	$\chi 2$ of best fit is too close to that of second best (< 0.01 in $\chi 2$ r)
5	Z FITLIMIT	$\chi 2$ min at edge of the redshift fitting range (Z ERR set to -1)
7	UNPLUGGED	Fiber was broken or unplugged, and no spectrum was obtained
8	NULL FIT	At least one template class had constant-valued $\chi 2$ surface

List of line-strength indices, available to be presented/analysed by the PyAPS pipeline. In this table, b3 and b4 indicate the minimum and maximum wavelength of feature bandpass, respectively.

Table 36 – List of available line strength indices to be measured by the LS module

#	**names**	**b3 (Å)**	**b4 (Å)**	**Reference**
1	BL1719	1709.0000	1729.0000	(Fanelli+90)
2	BL1853	1838.0000	1868.0000	(Fanelli+90)
3	Fe2332	2333.0000	2359.0000	(Fanelli+90)
4	Fe2402	2382.0000	2422.0000	(Fanelli+90)
5	BL2538	2520.0000	2556.0000	(Fanelli+90)
6	Fe2609	2596.0000	2622.0000	(Fanelli+90)

7	Mgwide	2670.0000	2870.0000	(Fanelli+90)
8	BL2720	2713.0000	2733.0000	(Fanelli+90)
9	BL2740	2736.0000	2762.0000	(Fanelli+90)
10	Mg2800	2784.0000	2814.0000	(Fanelli+90)
11	Mg2852	2839.0000	2865.0000	(Fanelli+90)
12	Fe3000	2965.0000	3025.0000	(Fanelli+90)
13	BL3096	3086.0000	3106.0000	(Fanelli+90)
14	Mg3334	3328.0000	3340.0000	(Serven+11)
15	NH3360	3350.0000	3400.0000	(Davidge,Clark94)
16	NH3375	3350.0000	3400.0000	(Serven+11)
17	Ni3520	3511.0000	3530.0000	(Gregg94)
18	Fe3581	3548.0000	3594.0000	(Gregg94)
19	Fe3619	3602.0000	3623.5000	(Gregg94)
20	Fe3631	3628.5000	3637.5000	(Gregg94)
21	Fe3646	3642.5000	3650.0000	(Gregg94)
22	Fe3683	3676.0000	3691.5000	(Gregg94)
23	Fe3706	3701.0000	3711.0000	(Gregg94)
24	Fe3741	3701.0000	3772.0000	(Gregg94)
25	UV_CN	3780.0000	3900.0000	(Pickles85,Davidge,Clark94)
26	H10Fe	3786.0000	3802.0000	(Gregg94)
27	CNB	3810.0000	3910.0000	(Brodie,Huchra90)
28	MgH93838	3814.0000	3843.0000	(Gregg94)
29	CNO3862	3840.3000	3883.4000	(Serven+05)
30	CN3883	3847.0000	3880.0000	(Gregg94)
31	CaHK	3899.5000	4003.5000	(Serven+05)
32	CaIIH_K	3915.0000	4000.0000	(Pickles85)
33	CaK3933	3924.5000	3942.5000	(Gregg94)
34	H_K	3925.0000	3995.0000	(Brodie,Huchra90)
35	CaH3968	3959.5000	3977.5000	(Gregg94)
36	FeBand	4025.5000	4085.0000	(Gregg94)
37	Fe4033	4027.5000	4039.0000	(Gregg94)
38	Fe4046	4041.0000	4051.0000	(Gregg94)
39	Fe4064	4056.0000	4070.0000	(Gregg94)
40	D4000	4050.0000	4250.0000	(Bruzual83) (TBC)
41	Sr4077	4071.0000	4083.0000	(Gregg94)
42	HdA	4083.5000	4122.2500	(Worthey,Ottaviani97)
43	HdF	4091.0000	4112.2500	(Worthey,Ottaviani97)
44	CNO4175	4129.4000	4219.8000	(Serven+11)
45	CN1	4142.1250	4177.1250	(Trager+98)
46	CN2	4142.1250	4177.1250	(Trager+98)
47	Ca4227	4222.2500	4234.7500	(Trager+98)
48	G4300	4281.3750	4316.3750	(Trager+98)
49	HgA	4319.7500	4363.5000	(Worthey,Ottaviani97)
50	Fe4326	4320.5000	4329.5000	(Gregg94)
51	HgF	4331.2500	4352.2500	(Worthey,Ottaviani97)
52	Hg_sigma_27	4331.5000	4351.8750	(Vazdekis,Arimoto99)
53	Hg_sigma_20	4332.0000	4352.2500	(Vazdekis,Arimoto99)
54	Hg_sigma_12	4333.0000	4352.7370	(Vazdekis,Arimoto99)
55	Hg_sigma_13	4333.2500	4363.0000	(Vazdekis+01)
56	Fe4383	4369.1250	4420.3750	(Trager+98)
57	Fe4457	4448.5000	4467.0000	(Gregg94)
58	Ca4455	4452.1250	4474.6250	(Trager+98)

59	Fe4531	4514.2500	4559.2500	(Trager+98)
60	Fe4592	4520.0000	4538.0000	(Gregg94)
61	FeII4550	4543.0000	4557.0000	(Gregg94)
62	Ca4592	4578.0000	4603.0000	(Gregg94)
63	CO4685	4626.4000	4743.3000	(Serven+05)
64	C2_4668	4634.0000	4720.2500	(Trager+98)
65	bTiO	4758.5000	4800.0000	(Spiniello+14)
66	Mg4780	4760.8000	4798.8000	(Serven+05)
67	Hbeta_o	4839.2750	4877.0970	(Cervantes,Vazdekis09)
68	Hbeta	4847.8750	4876.6250	(Trager+98)
69	Fe4920	4914.0000	4926.0000	(Gregg94)
70	Fe5015	4977.7500	5054.0000	(Trager+98)
71	Mg1	5069.1250	5134.1250	(Trager+98)
72	MgH	5084.0000	5140.0000	(Gregg94)
73	MgG	5150.0000	5195.0000	(Brodie,Huchra90)
74	Mg2	5154.1250	5196.6250	(Trager+98)
75	Mgb	5160.1250	5192.6250	(Trager+98)
76	Fe5270	5245.6500	5285.6500	(Trager+98)
77	Fe5335	5312.1250	5352.1250	(Trager+98)
78	Fe5406	5387.5000	5415.0000	(Trager+98)
79	aTiO	5445.0000	5600.0000	(Spiniello+14)
80	Fe5709	5696.6250	5720.3750	(Trager+98)
81	Fe5782	5776.6250	5796.6250	(Trager+98)
82	NaD	5876.8750	5909.3750	(Trager+98)
83	TiO1	5936.6250	5994.1250	(Trager+98)
84	Ca6162	6155.0000	6177.0000	(Gregg94)
85	Fe6189	6185.0000	6193.0000	(Gregg94)
86	TiO2	6189.6250	6272.1250	(Trager+98)
87	TiO2sdss	6189.6250	6272.1250	(LaBarbera+13)
88	CaH1	6357.5000	6401.7500	(Spiniello+14)
89	Fe6497	6487.0000	6504.0000	(Gregg94)
90	Halpha	6548.0000	6578.0000	(Cohen+98)
91	Ha_Gregg94	6554.0000	6570.0000	(Gregg94)
92	CaH2	6775.0000	6900.0000	(Spiniello+14)
93	IR_NaI	8170.0000	8220.0000	(Pickles85)
94	NaI8190sdss	8180.0000	8200.0000	(LaBarbera+13)
95	NaI8200A	8180.0000	8200.0000	(Vazdekis+12)
96	TiO1_0.85	8450.0000	8700.0000	(Carter+86)
97	Ca_DTT_1	8483.0000	8513.0000	(Diaz+89)
98	Ca1	8484.0000	8513.0000	(Cenarro+01,LaBarbera+13)
99	CaAZ88_1	8490.0000	8506.0000	(Armandroff,Zinn88)
100	Ca2	8522.0000	8562.0000	(Cenarro+01,LaBarbera+13)
101	Ca_DTT_2	8527.0000	8557.0000	(Diaz+89)
102	CaAZ88_2	8532.0000	8552.0000	(Armandroff,Zinn88)
103	Ca3	8642.0000	8682.0000	(Cenarro+01,LaBarbera+13)
104	Ca_DTT_3	8647.0000	8677.0000	(Diaz+89)
105	CaAZ88_3	8653.0000	8671.0000	(Armandroff,Zinn88)
106	MgI_DTT	8799.5000	8814.0000	(Diaz+89)
107	TiO2_0.89	8890.0000	9060.0000	(Carter+86)
108	IR_CN	9160.0000	9300.0000	(Pickles85)
109	CN_ZrO	9170.0000	9470.0000	(Carter+86)

Here we present the list of the emission lines that could be presented by the extragalactic module (GANDALF) in APS pipeline. The columns trivially specify an index, name, and rest-frame (vacuum/air) wavelength for each line.

Table 37 – List of the emission lines that could be presented by the extragalactic module (GANDALF) in APS pipeline

#	Name	Lambda (Å)	
		air	**vacuum**
0	HeII	3203.15	-
1	[NeV]	3345.81	-
2	[NeV]	3425.81	-
3	[OII]	3726.03	-
4	[OII]	3728.73	-
5	[NeIII]	3868.69	-
6	[NeIII]	3967.40	-
7	H5	3889.05	-
8	He	3970.07	-
9	Hd	4101.73	-
10	Hg	4340.46	-
11	[OIII]	4363.15	-
12	HeII	4685.74	-
13	[ArIV]	4711.30	-
14	[ArIV]	4740.10	-
15	Hb	4861.32	-
16	[OIII]	4958.83	-
17	[OIII]	5006.77	-
18	[NI]	5197.90	-
19	[NI]	5200.39	-
20	HeI	5875.60	-
21	[OI]	6300.20	-
22	[OI]	6363.67	-
23	[NII]	6547.96	-
24	Ha	6562.80	-
25	[NII]	6583.34	-
26	[SII]	6716.31	-
27	[SII2]	6730.68	-
30	[ArIII]	7135.67	-
31	OVI	-	1033.82
32	Lya	-	1215.24
33	NV	-	1240.81
34	OI	-	1305.53
35	CII	-	1335.31
36	SiIV	-	1398
37	OIV	-	1402
38	NIV	-	1483
39	NIV	-	1487
40	CIV	-	1549.48
41	HeII	-	1640.4

42	OIII	-	1665.85
43	NIII	-	1750
44	SiII	-	1814
45	SiIII	-	1883
46	SiIII	-	1892
47	AlIII	-	1857.4
48	CIII	-	1908.73
49	OIII	2320.28	-
50	CII]	2325.28	-
51	NeIV	2438.76	-
52	MgII	2798.29	-
53	NeVI	3425.87	-
54	H12	3749.93	-
55	H11	3770.93	-
56	H10	3797.92	-
57	H9	3835.91	-
58	HeI	3887.90	-
59	H8	3888.90	-
60	SII	4071.15	-
61	FeV	4229.80	-
62	OIII	4363.21	-
63	HeI	4471.74	-
64	HeII	4685.69	-
65	OIII	4931.23	-
66	NII	5754.40	-
67	HeI	5875.37	-
68	NaI	5896.36	-
69	NI	6527.23	-
70	HeI	6678.16	-
71	[OII]	7319.46	-
72	OII	7329.98	-
73	SIII]	9068.60	-
74	SIII]	9530.60	-
75	FeIII	2076.62	-
76	FeII	2324.58	-
77	FeII	2626.92	-
78	FeII	2964.28	-
79	FeII	3498.92	-
80	FeII	4564.71	-
81	[FeV]	4071.24	-
82	[FeXIV]	5302.86	-
83	[ClIII]	5537.89	-
84	[FeVII]	5720.71	-
85	[FeVII]	6086.29	-

6 CPS → USERS

The WEAVE survey teams will be able to access their survey data from either the WAS or from the operational repository hosted at CASU. Although most data access will be supported through the WAS, there will be instances where access to the most recent analysis is needed on short time scales and hence the need for the Operational Repository to be able to release data to users.

The Operational Repository will have a web interface that can be used by members of the survey teams to search for their data. The interface will be password protected and will ensure that each group can only search for data belonging to their own project (although special accounts will be set up to allow access to all data). Requests for data will be queued and when a background process completes the request, the user will receive an email with the location of their data in the form of an http link. The instructions in the email will tell them how to download their data using authenticated wget or lftp. This is the method currently used at CASU for the Gaia-ESO survey and such requests are filled in an average of 5 minutes.

7 WAS → USERS

7.1 The WAS signature

The WAS tags every file with the WASID keyword placed in the main FITS header; this is a mapping between the file name with an immutable unique identifier created by the archive, which permits identifying a file even if its name is duplicated. The WAS also creates the WASROOT keyword for the products coming from OCS, CPS and APS (including CS products); this keyword points to the combined target released by APS (considered by WAS as root), where the red arm is joined with the blue arm. This signature is particularly useful when producing new analysis, as for example CDPs (contributed data products).

7.2 Data download (WAS → users)

The different products stored in WAS are available for download in FITS format. The files may be downloaded via http protocol and with rsync protocol.

While the rsync protocol is the suggested method for synchronizing the data coming from an internal data release candidate or a regular subscription downloads, the http protocol (mostly used for downloading one time jobs) offers more flexibility and the ability to transfer files in parallel. This can be achieved with standard tools like wget and curl.

We refer users to [RD30] for more information about data access at WAS.

8 SPA BACKEND DATAFLOW SYSTEM

This section describes how the dataflow system has been designed. It is intended for any users that require some form of data access to CASU-hosted products, and indeed for users wishing to store data products on CASU machines.

Much of this section is a technical description of the planned implementation and design of the dataflow system. We refer users to Section 9 and 10 for concise guidance and usage of the interface.

To maintain an efficient and documented operational dataflow, it is important to:

1. Record where specific data products are stored
2. Know what the state of these products are, for example are they:
   a. Still being downloaded?
   b. Ready to be analysed?
   c. Validated by CASU?
   d. Ready to be transferred downstream?
3. Be able to update the state of data products when work has been done
4. Provide a dataflow ledger describing the historical states of a particular data product

We achieve points 1–4 above via a series of tables in the OR:

1. proc_state – the current dataflow and processing state of a particular product (e.g. L1) on a particular night (e.g. 20110505)
2. proc_log – a record of each time the proc_state table is updated with an actionable event
3. proc_daemon – a record of the last dataflow event processed by the daemon
4. proc_prefs – per-node configuration of alert notifications and the data deletion policy in the event of data retraction [see Sections 8.4 and 8.6.2.1]

Within the data transport, data processing and analysis context, the fundamental unit of dataflow is a night (ie. a YYYYMMDD *nightobs*). This is the level of granularity that CASU records data on points 1-4 above. We therefore do not describe changes[4] to the processing status on a per-OB or per-file resolution: either the night is properly processed, or remediation must be made until the night is fully processed.

End-users will have access to proc_state via the WEAVE Data Access Platform (WDAP) and web-based OR, and proc_log via the web-based OR. Where required, users will be granted permission to insert and update rows in proc_state .

[4]Under this scheme, we do nevertheless provide a mechanism do describe, for information purposes only, the progress of a dataflow / data processing task.

8.1 The proc_state table

Each row in this table represents a data product for a given night. As it is this table that end-users will interact with (albeit, likely in an obfuscated way via helper scripts), we fully document the columns in this section.

There are four broad groups of columns provided in this table, detailed in the following subsections.

8.1.1 Product specification

These columns describe the "what" (the type of data product) and the "when" (the night the data products are connected to).

column	format	description
product	char(3)	The type of data ("RAW", "L1"...)
nightobs	int	The night the data products are associated to

8.1.2 Product metadata

These columns describe additional information about the product that might is useful.

column	format	description
id	bigint	The primary key of the table
nfiles	int	The number of files for this particular product type
datapath	char(255)	The physical location on the disk
info	char(255)	Freeform text field providing additional information
<X>stamp	datetime	Timestamp when the processing state or particular dataflow state was last updated (X=proc,flow,wa,ap)
created	datetime	Timestamp when this row was created

8.1.3 Processing and analysis: *procstate*

This column defines what actions are (or have been) performed on the data product.

column	format	description
procstate	char(10)	The processing (or analysis) state of the product

There are a limited set of values this column may take:

- NONE
- PROCESSING

- PROCESSED
- FAILED

Only processing nodes may change the *procstate* column, with limitations on changes from one state to another. For example, changing from "PROCESSED" → "PROCESSING" is not a permitted route. We detail the valid pathways in Section 8.6.

8.1.4 Dataflow states: *flowstate* and *wastate*

These columns describe, for a particular node, the availability of data and once in-place, the validation progress of these data. Each node has dataflow representation under this scheme:

- *flowstate*: the OR-centric dataflow (relating to the availability / validation status of data at CASU)
- *apstate*: the equivalent signalling for the APS
- *wastate*: the state of data intended for the WAS

column	format	description
<X>state	char(12)	The dataflow state for this product either for the processing-side (*flowstate, apstate*) or archiving (*wastate*)

There are a limited number of values this column may take:

- NONE
- PENDING
- TRANSFERRING
- TRANSFERRED
- VALIDATING
- VALIDATED or
- FAILED

Nodes should (*generally*, CASU is the exception) react only to their specific *flowstate* and do not need to worry about the others.

Processing nodes **cannot** set a dataflow state column directly – they can only do so via changes to *procstate* (this triggers certain changes via the OR). We detail the linkage in Section 8.6. Other nodes (for example WAS) **can** set their dataflow state (*wastate*) directly.

When a dataflow state is set to "PENDING", it indicates that the relevant node needs to act, as data are available "to work on" – the exact action is context dependent:

- When *flowstate*="PENDING" for an entry with an L1 product type, this signals that the pipeline should commence processing raw data.
- When *wastate*="PENDING", this signals that the data are ready for transfer to the archive.

In the latter case, we note that WAS have authority to change the *wastate* to reflect archive-related dataflow progress. For example: under (simplified[5]) normal operations, once the L1 pipeline has completed processing, the OR will validate the processed products and mark them as "VALIDATED" under *flowstate*. This in turn triggers an update of *wastate* from "NONE" to "PENDING". This acts as the signal for WAS to begin transfer of the data. They can set *wastate*="TRANSFERRING" whilst this is happening, and to *wastate*="TRANSFERRED" once this process is completed.

As with the OR, the WAS will perform checks on the data prior to allowing users access to it – this can be signalled by changing *wastate* to "VALIDATING" followed by "VALIDATED". We note that this state in no way reflects (nor should it) the data release state (in terms of internal releases etc) – that is a matter for WAS and has no bearing on the data the CASU provides for onwards dissemination (which it does indiscriminately).

Further information on event pathways can be found in Section 8.6.

In some instances, the columns are inapplicable to the type of data described by the row. For example, no pipeline processing or analysis is done on RAW data. This means that the *procstate* column will remain "NONE" at all times. Moreover, any attempt to change this value will be rejected by the OR.

8.2 The proc_log table

Any changes to the *procstate*, *flowstate*, *apstate* and *wastate* are recorded in the proc_log table. Changes to the info column are not recorded (as not change in the dataflow is provided). The proc_log table provides the "before" and "after" values for each of these 3 states, as well as the user that made the change, and when. We describe this table below:

column	format	description
id	bigint	The primary key of the table
procstate_id	bigint	ID of the entry in the proc_state table
nightobs	int	Night of the entry (YYYYMMDD)
product	char(3)	The type of data ("RAW", "L1"...)
<X>state_change	bool	Did the Xstate change in this update (X=proc,flow,ap,wa)
<X>state_old	char(12)	The Xstate before the update (X=proc,flow,ap,wa)

[5] In reality, under the current data transfer model CASU will only make data available to the WAS once the full set of L1, L2 and CS data have been generated for that particular night. The example provided instead releases the product as soon as it has been validated by the OR.

<X>state_new	char(12)	The Xstate after the update (X=proc,flow,ap, wa)
timestamp	datetime	The time when the update was made
rescinded	boolean	Has this change been subsequently rescinded?
who	char(5)	The node that made the change
what	char(16)	The OR trigger that enacted the change

We describe the *rescinded* column in Sections 8.5.1.2 and 9.3.

8.3 The proc_daemon table

This one-row table provides top-level information to the OR, including the mount point of the data disk, and a pointer to the last event acted upon.

column	format	description
id	bigint	-1 (info-type event) or 0 (actionable event)
procstate_id	char(40)	ID of the data product in proc_state table
proclog_id	char(3)	Log ID of the event in proc_log table
timestamp	smallint	Timestamp of last event processed
grace_period	boolean	The number of seconds grace before an actionable event is processed
info_counter	datetime	Number of info notification events processed so far
counter		Number of events processed so far
data_loc		Pointer to disk store
was_release	char(7)	Release data to WAS on a per-night or per-product basis

8.4 The proc_prefs table

This table provides settings for how the OR should behave for certain nodes. It allows users to define the type of communications received from the OR and sets the policy for how data are treated in the event of a data retraction event. Each node associated with a data product has an entry in the table that can be updated. We describe the contents of this table below:

column	format	description
id	bigint	The primary key of the table
user	char(40)	The database username (casu, was,...)
product	char(3)	The name of the data product (L1, L2, CS,..)

alerts	smallint	The requested alert level (0: none, 1: error, 2: warning, 4: info)
clobber	boolean	Should the OR delete the nightobs data in the event of a retraction
modified	datetime	Timestamp when this row was modified

The alert levels are [TBC]

8.5 Dataflow events

A dataflow event is defined as the change in one of the columns in the proc_state table. The OR detects when a change in state has been made, and (depending on the type of event) makes an appropriate update to the relevant timestamp column (*procstamp, flowstamp, apstamp, wastamp*) to reflect when this change was made.

Broadly, there are two types of events, **non-actionable** and **actionable**.

8.5.1 Non-actionable events

These events do not require any further action from nodes within the context of operational dataflow. There are currently two non-actionable events:

8.5.1.1 Information updates

Nodes can provide free-form text updates to either processing or dataflow tasks by updating the *info* column in proc_state. This does not require any further action, as it is merely a means to provide (for example) a running commentary on progress. The L1 pipeline might use this to indicate, while processing RAW data, that it is currently: "Working on OB #5/25". These text fields are exposed to the OR web frontend and can therefore provide web-users with a more granular insight into processing progress. Updates to *info* are limited to the "owner" of the data: the L1 pipeline cannot make updates to the L2 *info* column.

8.5.1.2 Rescinded actions

As mentioned in Section 8.2 and in section 9.3 below, we describe a mechanism to rescind an actionable event. In doing so, this stops the OR from treating the update to the database as an event to be acted upon. This is useful in the event an update is issued prematurely or incorrectly.

8.5.2 Actionable events

These events require some (re)action from either the **node** or the **OR**.

8.5.2.1 OR-actionable events

Actions required of the OR may include:

8.5.2.1.1 A trigger to generate further entries in the proc_state table

As an example, the Incoming RAW data script inserts a *product*=’RAW’ entry into proc_state for a new night. This acts as a trigger for the OR to generate equivalent entries for L1, L2 and CS for that *nightobs*.

8.5.2.1.2 A trigger to change different rows and/or columns in the proc_state table

For example, the APS user has decided to retract their analysis for a night (for more on retraction, see Section 8.6.2). To do this, they set *procstate*=’NONE’. This is a retraction signal that OR acts on by in turn setting the *apstate*=’PENDING’ and the *wastate*=’NONE’ for this particular APS nightobs. OR acts on multiple rows where there is a re-processing cascade (see Section 8.6.2.3): an L1 retraction triggers changes to the APS and CS states because any new L1 data will require re-analysis by these pipelines.

8.5.2.1.3 Returning a warning or error when users attempt something that is not permitted

For example, the APS user has decided to retract their analysis for a night. Rather than setting the *procstate* as described above, they decide to set the *apstate*=’NONE’. This update is not permitted, and the OR will issue an EXCEPTION notification with a message describing the violation.

8.5.2.2 Node-actionable events

These events are generally much simpler and are limited to statements that:

8.5.2.2.1 Data need to be processed

For example, the incoming RAW data have been downloaded and prepared for the L1 node. The OR has set *flowstate*=’PENDING’ for *product*=’L1’ for that night. This event signals to L1 that data needs to be processed.

8.5.2.2.2 Data need to be transferred to an external node

For example, following validation of the aforementioned L1 data by the OR (*flowstate*=’VALIDATED’) the data are made available for transfer to the WAS – this is signalled by the OR setting *wastate*=’PENDING’ for this entry in the proc_state table.

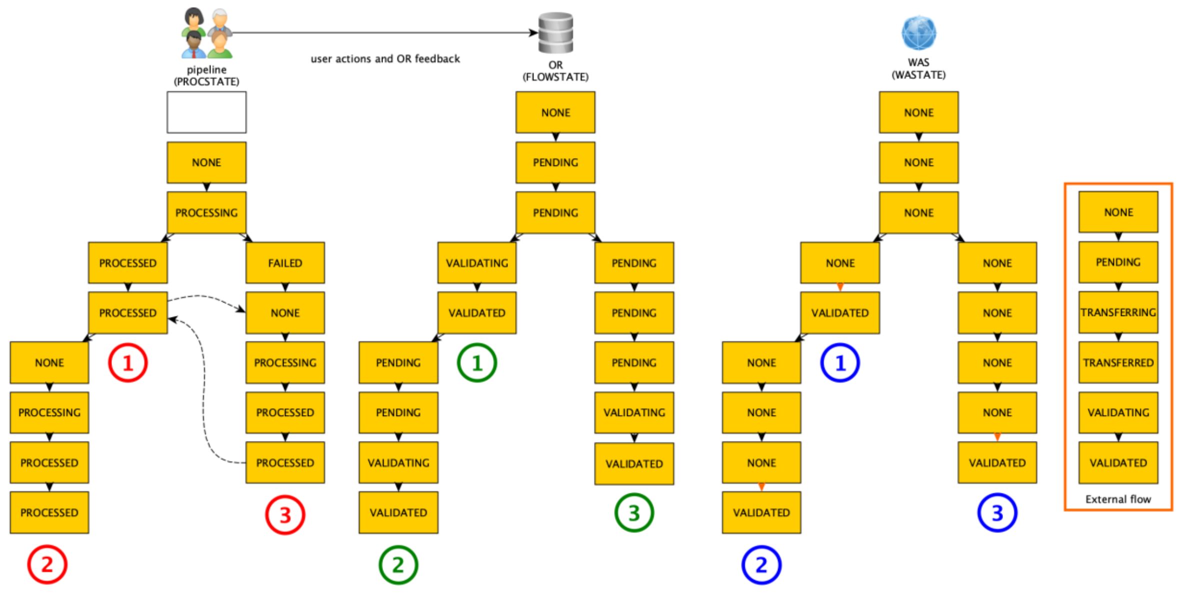
These last two examples highlight the important principle that a ‘**PENDING’** value in one of the dataflow states **indicates that work needs to be done**. From the data processing perspective, the combination of *product* and dataflow state identifies the node responsible for the work. From the archiving perspective, the presence of a *wastate*=’PENDING’ is the sole requirement to commence transfer of the data to the archive.

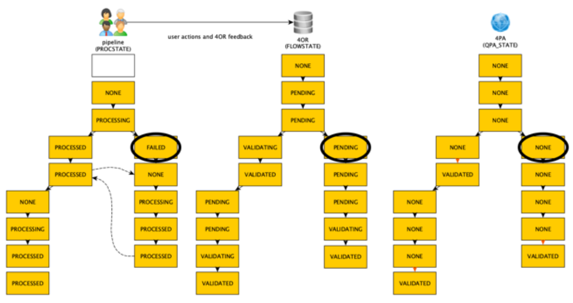
8.6 Event pathways

As alluded to above, the dataflow system accommodates multiple pathways leading to multiple end-states. In the idealised case, all data products progress to *procstate*=”PROCESSED” and a *flowstate*=”VALIDATED”. Similarly for the archive, the aspiration is that all *wastate*=”VALIDATED”. The route to these outcomes is not always direct, and indeed these outcomes are not guaranteed. The dataflow system therefore must accommodate different event pathways that permit users to retract the data, signal a failure in the processing, or indeed react to data that did not pass OR validation.

With one exception (cascade retraction), all event pathways arise directly from interaction that a particular end-user has with the proc_state table. In Figure 4 below, we show users changing the *procstate* (left) along three processing pathways, and how these updates in turn trigger changes in the *flowstate* (middle) and *wastate* (right). Each box in the *procstate* has an equivalent box in the same position representing the values of *flowstate* and *wastate* at the same point in the pathway. This first figure shows the “pipeline-centric” perspective, where the other two nodes (OR and WAS) react to decisions taken by the pipeline user.

Figure 4 – Dataflow event pathways





The circled boxes in this indicative plot show the values of the three proc_state values at the same point in the event pathway.
A dashed arrow indicates that the user could in principle switch to another branch at that juncture. For example, upon reaching the end of pathway 2, it is possible to join pathway 3.

To maintain the 1:1 mapping between events across these three states, we "compress" the WAS data transfer operations – instances of these compressed sequences of *wastate* are indicated by the red arrows. The sequence of values *wastate* adopts are show in the red box in the main figure. The three key event pathways depicted here are:

8.6.1 Normal processing (1)

The pipeline receives the *flowstate*="PENDING" signal, updates *procstate* to indicate that PROCESSING has started, and then concludes by updating this to PROCESSED. This state triggers the validation process in OR, and once this is complete, an update to the *wastate* is made to reflect availability of new data for transfer[6].

8.6.1.1 OR housekeeping

To ensure the integrity and veracity of data products stored within the OR, the repository will contextually grant or restrict read and write access to data products and/or the directories they are stored in. For example, the L1 pipeline should not write to the output directory until it receives a signal that processing should start. This signal, as described above, provides a trigger for the OR to change the permission state of the target directory to allow writes. Similarly, once the processing is complete, the OR will prevent writes to the directory. This provides an assurance that been *flowstate*='VALIDATED' has not been subsequently modified.

From the archive perspective, the above data will not be readable until the *wastate*='PENDING'. This is to ensure that data are not prematurely transferred.

8.6.2 Retraction (2)

Once reaching the "normal processing" endstate (Section 8.6.1), there is a decision to retract the data. This is performed by setting *procstate*="NONE". Once the OR has completed housekeeping tasks related to this request, it will update the *flowstate*="PENDING". This indicates to the pipeline that data can be re-generated for that night, and the end-user proceeds along the "normal processing" route. Note that the archive *wastate*="NONE" once a retraction is issued. This prevents WAS from inadvertently transferring old/incomplete data whilst the pipeline is in the process of retraction.

Once the data have been re-validated by OR (*flowstate*="VALIDATED") then *wastate* is updated to "PENDING" to signal that updated data are available for that night. It is the responsibility of the archive to recognise that such a signal for previously transferred data is due to a retraction. This can be achieved by comparing the *wastamp* in the OR proc_state table to a copy stored by WAS at the point they originally signalled the previous data were successfully transferred and *wastate*="VALIDATED".

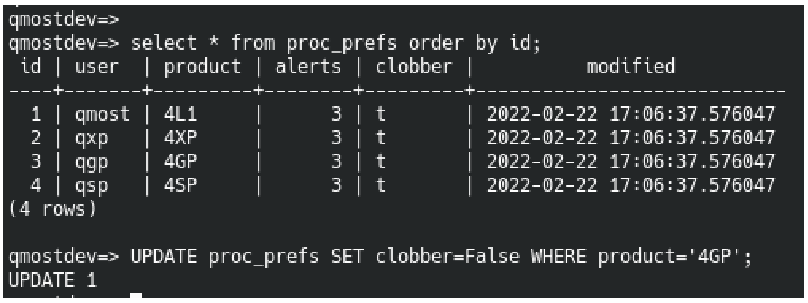
[6] We note under the current operational model that *wastate* is only updated for each data product once all three have been processed for a given night. OR does not release a night for transfer until all pipelines have finished and validated.

8.6.2.1 OR retraction housekeeping

As detailed above, OR performs housekeeping tasks prior to releasing the night for re-processing. In particular:

1. Entries relating to this night are removed from the OR database
2. Access permissions to nightobs directories are modified and
3. (optionally) data within the output directory are deleted

The latter point (deletion) can be controlled by users changing the *clobber* entry in the proc_prefs table of the OR. If users do not wish for the data to be deleted in the event of a retraction, they should set *clobber*="False":



```
qmostdev=>
qmostdev=> select * from proc_prefs order by id;
 id | user  | product | alerts | clobber |          modified
----+-------+---------+--------+---------+----------------------------
  1 | qmost | 4L1     |      3 | t       | 2022-02-22 17:06:37.576047
  2 | qxp   | 4XP     |      3 | t       | 2022-02-22 17:06:37.576047
  3 | qgp   | 4GP     |      3 | t       | 2022-02-22 17:06:37.576047
  4 | qsp   | 4SP     |      3 | t       | 2022-02-22 17:06:37.576047
(4 rows)

qmostdev=> UPDATE proc_prefs SET clobber=False WHERE product='4GP';
UPDATE 1
```

8.6.2.2 Retraction during dataflow operations

The OR will prevent attempts to retract data whilst dataflow operations are ongoing. For example, if data are still *wastate*="TRANSFERRING" to WAS, or are in the process of being *flowstate*="VALIDATED" by the OR, these actions **must** reach an end state before the retraction can be issued.

8.6.2.3 Cascade retraction

Cascade retractions occur when a product that other products rely on (for data input) are retracted. If CASU issues a retraction for a night of L1 data, the intent is to re-process the night and require the L2 pipelines to re-analyse it. To this end, a retraction in L1 data will cascade down to L2 and CS.

In these instances, *flowstate*s for the L2 pipelines will be set to 'NONE' (as they were whilst waiting for L1 data originally) until the L1 pipeline has finished re-processing. Similarly, where RAW data have been retracted, the *flowstate* for the L1 pipeline will be set to NONE. These event pathways are not detailed in the above figure, because they do not arise from the end-user setting the *procstate* for their product.

Whilst such a retraction event may trigger a notification email, pipelines should be prepared to pick up previously analysed nights that suddenly re-appear. For this reason, CASU's preference is that the *clobber* preference in proc_prefs be set to True. This ensures no cross-contamination of (for example) residual files persisting between retraction events.

8.6.3 Pipeline failure (3)

The third pathway in the figure is pipeline failure. Whilst *procstate*='PROCESSING', when a pipeline fails (for example it aborts), no update is made to the OR. It is the responsibility of the end-user to recognise these cases and act on them.

Whilst it is not mandatory to do so, users can signal a failure in the pipeline by setting *procstate*="FAILED". No housekeeping tasks are performed on the nightobs directory upon reaching this state, and developers are free to bugfix and test their code. Setting a FAILED status acts as an indicator that the pipeline is not still *procstate*='PROCESSING', and might help avoid emails from stakeholders asking why the pipeline appears to still be working on the data. The OR will not attempt validation and ingestion of any night with FAILED status, whereas it will attempt to do so for a PROCESSED status.

Transition out of a *procstate*='FAILED' state is by way of retraction described in Section 8.6.2: once ready to proceed, users set *procstate*="NONE" to signal a re-processing should commence. Retention of the data in this circumstance is subject to *proc_prefs* and is as described in pathway 2.

8.6.4 Validation failure

Whilst not shown in the pathway figures, at each point the OR is tasked with VALIDATING a night, an alternative endpoint in the dataflow is failure due to either:

1. The validation tasks run by the OR highlight an error in the data
2. The code performing the validation failed

Whilst the latter occurs only rarely, it is important to communicate instances where validation has failed (*flowstate*='FAILED') and under what circumstances. The two cases above can be distinguished by checking the *info* column in the proc_state table. Specifics on why the validation failed can be provided by email alert.

Transition out of a *flowstate*='FAILED' state involves issuing a retraction by setting *procstate*='NONE'. After retraction housekeeping, this will reset the *flowstate* to 'PENDING' and allow the user to continue under one of the above event pathways. Please note that in cases where the data products fail validation (case 1 above), they will continue to do so until the underlying issues are fixed with the data provided. Any failures under case (2) should be communicated to CASU via JIRA.

8.6.5 External transfers to/from CASU

Nodes operating outside of CASU require transfer of data products from IoA servers to their locale. In some cases, data products also need to be returned. These instances are signalled by modifications to the relevant flowstate column. In the above event pathway diagram, we show the sequence of *wastate* values for transfer of data to the WAS:

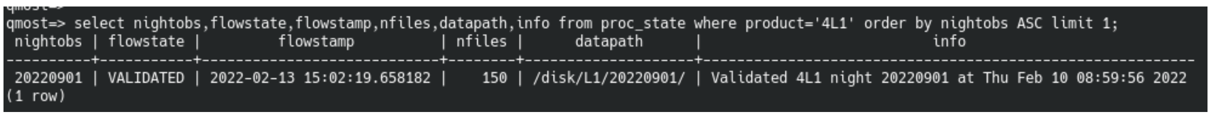
NONE→PENDING→TRANSFERRING→TRANSFERRED→VALIDATING→VALIDATED

We note that the latter two states **do not** represent the validation state at CASU, but rather at the destination node. It is the responsibility of the end-user to set their dataflow values accordingly and to perform their own validation as appropriate.

9 SPA DATAFLOW INTERFACING WITH THE OR

This section describes how to interface with the Operational Repository (hereafter OR) Database under the design described in Section 8.

Armed with the above information in Section 8 and authenticated user access to the OR proc_state data table, end-users can interact with and engage in the operational dataflow. Section 11 provides the full DB schema for the tables described in Sections 8.1–8.4. The OR is a PostgreSQL database meaning users can form SQL queries (where permitted) to issue SELECT, UPDATE and INSERT requests to the database via (for example) the *psql* client:



```
qmost=> select nightobs,flowstate,flowstamp,nfiles,datapath,info from proc_state where product='4L1' order by nightobs ASC limit 1;
 nightobs | flowstate |         flowstamp          | nfiles |      datapath      |                          info
----------+-----------+----------------------------+--------+--------------------+--------------------------------------------------------
 20220901 | VALIDATED | 2022-02-13 15:02:19.658182 |    150 | /disk/L1/20220901/ | Validated 4L1 night 20220901 at Thu Feb 10 08:59:56 2022
(1 row)
```

Users engaged in the data workflow are responsible for "housekeeping" of their individual workflows, including handling cases where their pipelines fail (as described in Section 8.6.3 above). CASU however provides a series of recipes that provide means to "get" and "set" the processing and dataflow status without having to deal with the database directly. This collection of recipes is called the **WEAVE Data Access Point (WDAP).**

The scope of the WDAP is purely limited to engagement with the OR data processing tables and is functionally equivalent to issuing SQL queries in the manner shown above. End-users are responsible for the appropriate use of these recipes, although the WDAP does provide some degree of pre-vetting of calls to the database.

In general, there are some advantages of using WDAP over direct interaction with the database, these are (but not limited to):

- Recipe integration into Python workflows…
- …that perform data transfers (via scp or rsync)
- Pipeline "friendliness" with convenience arguments that help with processing workflow ("don't return me nights I'm currently processing; I don't want to try and process them at the same time!")
- OR notifications integrated into a Python logging instance (where instantiated by the end-user)
- Validation queries to ensure updates have been committed
- Some pre-vetting of actions before calls to the database
- Quick rescinding of previous actionable event issued by the end user

- Easy switching between operational and test databases

Limitations: because WDAP is just an interface to the OR database, it is not responsible for (for example):

- Data permission issues whereby certain directories/files cannot be read
- Network outages preventing access to the OR
- The availability and performance of the OR database
- Handling pipeline faults, failures, and how these should be communicated to OR (though it *does* provide you with the recipe to report this)

Despite these, WDAP should report via raised exception when there are communication issues with the database. In those instances, users should report the issue to CASU via JIRA referring to an issue with database connectivity rather than a fault with WDAP.

In what follows below, we describe the main recipes available via the WDAP, and the

```
equivalent SQL queries
```

that can be made via an appropriate SQL client.

Recipes are provided on a per-node basis, meaning that each node will have recipes tailored to their participation in the overall operational dataflow. Each recipe can be run either standalone from a terminal, or via a Python function call.

The collection of recipes is version controlled and available for download via the consortium software repository platform. Please consult the README.md file for more detailed deployment information as well as additional usage examples.

Connection to the CASU machine and database is predicated on authentication via IP address and matching SSH key. Would-be users attempting to connect to the system using these recipes without the correct credentials will not be granted access.

9.1 Preliminary installation

To use the recipes, first ensure you have an account on the consortium's software repository platform. Working within the machine that you have provided access credentials for, take a clone of the repository:

```
git clone git@gitlab.ast.cam.ac.uk:weave/wdap.git ./wdap
[this will be updated in later draft]
```

Now `cd` into the cloned copy and check the README file. This will provide more detailed install and running instructions, but in general these are:

1. Create a new Python virtual environment
2. Activate the environment

3. Install the required packages
4. Change to the node directory
5. Update the node's configuration file (`node.json`)

9.1.1 Configuration files

There are two configuration files, both are encoded in JSON format:

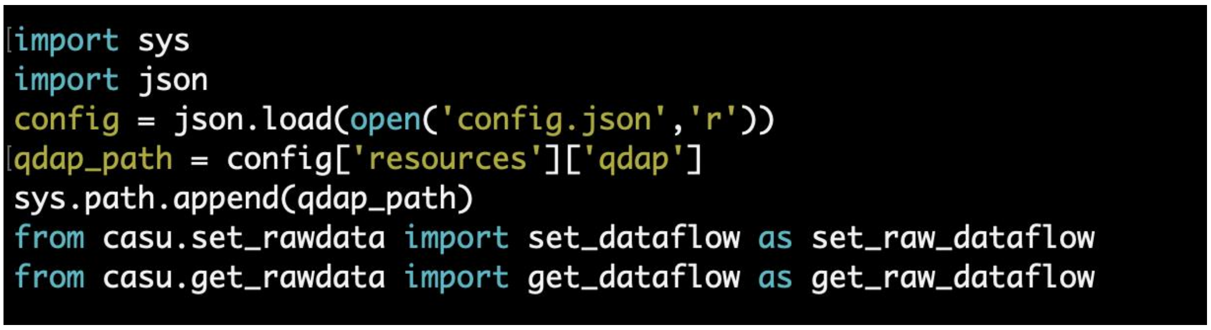
1. `config.json` - global parameters applicable to all nodes. These include the various connection profiles detailing the target database (host, port, database name). End users should not generally update this file.
2. `node.json` – node-level parameters applicable only to the node. This configuration file allows end-users to customise and configure some aspects of their interaction with the OR. This includes the logging level (if running within a Python environment) and paths where data should be transferred to.

9.1.2 Integration with existing Python code

If users wish to call the provided Python functions within their existing code, they should ensure:

1. The Python environment the existing code is running under satisfies the requirements provided in requirements.txt
2. The location of the cloned repository is
   a. added to `$PYTHONPATH` or
   b. appended to `sys.path` at runtime

For (2), CASU tend adopt (b), by reading in the path from a JSON-format configuration file when other toplevel modules are imported:



```
import sys
import json
config = json.load(open('config.json','r'))
qdap_path = config['resources']['qdap']
sys.path.append(qdap_path)
from casu.set_rawdata import set_dataflow as set_raw_dataflow
from casu.get_rawdata import get_dataflow as get_raw_dataflow
```

[Deployment instructions and Python integration subject to updates prior to operations]

9.2 Calls to the OR database

The operational dataflow requires two types of interaction with the OR proc_state table:

- poll the OR for new data

- update the OR to indicate an action has been performed

These two actions respectively are performed with a "get" and a "set" within the recipes available to the end-user, with recipe names identified by this action and the associated data product, e.g.:
`get_l1data` – poll the OR for availability of an L1 night to process
`set_l1data` – update the OR proc_state table for L1 data

In both cases, terminal-line help on the options can be found with the *-h* switch, eg:
```
./set_l1data.py -h
```

Full documentation on the function can be retrieved with the *-doc* switch:
```
./set_l1data.py -doc
```

We detail below the specifics of the "get" and "set" actions available across all relevant data products.

9.2.1 Availability of data (for processing, for transfer)

If a node requires input data to work on, or a data product to transfer, it must know if that data is available, and if so, where it is stored. If the node is off-site (e.g. the archive facility), then enough information must be provided to be able to transfer the data to the node, and optionally ensure that the transfer did not corrupt the data.

9.2.1.1 Terminal recipe

Running the core "get" command will provide an output that indicates if a night of data is available. This can be supplemented by additional switches that provide further functionality:

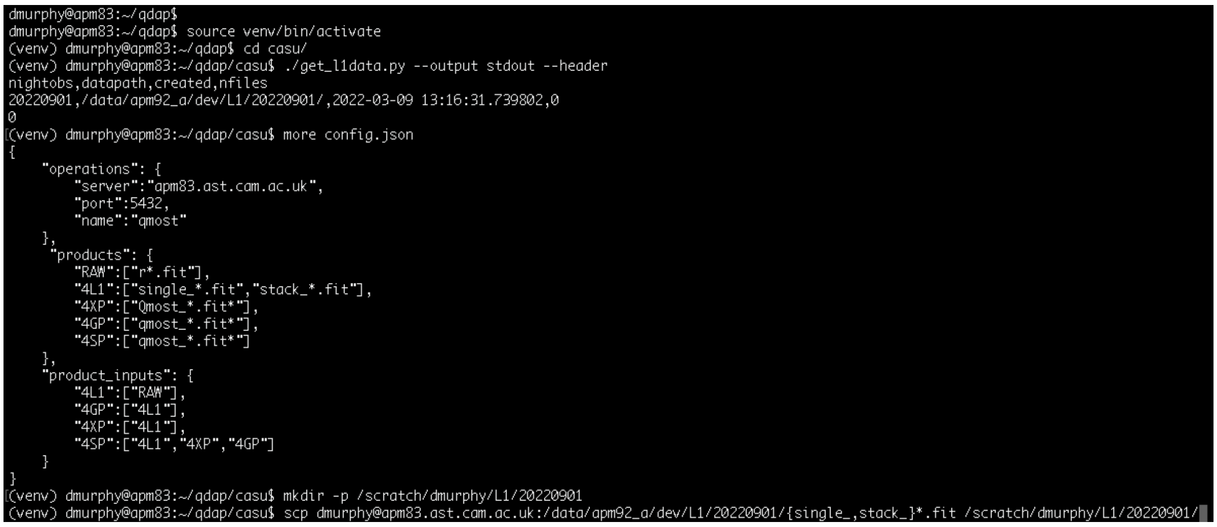
debug – provides more verbosity for the operation, including queries made to the database;
transfer – if data are available, transfer them. This will use the path described in the node-level configuration. The types of files that will be transferred are described in the "products" section of the global config.json file;
validate (only where transfer is set) - if data are available, once it has been transferred perform a series of checks to ensure the files are not corrupted. This includes retrieving datasum and checksum information from the OR for cross-checking against the files that have been transferred.

There is no requirement to use the functional switches above. Where data are available for transfer, the output of the recipe provides enough information to construct an *scp* or *rsync* command manually and perform the transfer. Under this approach, validation of the data products transferred is the responsibility of the end-user.

We demonstrate this below, where a would-be end user performs the following:

- Activates the WDAP virtual environment

- Changes into their node's directory
- Runs a "get" recipe to check if an L1 night is available to work on
- Looks at the settings in the WDAP configuration
- Creates a directory to copy the available data into
- Constructs an *scp* command from the provided information to transfer over the data



```
dmurphy@apm83:~/qdap$
dmurphy@apm83:~/qdap$ source venv/bin/activate
(venv) dmurphy@apm83:~/qdap$ cd casu/
(venv) dmurphy@apm83:~/qdap/casu$ ./get_l1data.py --output stdout --header
nightobs,datapath,created,nfiles
20220901,/data/apm92_a/dev/L1/20220901/,2022-03-09 13:16:31.739802,0
0
(venv) dmurphy@apm83:~/qdap/casu$ more config.json
{
    "operations": {
        "server":"apm83.ast.cam.ac.uk",
        "port":5432,
        "name":"qmost"
    },
     "products": {
        "RAW":["r*.fit"],
        "4L1":["single_*.fit","stack_*.fit"],
        "4XP":["Qmost_*.fit*"],
        "4GP":["qmost_*.fit*"],
        "4SP":["qmost_*.fit*"]
    },
    "product_inputs": {
        "4L1":["RAW"],
        "4GP":["4L1"],
        "4XP":["4L1"],
        "4SP":["4L1","4XP","4GP"]
    }
}
(venv) dmurphy@apm83:~/qdap/casu$ mkdir -p /scratch/dmurphy/L1/20220901
(venv) dmurphy@apm83:~/qdap/casu$ scp dmurphy@apm83.ast.cam.ac.uk:/data/apm92_a/dev/L1/20220901/{single_,stack_}*.fit /scratch/dmurphy/L1/20220901/
```

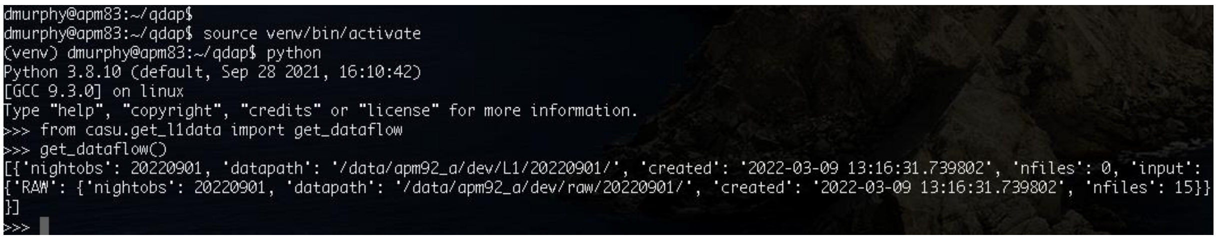
For a "get" recipe, the output is universal across all recipes, nodes and datatypes, namely:

```
nightobs,datapath,timestamp,nfiles
```

9.2.1.2 Python recipe

A similar command can be run under Python, with in principle the same switches (as arguments to the function). However, it is more useful to return the value as a dict under Python, rather than (for example) having to scrape STDOUT output via pipe or read in a CSV file. The below demonstrates:

- enabling the WDAP virtual environment (this is not required if the WDAP requirements have already been satisfied by the user)
- starting the Python interpreter
- importing the recipe
- running the recipe (with the resultant dictionary shown)



```
dmurphy@apm83:~/qdap$
dmurphy@apm83:~/qdap$ source venv/bin/activate
(venv) dmurphy@apm83:~/qdap$ python
Python 3.8.10 (default, Sep 28 2021, 16:10:42)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from casu.get_l1data import get_dataflow
>>> get_dataflow()
[{'nightobs': 20220901, 'datapath': '/data/apm92_a/dev/L1/20220901/', 'created': '2022-03-09 13:16:31.739802', 'nfiles': 0, 'input':
{'RAW': {'nightobs': 20220901, 'datapath': '/data/apm92_a/dev/raw/20220901/', 'created': '2022-03-09 13:16:31.739802', 'nfiles': 15}}
}]
>>> 
```

We note here some differences in output between the Python recipe and the terminal-run recipe. The same outputs are provided as for the terminal, but for nodes that will process the data, a nested "input" dictionary provides that relevant data that should be used to proceed with the processing job.

This is not provided for the archive recipes, as no knowledge of the input data location is required.

9.2.1.3 SQL equivalent

For processing nodes, the *flowstate* is the critical dataflow column:

```
SELECT nightobs, datapath, created, nfiles FROM proc_state WHERE product='L1' AND flowstate='PENDING' AND procstate != 'PROCESSED'  ORDER BY nightobs ASC LIMIT 1;
```

For external nodes (eg. the archive), the *wastate* is the critical dataflow column (and it is assumed that all other *procstate*s / *flowstate*s are irrelevant):

```
SELECT nightobs, datapath, created, nfiles FROM proc_state WHERE product='L1' AND wastate='PENDING' ORDER BY nightobs ASC LIMIT 1;
```

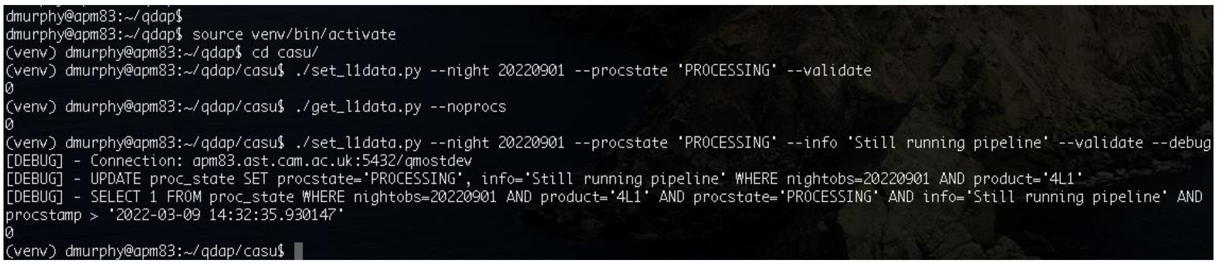
9.2.2 Updating the state of data (processing, transfers)

Once a node has started work (be that analysis of the data or the transfer of data to an external node), then the OR should be informed that this event is underway. As described in the above sections, this involves a change to the proc_state *wastate* (for archive operations) or *procstate* (for data processing operations).

9.2.2.1 Terminal recipe

This is achieved by running the "set" analogue of the "get" recipe described in Section 9.2.1. We demonstrate this below, where a would-be user running the L1 pipeline tells OR that processing of the data has started:

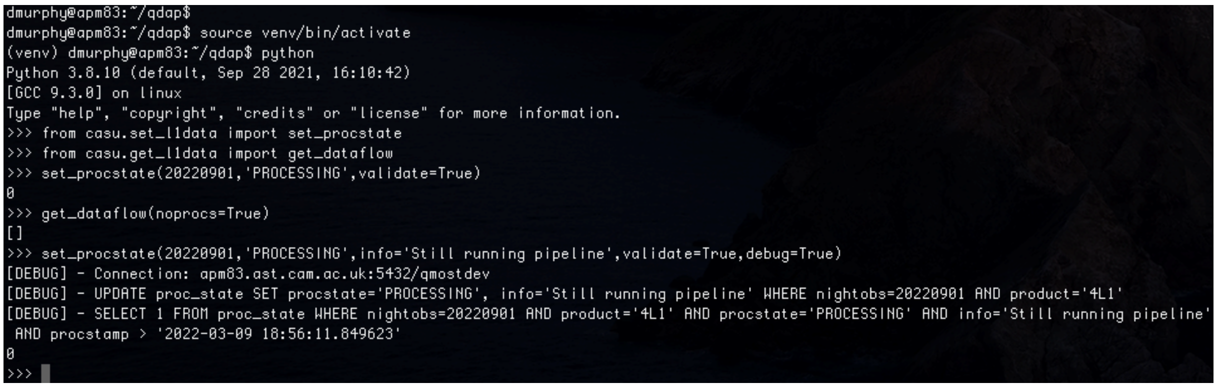
- Initialising the virtual environment in the same way as above
- Changes to the node directory
- Runs the "set" recipe to signal that processing has started, based on the output of the above "get" recipe, with a validation switch to ensure the update was successful
- Check that this night no longer appears available ("get" recipe) with the *noprocs* switch
- Issues a further update, this time with an information update (Section 8.5.1.1). This time, including the *debug* command to see what has been sent to the proc_state table.



```
dmurphy@apm83:~/qdap$
dmurphy@apm83:~/qdap$ source venv/bin/activate
(venv) dmurphy@apm83:~/qdap$ cd casu/
(venv) dmurphy@apm83:~/qdap/casu$ ./set_l1data.py --night 20220901 --procstate 'PROCESSING' --validate
0
(venv) dmurphy@apm83:~/qdap/casu$ ./get_l1data.py --noprocs
0
(venv) dmurphy@apm83:~/qdap/casu$ ./set_l1data.py --night 20220901 --procstate 'PROCESSING' --info 'Still running pipeline' --validate --debug
[DEBUG] - Connection: apm83.ast.cam.ac.uk:5432/qmostdev
[DEBUG] - UPDATE proc_state SET procstate='PROCESSING', info='Still running pipeline' WHERE nightobs=20220901 AND product='4L1'
[DEBUG] - SELECT 1 FROM proc_state WHERE nightobs=20220901 AND product='4L1' AND procstate='PROCESSING' AND info='Still running pipeline' AND
procstamp > '2022-03-09 14:32:35.930147'
0
(venv) dmurphy@apm83:~/qdap/casu$ 
```

Note here that for each "set" recipe, at the terminal there was a return value of 0. This indicates that the "set" recipe correctly updated the proc_state table. If the set recipe was run without the *validate* switch, the return value could still be 0, but this only indicates that no exceptions were raised during the running of the recipe. When *validate* is specified, this return value also depends on the validation query (an entirely separate server query) returning the expected result.

9.2.2.2 Python recipe



```
dmurphy@apm83:~/qdap$
dmurphy@apm83:~/qdap$ source venv/bin/activate
(venv) dmurphy@apm83:~/qdap$ python
Python 3.8.10 (default, Sep 28 2021, 16:10:42)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from casu.set_l1data import set_procstate
>>> from casu.get_l1data import get_dataflow
>>> set_procstate(20220901,'PROCESSING',validate=True)
0
>>> get_dataflow(noprocs=True)
[]
>>> set_procstate(20220901,'PROCESSING',info='Still running pipeline',validate=True,debug=True)
[DEBUG] - Connection: apm83.ast.cam.ac.uk:5432/qmostdev
[DEBUG] - UPDATE proc_state SET procstate='PROCESSING', info='Still running pipeline' WHERE nightobs=20220901 AND product='4L1'
[DEBUG] - SELECT 1 FROM proc_state WHERE nightobs=20220901 AND product='4L1' AND procstate='PROCESSING' AND info='Still running pipeline'
 AND procstamp > '2022-03-09 18:56:11.849623'
0
>>> 
```

We note here under the Python recipe that the get_dataflow function returns an empty list – this indicates there are no rows in the database that match the query the recipe made. Similarly, the return values for each of the 'set' recipes was 0, indicating that the update to the proc_state table was successful.

9.2.2.3 SQL equivalent

For processing nodes, the *procstate* is the dataflow column that should be updated:
UPDATE proc_state SET procstate='PROCESSING', info='Still running pipeline' WHERE nightobs=20220901 AND product='L1';

For external nodes (eg. the archive), the *wastate* is the dataflow column to be updated. For example, to indicate that the WAS has started transferring the data over to the archive:

UPDATE proc_state SET wastate='TRANSFERRING' WHERE nightobs=20220901 AND product='L1';

9.3 Rescinding updates

Made a terrible mistake? That's OK! Actionable events driven by the end-user, namely changing *procstate* to one of:

- NONE
- PROCESSED
- FAILED

or for external nodes changing a dataflow state (*wastate* for example) to one of:

- TRANSFERRED
- VALIDATED

will not be processed immediately: there is a grace period (defined in the proc_daemon table, set to 2 minutes at present). Within this grace period, the OR will not act on the event (but the value will be updated in the proc_state table).

The status update can be rescinded by running the "rescind" WDAP recipe or calling the rescind Python function:

```
./rescind -nightobs 20220901
```

The dataflow status will then revert to the previous values (and the OR will not act), but only if the request to rescind was within the grace period. No "countdown timer" is provided, and users should check the relevant timestamp to gauge how much time they have remaining to issue a rescind order. Only the last change can be undone.

Consider this functionality as "breathing space". If you update in haste, you may rescind at leisure. The operational grace period is TBD following engagement with stakeholders. We note also that backend operations might also be subject to similar grace periods.

9.3.1 SQL equivalent

The L1 pipeline issued a procstate change night 20220901 from "PROCESSED" → "NONE". This was issued before additional checks were made, so they wish to rescind the retraction event.

- Identify the *id* (e.g. 30) of the change from the proc_log, then:

```
UPDATE proc_log SET rescinded=True WHERE id=30;
```

- Perform a validation check as appropriate to ensure reversion, for example:

```
SELECT 1 FROM proc_state WHERE nightobs=20220901 AND product='L1' AND
procstate="PROCESSED";
```

10 DETAILED NODE USE CASES

This section details how each node interfaces with the OR to facilitate the flow of data. In this section, we demonstrate **examples** of running the dataflow recipes from the terminal. It is assumed that the Python virtual environment is activated and that the node is in the directory containing their recipes. Other approaches beyond these examples are of course possible, and implementation is the responsibility of the nodes.

10.1 L1 pipeline

[WIP]

10.2 L2 pipeline

[WIP]

10.3 WAS archive

There are two broad categories of data transferred to the WAS:

- Regular nightly release of processed data products
- Irregular release of catalogue and calibration data

In checking for data availability and updating of dataflow state, the WAS specifically uses the *wasate* column in the proc_state table. This “WEAVE Archive State” value defines the availability of data, and subsequently the stage of data transfer and validation the data is in with respect to the archive.

10.3.1 Regular data releases

As described in [RD30], nights are released from CASU to WAS only when all data products (L1, L2, CS) have been generated, received at CASU and successfully ingested. The WAS may check for any of these products at any point, but we note that we will release all data products for a given night at the same time.

Two options are therefore open to WAS for checking the availability of data:

1. Check for availability of just L1 data: **it is implied that L2 and CS data are available if the L1 check returns a viable night**.
2. Check for L1, L2 and CS products in a cyclic manner: **every period (TBD by WAS), check for each data product.**

WAS should therefore run at minimum one “get” recipe to determine the availability of data to transfer: In this example, we will adopt option (2) above. Whilst these can (and should) be integrated into a Python workflow, we adopted the simplified alternative or running hourly cronjobs calling each of the 3 recipes "from the terminal" (via cron in this case).

If we assume the DAP has been cloned to:
`/wdap`

and the virtual environment to:
`/wdap/venv/`

Then the crontab entries would be:

```
0 * * * * /wdap/venv/bin/Python /wdap/was/get_l1data.py –transfer --validate >> /wdap/l1cron.log 2>&1
0 * * * * /wdap/venv/bin/Python /wdap/was/get_l2data.py –transfer --validate >> /wdap/l2cron.log 2>&1
0 * * * * /wdap/venv/bin/Python /wdap/was/get_csdata.py –transfer --validate >> /wdap/cscron.log 2>&1
```

Each of these commands will poll the OR, determine if a night is available for download, and then transfer the files from CASU to the location specified in the WAS `node.json` file, and perform a simple validation on the transferred files.

Once WAS have verified that the transfers were completed, and validation passed, they should update the OR to signal that the transfer has completed and the data have been validated. We will assume (at the moment) this is done manually by the end-user, but in reality these recipes should be run within a workflow that identifies when data have been picked up and provides updates to the dataflow process. If the night in question was 20220901:

```
./set_l1data.py --night 20220901 --wastate VALIDATED --verify
```

This sets the archive's dataflow state to "VALIDATED" for the night 20220901, and performs a verification that this update has been made in the OR database. **This serves as notice to CASU that the data have been fully transferred downstream**.

If WAS decided to manually transfer the files and not make use of the recipe's built-in validation, then they should, as a courtesy to other users (including those observing the data processing and dataflow status on the OR web page), provide updates to the dataflow manually, namely:

Upon starting transfer of L1 data from CASU to WAS:

```
./set_l1data.py --night 20220901 --wastate TRANSFERRING --verify
```

Once transfer of L1 data from CASU to WAS has completed:

```
./set_l1data.py --night 20220901 --wastate TRANSFERRED --verify
```

When WAS validation of the products starts:

```
./set_l1data.py --night 20220901 --wastate VALIDATING --verify
```

Upon (successful) completion of the validation:

```
./set_l1data.py --night 20220901 --wastate VALIDATED --verify
```

Once the L1 data have been validated, the focus should shift to the other data products. The equivalent above calls should also be made for the L2 and CS data products.

It is entirely possible that a bulk-reprocessing of data has occurred, and as a result multiple nights are available at the same time (under a retraction event). The WAS should therefore be able to handle cases where the results of these “get” recipes return nights that have previously been ingested by the archive. Communication of *why* the night (or nights) are available once again is outside of the scope of the recipes and should be discussed with the relevant node.

It is possible for a retraction of just one data product for a night, but (as per Section 8.6.2.3) cascade retractions **will** provide multiple product types for WAS to transfer. Specifically:

- CS: can be retracted with no further impact
- APS: cascade retraction to CS
- CPS: cascade retraction to APS and CS

10.3.2 Irregular data releases

Although the catalogue (proc_state *product*=’CAT’) and calibration (proc_state *product*=’CAL’) data are not released on a nightly basis, polling the OR to determine data availability is the same procedure as described above. These files will be put into a nightobs directory, and an entry will be made in the proc_state table. In both cases, the equivalent recipes can be run on a frequency (TBD) to check and optionally transfer the data. The archive should similarly provide updates on the data transfer state and validation state for exchange of these data products.

11 FULL DB SCHEMA FOR EACH DATAFLOW TABLE