Validation of the laboratory's own software

The code the laboratory wrote and runs: its components, the functions that affect a result or a record with the risk of each and the suite that covers it, the version validated by the fingerprint of every file, the acceptance rule and the change control.

Not yet approved

No approval of this revision is on file. This plan states the validation the laboratory performs on its own software: a version is validated when python make.py check passes on it and this document is approved at the fingerprint that names it.

Document		Laboratory	
Number	SV-01	Name	Cupel Laboratories
Revision	1, first issue	Address	2599 Speakman Dr Mississauga, ON L5K 1B1
Status	Not yet approved		
Effective	Not yet in effect; takes effect when approved	Country	Canada
		Not on file	Not yet published: legal entity, postal address, telephone and country
Applies to	Every program in the project folder		
Supersedes	Nothing; first issue	Web	cupel.ca
Fingerprint	1bdf 50f0 fa90 2ea2	Email	help@cupel.ca

1 Purpose

To validate the software the laboratory itself wrote and runs, for its intended use: what it must do, the functions that affect a result or a record, the risk if each is wrong, the test that covers it, the version validated (identified by the fingerprint of every file) and the rule by which a change is authorised, tested and recorded before use (ISO/IEC 17025:2017 7.11.2).

2 Scope: the system and its components

The system is the Python code in the project folder, run with Python 3.10 or later and the packages make.py doctor checks (reportlab, the certificate and intake PDFs; fonttools, cutting the embedded fonts; brotli, reading the woff2 fonts; PyMuPDF, rasterising PDFs for review; Pillow, image handling in the PDF audit; pdfminer.six, test_pdf.py reads the PDFs back; PyPDF2, test_pdf.py reads the PDF structure); node and headless Chrome for the site's checks. It keeps no database server, no account system and makes no outbound network call: the register is one SQLite file, the backend answers on this computer only, and every email is prepared as a file for a person to send. The instruments' own data systems are outside this plan (SOP-09).

Component	Files and what they do	What it reaches
The calculation model	lab/results.py, chem.py, damm.py: validate, compute, the uncertainty and the decision rule; the masses from the formula; the check digit	every number on every certificate, page and email
The results backend	lab/entry_server.py, lab/serve.py: the forms, the audit trail, logging, review, the notice, issue, revision, withdrawal, holds, the case logs	every record the laboratory keeps, and who is named on it
The register	register.py: reserve, issue, correct, withdraw; the two hashes; the guards	every accession and every published row
The certificate composer	lab/coa_report.py, coa_pdf.py, lab/figures_pdf.py, pdf_fonts.py, qr.py: the certificate's pages and figures from the computed set	the paper the reader holds
Instrument import	lab/cds_import.py: instrument exports read into the set's shapes, with provenance	the raw values a result is computed from

Scope: the system and its components, continued

Orders, emails, labels and the intake sheet	lab/orders.py, lab/emails.py, lab/labels_pdf.py, intake_pdf.py: the order read in, every email, the vial labels, the intake sheet	what the sender receives and what is on the vial
The site and the vendor kit	build.py, lab/vendor_kit.py, theme.py, contrast.py, figures.py: every public page, the report pages, the badges and the lot index	what a buyer reads
The controlled documents	lab/sop_pdf.py, lab/validation.py: the quality manual, the procedures, the protocols and reports	what the laboratory says it does
Quality control, metrics and the review	lab/qc_trend.py, lab/metrics.py, lab/management_review.py: the control charts and limits, the service in numbers, the review's input and record	what management decides on
Integrity, backup and the checks	lab/integrity.py, lab/backup.py, lab/drill.py, make.py: re-verification from disk, archives and restores, the check runner	whether any of the above can be trusted after the fact

3 Intended use

- To take a sample from an order to a published certificate with every number computed from the recorded data and nothing typed twice (ARCHITECTURE.md, from an order to a published certificate).
- To refuse what the procedures refuse: a set short of its design, an unnamed actor, a reviewer who is the analyst, an issue before the notice an adverse report needs, a change to an issued report.
- To keep every record so that it can be re-verified from disk: the register that only grows, the chained trails, the two hashes on every certificate, the archives and the restore drill.
- To say on every surface what is not yet in place, from data.json, rather than invent it.

4 The functions that affect results and records, and their evidence

Each function with where it lives, the risk if it is wrong, and the suite whose text exercises it (read from the suite files; a function no suite names is an open point).

Function	Where	Risk if wrong	Evidence
Accept or refuse a result set: shape, counts, equipment, bookkeeping	lab/results.py validate	A set short of its design issues	lab/test_results.py (names validate()
Compute every number a certificate prints	lab/results.py compute	A wrong purity, content, water or endotoxin value	lab/test_results.py (names compute()
Expanded uncertainty and effective degrees of freedom	lab/results.py expanded_uncertainty	A wrong U, or coverage below 95 percent	lab/test_results.py (names expanded_uncertainty)
The statement of conformity on the rounded value	lab/results.py decision_text	A wrong verdict against a claim or a specification	lab/test_results.py (names decision_text)
Monoisotopic and average masses from the formula	chem.py mono	A wrong identity criterion	chem.py (names mono(); lab/test_analytes_reference.py (names mono()
The accession's check digit	damm.py issue	A number that cannot be checked, or two that collide	damm.py (names issue(, damm); test_parity.py (names issue(, damm)
Reserve, issue, correct and withdraw a number; the guards	register.py reserve; register.py issue; register.py correct; register.py withdraw; register.py guard_statements	A report under the wrong number, a deletion, an edit after issue	register.py (names reserve(, correct(, withdraw(, guard_statements)
The two hashes of every report	register.py payload_hash; register.py record_hash	A served file that does not match what was issued	register.py (names payload_hash, record_hash)

The functions that affect results and records, and their evidence, continued

The audit trail and its chain	lab/entry_server.py audit; lab/entry_server.py verify_audit	A change nobody can see, or one that names nobody	lab/entry_server.py --selftest (names verify_audit, audit(); lab/test_e2e.py (names audit()
Review by a second person and the gate at issue	lab/entry_server.py review_refusal; lab/entry_server.py issue_blockers; lab/entry_server.py do_issue	One person issues a report alone, or an unreviewed one	lab/entry_server.py --selftest (names review_refusal, issue_blockers, /review/, /issue/); lab/test_e2e.py (names /review/, /issue/)
The notice before an adverse report	lab/entry_server.py notice_state	An adverse report published without the sender told	lab/entry_server.py --selftest (names notice_state, /notice/); lab/test_e2e.py (names /notice/)
Instrument files read into the set	lab/cds_import.py import_bytes	A value read from the wrong column, or without its provenance	lab/test_cds_import.py (names import_bytes)
The certificate composed from the computed set	lab/coa_report.py build_twice	A page that does not say what the record says	test_pdf.py (names build_twice, coa_report)
Every email composed from facts	lab/emails.py TEMPLATES	A sender told something the record does not hold	lab/test_emails.py (names TEMPLATES)
The order read from the message	lab/orders.py parse_order	A draft for the wrong sample, or a sender's detail in a set	lab/test_orders.py (names parse_order)
Control charts and the limits record	lab/qc_trend.py establish	A run judged against the wrong limits, or limits frozen by nobody	lab/test_qc_trend.py (names establish()
The service in numbers	lab/metrics.py compute	A published figure that misstates the service	lab/test_metrics.py (names compute()
The controlled documents from their sources	lab/sop_pdf.py build_all	A procedure that disagrees with the certificate it governs	lab/test_sop.py (names build_all); lab/test_validation.py (does not name it)
Re-verification from disk	lab/integrity.py run_checks	Damage that stays unseen	lab/integrity.py --selftest (names run_checks, selftest)
Backup, verify and restore	lab/backup.py backup; lab/backup.py verify; lab/backup.py restore	Records that cannot be recovered	lab/backup.py --selftest (names restore(, verify(, selftest); lab/drill.py (names restore(, verify()

5 The evidence: the test suites

python make.py check runs every suite in this order and fails on any failure: register, test_analytes_reference, test_cds_import, test_e2e, test_emails, test_launch_facts, test_launch_preview, test_management_review, test_metrics, test_orders, test_qc_trend, test_report_api, test_results, test_safety, test_sop, test_tracking, test_validation, test_vendor_kit, test_worksheets, selftest, build, integrity, integrity_selftest, backup_selftest. The expected values of the arithmetic suites are closed forms, tables and sums written out in the test, not the model's own functions; the document and certificate suites read the PDFs back with an independent reader; the end-to-end suite drives a copy of the backend as a black box in a temporary copy of the project; the self-tests copy the project and damage the copy twenty ways. Each suite's own fingerprint identifies the evidence as this document identifies the software.

Suite	What it checks	Size	Fingerprint
lab/test_analytes_reference.py	Proof suite for lab/analytes_reference.json	script: its own count	7ec6 be32 9aa9 c495
lab/test_cds_import.py	Proof suite for lab/cds_import.py	script: its own count	650d 3f0e 5077 615f

The evidence: the test suites, continued

lab/test_e2e.py	the backend lifecycle, end to end, as a black box	1 test function	a00d 22a8 3800 464c
lab/test_emails.py	Checks on lab/emails.py	47 test functions	d9f9 c1dc 7fdc fa5c
lab/test_launch_facts.py	its docstring's first sentence, not quoted here: it speaks of a test copy marked accredited, which is no claim about the laboratory	13 test functions	ee59 b713 1bff 253b
lab/test_launch_preview.py	the launch-day rehearsal (lab/launch_preview.py) in its quick mode, and the helpers it is built from	6 test functions	c12e e950 44eb e3d0
lab/test_management_review.py	The management-review input and the record of a review held (lab/management_review.py), on a temporary project	script: its own count	9549 eb7a b5fd aeef
lab/test_metrics.py	the laboratory metrics, checked against figures computed by hand	29 test functions	0b99 99fd f804 809a
lab/test_orders.py	Checks on lab/orders.py	49 test functions	23a6 75d0 efb5 51fe
lab/test_qc_trend.py	Quality-control trending (lab/qc_trend.py), checked against hand arithmetic	script: its own count	04f7 f4aa 8b13 0ed5
lab/test_report_api.py	the report API (lab/report_api.py) end to end, on a temporary copy of the project	1 test function	d75f cb77 54cf 44bf
lab/test_results.py	Tests for lab/results.py against the three example result sets	16 test functions	105f 1dec 387f e318
lab/test_safety.py	SAF-01 (lab/safety_pdf.py), audited from the file it produces and against the procedures' composed text	script: its own count	f18e b8a2 aa21 672c
lab/test_sop.py	The laboratory's controlled documents (lab/sop_pdf.py), audited from the files they produce	script: its own count	d469 b0f3 a993 5069
lab/test_tracking.py	Tests for lab/tracking.py, the private order-status page, and for the token lab/orders.py keeps for it:	8 test functions	433a 7662 4294 3fbe
lab/test_validation.py	Method validation (lab/validation.py), checked against arithmetic done independently of it	script: its own count	6abb 6134 7df5 09e8
lab/test_vendor_kit.py	Checks on lab/vendor_kit.py	19 test functions	d716 849c 62dc df6d
lab/test_worksheets.py	The bench worksheets (lab/worksheets_pdf.py), audited from the files they produce and against the result set and the backend's form	script: its own count	e34b a612 a689 b124
register.py	The register: append-only storage for issued reports	run first by the check and by the build	90d6 0ee4 f86e cd4b
chem.py	Mass arithmetic for the analyte pages	run by the build	3c93 61aa 529f 1727
damm.py	Damm check digit, order 10	run by the build	ba17 5ee8 565a 8420
test_parity.py	Differential test: the inline browser validator against damm.py	run by the build	d9fa a94a 8e31 ccac
contrast.py	Compute every contrast ratio on the site from the tokens, and assert it	run by the build	06a9 4f94 8a92 448b
test_pdf.py	Audit our own certificates with the test that failed a competitor's	run by the build	3e66 589a 2735 3e57
lab/entry_server.py --selftest	the module's own self-test	script: its own count	a200 90e1 0084 1968
lab/integrity.py --selftest	the module's own self-test	script: its own count	aa3c 07f2 4d95 5ddd
lab/backup.py --selftest	the module's own self-test	script: its own count	9669 597b 2e81 bade

6 Version identification

The version validated is the set of files below, each identified by the SHA-256 of its bytes (first 16 hex digits). These fingerprints are part of this document's text, so this document's own fingerprint changes the moment a line of any of them changes: an approved SV-01 names exactly the software it validates, and prints "Changed since approval" after any change until it is approved again.

Version identification, continued

File	Lines	Fingerprint
lab/results.py	8771	2444 4b96 02b0 7aad
chem.py	563	3c93 61aa 529f 1727
damm.py	205	ba17 5ee8 565a 8420
lab/entry_server.py	15123	a200 90e1 0084 1968
lab/serve.py	392	9783 daff cbe5 1713
register.py	4120	90d6 0ee4 f86e cd4b
lab/coa_report.py	2780	ef45 18e5 e97c 79bd
coa_pdf.py	1017	d14d f198 bd3d b7f6
lab/figures_pdf.py	3276	b301 aeb9 b1b3 b131
pdf_fonts.py	173	82df cf45 8346 1b3c
qr.py	55	a4e7 fcef 5d46 422d
lab/cds_import.py	2636	52a4 cf44 a721 9016
lab/orders.py	3629	2170 ff74 52e7 17f1
lab/emails.py	4002	d0b0 98d5 571f b7c4
lab/labels_pdf.py	321	1235 8226 a02d ce6c
intake_pdf.py	771	5a4a dbed f51f 47c3
build.py	9034	986c 5d8a 9429 792c
lab/vendor_kit.py	798	d586 d86e 67c3 f835
theme.py	1851	3415 6977 6871 39cd
contrast.py	488	06a9 4f94 8a92 448b
figures.py	704	bc59 f7ec 5568 f72d
lab/sop_pdf.py	10365	7f18 9e64 6a76 645b
lab/validation.py	6676	bfad d762 168e aef1
lab/qc_trend.py	4225	e368 671d 0d70 70f7
lab/metrics.py	1945	3611 95ee 15ed 6b6c
lab/management_review.py	1934	80c6 86ec 3bdd b75a
lab/integrity.py	4645	aa3c 07f2 4d95 5ddd
lab/backup.py	2374	9669 597b 2e81 bade
lab/drill.py	477	8ac0 6287 d376 4916
make.py	1686	2f3e 81d3 041b 8848

7 Acceptance

7.1 The system is validated at a version when python make.py check passes every suite (exit 0) on the tree whose fingerprints this document records, and this document is approved at that fingerprint (data.json documents).

7.2 The transcript of that run (logs/make.log, which make.py writes) is the record of the validation run; it is kept with the quality records under the date and this document's fingerprint.

7.3 A failing suite is nonconforming work (SOP-08): the build refuses to publish while its proofs fail, and no change goes into use until the check is green again.

7.4 The dress rehearsal (lab/rehearsal.py: an order, five samples, three reports, a withdrawal, a hold, the build and the integrity check in a throwaway copy) and the launch preview (lab/launch_preview.py) are run before the first real report and after any change to the backend or the register.

8 Change control

ISO/IEC 17025:2017 7.11.2

8.1 A change to any file in the version table is a new version: it is made in the code, tested by python make.py check, and recorded by approving this document again at its new fingerprint; nothing else records a software version.

Change control, continued

8.2 A change to lab/results.py also moves the fingerprint of every procedure that prints one of its constants (SOP-02 to SOP-06, QM-00), and of the protocols that read them: the fingerprints enforce that a changed model re-opens the documents that depend on it, and each is approved again or stands as changed.

8.3 A change that corrects a fault is a case in the corrective-action log (SOP-08) with the fault, the fix and the suite that now catches it; the verification is the next green check.

8.4 No change is made on the live project while a report is between review and issue; the check lock (logs/check.lock) serialises the suites that write the live tree.

8.5 Configuration: the packages and the Python version above; a change of either is a change to the system and is checked the same way (python make.py doctor, then check).

9 Data integrity controls

ISO/IEC 17025:2017 7.11.3

- The register only grows: its guards refuse deletion and alteration through the database, and lab/integrity.py check_register detects a dropped or altered guard by comparing every trigger by name and text (refusal, then detection).
- Every change to a set is one line in a chained trail (lab/entry_server.py audit); every issued report carries two hashes recomputed by the build, the browser and the integrity check.
- Every write is atomic (a temporary file, fsync, rename) and under one lock; the backend answers on this computer only, checks the Host of every request and a form token on every change.
- Backups are verified archives with a manifest of hashes; the restore drill rebuilds the site from an archive in a temporary folder and compares it byte for byte with the live one (lab/drill.py).

10 Deviations and open points

- A departure from this procedure, planned or found afterwards, is nonconforming work and is handled under SOP-08. One that bears on a result is stated on the certificate, in the open point or the notes of the section concerned, never left out.
- No validation run is recorded against an approved version: this document is not yet approved.
- The backend authenticates nobody: the actor is a typed name (QM-00 7.11); the signatures a certificate prints rest on it.
- The audit trail's field diff summarises a long value (lab/entry_server.py summarise), so the pre-edit trace or peak list is not retained in full (CRITIQUE_17025 F14).
- No clock-synchronisation rule between the backend host and the instrument PCs (SOP-09).
- A function above whose evidence reads "does not name it" is exercised, if at all, only through another function; the next revision of that suite names it.

11 Records

- This document at each approved fingerprint; the version table in it.
- logs/make.log, the transcript of each check run; the check lock's log of what ran when.
- The corrective-action cases that record faults and their fixes.
- The rehearsal's and the launch preview's reports, when run.

12 Revision history

Revision	Change	Approval
1	First issue.	Not yet approved

The fingerprint in the control block is the SHA-256 of this document's text as composed, first 16 hex digits. A change in any controlled source changes it; a copy whose fingerprint differs from the approved one is not the approved text, and a change is approved as a new revision.

Cupel holds no ISO/IEC 17025 accreditation; results are not accredited results.

This document is published so that a chemist can judge how a result was produced and what it can and cannot show; it is not an accredited procedure and claims no conformity with any standard it cites.

The current revision of every laboratory document is the one published at cupel.ca/methods. A printed copy is uncontrolled: compare its fingerprint with the published one.

Current revisions

End of document