Format Validation (CPP-010)

CPP-Identifier	CPP-010
CPP-Label	Format Validation
Author	Bertrand Caron
Contributors	Kris Dekeyser
Evaluators	Matthew Addis, Maria Benauer, Laura Molloy
Date of edition completed	29.08.2025
Change history	**Comments**
Version 1.0 - 29.08.2025	Milestone version

1. Description of the CPP

The TDA validates *Files* against *File* format specifications.

Inputs and outputs

Input(s)		
Data	*File*	
Metadata	*Technical metadata* (File format identifier)	
Documentation / guidance	File format specification	
	File format policy - Validation	
Output(s)		
Metadata	*Provenance metadata* (Validity status)	
	Optional	Warnings and errors
		Technical metadata

Definition and scope

Format validation is the process of checking a *File's* structure against the specifications of the format it purports to be, or it was identified as in **File Format Identification** (CPP-008). In addition to file structure, validation generally encompasses internal integrity checks to verify that the *File* is not truncated and that each of its components are intact (e.g. by using internal *Metadata* like CRC checksums of the frames, or size of the data streams).

Erroneous format structures have many different causes. Some examples are listed below:

- Improper behaviours of creation software (e.g. InDesign, in some previous versions, was known to export corrupted PDFs);
- Partial or failed transfers (e.g. digital photographs transferred through an unstable Bluetooth connection);
- Edition with a different software than the creation tool (e.g. PDFs produced with pdfTeX, then edited with Quark XPress).

The TDA validation policy should define the following aspects for each format:

- Tool(s) used to validate the *File*. Since validation tools may produce false positives and false negatives, combining multiple tools - if possible - is suggested depending on the complexity of the format[1];
- Tool settings (e.g. strict or relaxed mode, verbosity, etc.);
- Severity rating of the issue;
- Guidelines for validation error handling.

The scope of format validation is limited to the conformance of the *File* against the structures as described in the format specification. It does not assess the quality of the information

[1] PDF, in particular, is a very complex format for which validation tools may return very different results.

conveyed by the *File*. Thus, it is only one aspect of quality assurance, which can be automated or done manually. Although most of the Format Validation process described in this CPP can be automated, human intervention is mandatory if the validity status of the *File* is “invalid”.

Some tools (e.g. JHOVE[2]) follow the XML validation logic and differentiate “well-formedness” (i.e. the *File* conforms to purely syntactic requirements) and “validity” (i.e. the *File* also complies with semantic requirements). In addition, tools may return additional information or warnings, e.g. if unrecognised or private chunks are found in the *File*. In such cases,the output of a validation process should include some additional free-text besides a short controlled term (e.g. “well-formed and valid”, “well-formed but not valid”, “not well-formed”).

This CPP does not prescribe the action the TDA should undertake if File Format Validation returns errors, but rather includes fallback solutions, such as 1) ingesting the *File* as it is with warnings, 2) requesting a new delivery or searching for a suitable replacement, or 3) trying to repair the *File* (by triggering CPP-027 **File Repair**). One of these solutions should generally be preferred over rejecting the *Object*.

Format validation is a recommended process to detect potential risks caused by incorrect format structures. However, its application must be carefully evaluated based on organisational capacities and the digital *Objects* of investigation[3]. Depending on organisational capacities, and the control the TDA has over digital *Objects*[4], it may decide to not perform File Format Validation due to:

- The lack of suitable tool(s), in particular for proprietary formats;
- The lack of skilled staff to handle errors returned by the process.

[2] JHOVE, Open source file format identification, validation and characterisation, available at https://jhove.openpreservation.org/.
[3] See on this subject Paul Wheatley’s blog post “A valediction for validation?”, 11 October 2018, available at https://www.dpconline.org/blog/a-valediction-for-validation.
[4] The TDA may be able to require new, valid *Files* from a digitisation contractor or a depositor subject to legal deposit; on the other hand, it may be impossible for a private donor to meet the requirements for valid *Files*.

Process description

Trigger event(s)

Trigger event	CPP-identifier
Ingest	CPP-029 (Ingest)
Re-run of format validation because of the release of a new validation tool or tool version[5]	/
Re-run of format identification and discrepancy between previous and current result	CPP-008 (File Format Identification)

Step-by-step description

No	Supplier	Input	Steps	Output	Customer
1	CPP-008 (File Format Identification)	File format identifier	According to input, select the right tool(s) to perform format validation. If the provided *Metadata* is not sufficient, perform CPP-009 (Metadata extraction)	Validation tool(s)[6]	
	Optional: CPP-009 (Metadata Extraction)	File format policy - Validation			
2		Data: *File*	Apply the validation tools(s)	Raw validation output	

[5] Because validation tools may evolve, especially for complex formats, re-assessing the validity of files already in storage is advised.
[6] Format validation is generally performed by a single tool. Nevertheless, for some complex formats like PDF, TDAs may want to use more than one tool to identify possible structure errors.

		Tool: Validation tool(s); as a default, reference parsing tools associated with the format		Optionally, warnings and/or errors	
3		Raw validation output	Check validity status	Confirm Validity status (step 4)	
		Optionally, warnings and/or errors		Invalid file detected (step 3a)	
3a		Validity status	Conduct technical analysis (e.g. rendition and other uses of the *File*, comparison with the file format specification, etc.)	Results of manual tests	
		Warnings and/or errors			
		File format specification			
3b		Results of manual tests	Assess and choose one among the following options (in the given order)	Retain the *File* as it is (step 4)	
				Find another valid *Representation* (e.g. by establishing contact with the producer (step 4)	
				Trigger File Repair to correct the file structure (step 4)	CPP-027 (File Repair)
				Reject the *Object* (step 4)	
4			Document the process	*Provenance metadata*	

Rationale(s)[7] and worst case(s)

Rationale	Impact of inaction or failure of the process
Format validation may identify structural errors that would cause issues when attempting to render or transform the *File*. In particular, truncated *File* can be identified by File Format Validation.	/

2. Dependencies and relationships with other CPPs

Dependencies

CPP-ID	CPP-Title	Relationship description
CPP-008	File Format Identification	File Format Validation requires a specialised tool. In general, the result of File Format Identification is enough to determine which tool should be used.
CPP-009	Metadata Extraction	Soft dependency (i.e. may require): Depending on the precision of the format registry used in the File Format Identification process, the resulting information may be insufficient for selecting the right validation tool. In such cases, additional *Metadata* from an extraction tool may be required. For example, if an organisation uses Unix *File* as its identification tool, which does not distinguish between different PDF "flavours", and wants to validate PDF/A against the PDF/A standard. In that case, metadata extraction will be necessary to identify the conformance level and select veraPDF as the suitable validation tool.

[7] Term derived from PREMIS.

Other relations

Relation	CPP-ID	CPP-Title	Relationship description
Required by	CPP-013	Object Management Reporting	File Format Validation reports essential information on the well-formedness and validity of the *Objects*; validation errors; and data on the tools used in the process.
Required by	CPP-014	File Migration	File Format Validation should be undertaken after File Migration was performed to ensure that the target *File(s)* or *Representation(s)* are valid.
Required by	CPP-015	Emulation and Rendering Tools	In order to have a decent level of confidence in the rendering process, the file format needs to be validated.
Required by	CPP-023	Risk Definition and Extraction	Risks can be related to specific file format erroneous structures.
Required by	CPP-027	File Repair	File Repair is generally triggered by File Format Validation and is one of several ways to handle errors from this CPP.
May be required by	CPP-029	Ingest	A TDA may validate the format of the submitted *Files* in the ingest phase.
Not to be confused with	CPP-007	Virus Scanning	Both processes scan the *Files* to ensure that they are suitable for preservation. File Format Validation checks if a *File* conforms to its purported format specification (e.g. is this a valid PDF/A *File*?), while Virus Scanning checks for malware, regardless of format validity.
Not to be confused with	CPP-008	File Format Identification	File Format Identification is only about identifying the format while File Format Validation describes full scanning of the *File* to ensure it complies with the format standard.

3. Links to frameworks

Certification

Certification framework	Term used in framework to refer to the CPP	Section
CTS Link	/	CTS does not explicitly mention format validation but it is in the scope of section Quality Assurance (R10) and understood as one of the "quality control checks in place to ensure the completeness and understandability of data".
Nestor Seal Link	/	Format validation is not explicitly mentioned by Nestor Seal but is in scope of C21 "Submission Information Packages"; in particular, in the question "Which measures exist for validating the conformity of submission information packages?".
ISO 16363 Link	"checking that file formats are what they claim to be"	Section 4.1.5: "The repository shall have an ingest process which verifies each SIP for completeness and correctness"

Other frameworks and reference documents

Reference document	Term used in framework to refer to the process	Section
OAIS Link	"validation of SIP data"	This operation is also in scope of what OAIS calls "validation of SIP data" in section 2.6.3 "Producer interaction", where the standard suggests that the "validity" of SIPs are negotiated between the producer and the archive.
PREMIS Link	(Format) validation	Section: 'Non-core metadata', subsection: 'Quirks and anomalies' (p. 262). Section: 'Fixity, integrity, authenticity', p. 258.

4. Reference implementations

Example use case(s)

Format validation of born-digital sound

Institutional background	
Institution	Bibliothèque nationale de France
Hyperlink	/
Description	
Trigger event	BnF started collecting albums published by the phonographic industry in 2019, in the context of legal deposit of born-digital sound. *Objects* are collected in the form of highly standardised FLAC *Files*.
Problem statement	BnF decided to validate *Files* to identify transfer issues. However, the substantial volume of data and the considerable processing time have presented significant challenges to the performance and scalability of the validation operation.
Proposed solution	FLAC *Files* are verified with the flac command-line tool, which ensures both the overall *File*'s internal integrity and that of every individual frame within the audio stream. This is achieved through calculation of the CRC32 checksum for each frame. As this process is labour-intensive, the choice was made to do it by sampling. As the producer is generally a major label with trained professionals in the domain of audio data, the chosen error handling method is to request a new, hopefully valid *File*.

Publicly available documentation

Institution	Organisation type	Language	Hyperlink
TIB – Leibniz Information Centre for Science and Technology and University Library, Germany	National library	English	https://wiki.tib.eu/confluence/spaces/lza/pages/93608618/Ingest
	Non-commercial digital preservation service		
	Research infrastructure		
	Research performing organisation		
CSC – IT Center for Science Ltd., Finland	Non-commercial digital preservation service	English	https://urn.fi/urn:nbn:fi-fe2025040925236 (section 5.2)
		Finnish	https://urn.fi/urn:nbn:fi-fe2024051731943 (Annex 4, section 2.2.1)
Archivematica	Digital preservation system	English	https://www.archivematica.org/en/docs/archivematica-1.17/user-manual/preservation/preservation-planning/#characterization