



SLUB

Wir führen Wissen.

DIP Spezifikation für automatischen Access SLUBArchiv

SLUB Dresden

Version 1.0, 2020-12-14

Inhaltsverzeichnis

Überblick	1
Warum BagIt?	1
Aufbau DIP	2
Generelle Festlegungen	2
Bestandteile des DIP	3
Weitere Hinweise	5
Beispiel	5



Release Note 1.0 vom 2020-12-14

- Erstfassung auf Grundlage von [BagIt](https://tools.ietf.org/html/rfc8493) [https://tools.ietf.org/html/rfc8493]

Überblick

Dieses Dokument beschreibt den Aufbau eines Auslieferungspaketes (englisch: Dissemination Information Package, DIP), welches für die automatische Weiterverarbeitung zielgruppengerechter Ausspielungen (Access) von im SLUBArchiv archivierter digitaler Datenobjekte (IE) geeignet ist.

Dieses Dokument ist ein Teil der Übernahmespezifikation für das SLUBArchiv. Zur Übernahmespezifikation gehören die folgenden Dokumente ^[1]:

- In der **Übernahmevereinbarung** zwischen SLUB und Dienstnehmer sind die Daten, Ansprechpartner und organisatorischen Randbedingungen beschrieben. Dies schliesst die zu verwendenden Handreichungen für Dateiformate bzw. Objektgruppen mit ein.
- In der **Langzeitarchivfähige Dateiformate** sind die Formate aufgeführt, die die SLUB als potenziell archivfähig bewertet und für die die Funktionalitäten der Formaterkennung, Formatvalidierung und Metadatenextraktion in einem ausreichenden Maß durch das SLUBArchiv gewährleistet werden könnten. Die genaue Festlegung erfolgt spezifisch für jeden Workflow und jede Objektgruppe auf Basis der ermittelten signifikanten Eigenschaften.
- Die **SIP Spezifikation für automatischen Ingest SLUBArchiv** beschreibt den Aufbau der Ablieferungspakete (englisch: Submission Information Package, SIP) mit denen der Dienstnehmer die zu archivierenden Dokumente für das SLUBArchiv bereitstellt.
- Die **DIP Spezifikation für automatischen Access SLUBArchiv** beschreibt den Aufbau eines Auslieferungspaketes (englisch: Dissemination Information Package, DIP), welches für die automatische Weiterverarbeitung zielgruppengerechter Ausspielungen (Access) von im SLUBArchiv archivierter digitaler Datenobjekte (IE) geeignet ist
- Die **Workflow Spezifikation für automatisierte Interaktionen mit dem SLUBArchiv** beschreibt den Prozess der Übergabe zu archivierender Dokumente in das SLUBArchiv (Ingest / AIP Update), das Fehlerprotokoll und den Zugriff auf die archivierten digitaler Objekte (Access).
- Das Dokument **Spezifikation Rechteausszeichnung SLUBArchiv** beschreibt, wie rechtliche Informationen zu einem Datenobjekt kodiert und abgelegt werden müssen.
- Das Dokument **Webservice SLUBArchiv** beschreibt Funktionen, die Dienstnehmer nutzen können, um Informationen über ihre Daten im SLUBArchiv über einen Webservice abzufragen.
- Vom SLUBArchiv verwendete Begriffe sind im **Glossar SLUBArchiv** definiert.

Warum BagIt?

Das SLUBArchiv hat sehr gründlich die Vor- und Nachteile die Verwendung des [BagIt](https://tools.ietf.org/html/rfc8493) [https://tools.ietf.org/html/rfc8493] Formates der Dissemination Information Packages (DIP) abgewogen.

Für [BagIt](https://tools.ietf.org/html/rfc8493) [https://tools.ietf.org/html/rfc8493] sprechen mehrere Gründe:

- [BagIt](https://tools.ietf.org/html/rfc8493) [https://tools.ietf.org/html/rfc8493] ist mittlerweile in Version 1.0 als [RFC](https://www.rfc-) [https://www.rfc-

editor.org] standardisiert.

- Immer mehr Archivlösungen setzen auf BagIt als Ein- und Auslieferungsformat (z. B. Archivemata und Rosetta).
- Es gibt bereits eine große Anzahl an Programmen und Programmbibliotheken, die Funktionalitäten zur einfachen Verarbeitung von BagIt bereitstellen, hier eine kleine Auswahl:
 - [libcbag](http://andreas-romeike.de/software.html#_libcbag_a_free_and_opensource_c_library_to_handle_bagit_structures) [http://andreas-romeike.de/software.html#_libcbag_a_free_and_opensource_c_library_to_handle_bagit_structures] C++-Bibliothek
 - [pybagit](https://pypi.python.org/pypi/pybagit/) [https://pypi.python.org/pypi/pybagit/] Python-Modul
 - [Archive::BagIt](http://metacpan.org/module/Archive::BagIt) [http://metacpan.org/module/Archive::BagIt] Perl-Modul
 - [Bagger](https://github.com/LibraryOfCongress/bagger) [https://github.com/LibraryOfCongress/bagger] GUI-Programm
 - [bagit-python](https://github.com/LibraryOfCongress/bagit-python) [https://github.com/LibraryOfCongress/bagit-python] CLI-Programm
 - [bagit-java](https://github.com/LibraryOfCongress/bagit-java) [https://github.com/LibraryOfCongress/bagit-java] Java-Bibliothek
- Das Datenformat ist einfach aufgebaut und kann mit Bordmitteln erzeugt und verarbeitet werden.
- Durch die Standardisierung von [BagIt](https://tools.ietf.org/html/rfc8493) [https://tools.ietf.org/html/rfc8493] und weite Verbreitung erfolgt eine Normierung der Werkzeuge für die Erstellung, Prüfung und Verarbeitung von SIPs.
- Die Hürden für die Arbeit mit BagIt sind gering.
- [BagIt](https://tools.ietf.org/html/rfc8493) [https://tools.ietf.org/html/rfc8493] ist modular aufgebaut und erlaubt die einfache Extraktion von Bestandteilen des SIPs.

Aufbau DIP

Generelle Festlegungen

Jedes DIP besteht aus einem Verzeichnis mit einem eindeutigen Namen, welches die [BagIt](https://tools.ietf.org/html/rfc8493) [https://tools.ietf.org/html/rfc8493]-Verzeichnisstruktur abbildet.

Jedes DIP enthält eine Intellektuelle Einheit (IE).



Eine Intellektuelle Einheit ist ein Datenobjekt, welches aus einer oder mehreren Dateien besteht, die gemeinsam verwaltet und charakterisiert werden.

Alle Dateien dieser IE sind innerhalb der [BagIt](https://tools.ietf.org/html/rfc8493) [https://tools.ietf.org/html/rfc8493]-Struktur referenziert.

Das DIP spiegelt den Stand des archivierten Datenobjekts (IE) zum Zeitpunkt der Ausspielung wider!

Fall 1, das Datenobjekt kommt neu in das Archiv

Ist ein SIP auf dem Weg ins Archiv (Erstingest), aber noch **nicht vollständig** im Archivinformationssystem verarbeitet, wird daher **kein** DIP ausgespielt.

Fall 2, das Datenobjekt wird im Archiv aktualisiert:

Ist ein SIP auf dem Weg ins Archiv (AIPupdate), aber noch **nicht vollständig** im

Archivinformationssystem verarbeitet, wird ein DIP mit dem IE der **vorherigen Version** ausgespielt.

Bestandteile des DIP

Grundlage

Basis des DIPs ist eine **BagIt** [<https://tools.ietf.org/html/rfc8493>]-Struktur nach RFC8493. Ergänzend gelten folgende Festlegungen.

Deskriptive Metadaten

Metadaten des Bags werden als Schlüssel-Werte-Paare in der 'bag-info.txt'-Datei kodiert (sh. 2.2.2, RFC 8493).

Administrative Metadaten

Ebenfalls in der Datei 'bag-info.txt' sind, soweit möglich, nachfolgende Informationen als Schlüssel-Werte-Paare kodiert

SLUBArchiv-dipVersion

Der Schlüssel 'SLUBArchiv-dipVersion' enthält 'v2021.1', für die eindeutige Identifizierung des DIP-Formates des SLUBArchivs.

SLUBArchiv-externalWorkflow

In diesem Schlüssel wird der Name des Übergabeworkflows des **ursprünglichen Produzenten** angegeben. Zusammen mit der 'SLUBArchiv-externalId' und den Produzenteninformationen wird diese Information im SLUBArchiv genutzt, um Archivpakete eindeutig zu identifizieren. Anhand dieser beiden Einträge können SIPs zu einem digitalen Dokument (einer IE) zugeordnet werden.

SLUBArchiv-externalId

In diesem Schlüssel wird eine eindeutige ID innerhalb eines Workflows des **ursprünglichen Produzenten** angegeben. Siehe Schlüssel 'SLUBArchiv-externalWorkflow'.



Dieses Feld ist nicht identisch mit dem in der **BagIt** [<https://tools.ietf.org/html/rfc8493>]-Spezifikation angegebenen Schlüssel 'External-Identifizier'.

SLUBArchiv-externalIsilId

Dieser Schlüssel wird von Bibliotheken und andere Gedächtnisorganisationen mit ISIL Nummern genutzt. Er erlaubt die Zuordnung von Vorgängen zu Einrichtungen, hier: zum **ursprünglichen Produzenten**.

Für weitere Informationen nutzen Sie bitte <https://sigel.staatsbibliothek-berlin.de/de/vergabe/isil/>.

Prüfsummen

Die Prüfsummen werden gemäß [BagIt](https://tools.ietf.org/html/rfc8493) [https://tools.ietf.org/html/rfc8493]-Spezifikation (2.4.) hinterlegt. Ebenso wird das Tag 'Payload-Oxum' in der 'bag-info.txt' genutzt.



Die Berechnung der 'Payload-Oxum' ist nur für die Dateien im 'data/'-Verzeichnis anzuwenden.

Dateien der IE und zugehörige Prüfsummen

Alle Dateien, die zum eigentlichen Datenobjekt gehören, sind unterhalb des 'data/'-Verzeichnisses abgespeichert (sh. [BagIt](https://tools.ietf.org/html/rfc8493) [https://tools.ietf.org/html/rfc8493], Abschnitte "Payload" 2.1.2, 2.1.3).

Diese sind durch die Manifest-Dateien prüfsummengesichert.

Ablage Metadaten-Dateien

Gegebenenfalls können Metadaten-Dateien, die Zusatzinfos über das IE enthalten, in dem Verzeichnis 'meta/' abgelegt sein.

Diese sind in den Tag-Manifest-Dateien referenziert.

Unreferenzierte Dateien der IE und Prüfsummen

Alle Dateien, die zum eigentlichen Datenobjekt (IE) gehören, aber deren Dateinamen oder Dateipfad nicht, bzw. nicht mehr abbildbar sind, sind unterhalb des 'unreferenced_data/'-Verzeichnisses wie folgt abgespeichert:

- jede Datei liegt in einem eigenen Unterordner. Der Unterordner wird aus einer [UUID](https://tools.ietf.org/html/rfc4122) [https://tools.ietf.org/html/rfc4122] Version 4 nach [RFC](https://www.rfc-editor.org) [https://www.rfc-editor.org] 4122 gebildet
- ist für eine Datei ein Dateiname nicht mehr bestimmbar oder nicht abbildbar, so wird dieser aus einer [UUID](https://tools.ietf.org/html/rfc4122) [https://tools.ietf.org/html/rfc4122] Version 4 nach [RFC](https://www.rfc-editor.org) [https://www.rfc-editor.org] 4122 gebildet

Alle Dateien des 'unreferenced_data/'-Verzeichnisses sind in den Tag-Manifest-Dateien referenziert.

Liegen keine solchen Dateien vor, so wird kein 'unreferenced_data/'-Verzeichnis erzeugt.

Hintergrund

In äußerst seltenen Fällen kann es vorkommen, dass der originale Dateinamenspfad bzw. der originale Dateiname nicht darstellbar ist (zum Beispiel, wenn Dateinamen in einem bestimmten Dateisystem nicht erzeugt werden können).



Um dem Konsumenten einen klaren Hinweis zu geben, dass der Dateiname bzw. Dateipfad nicht erhalten werden konnte, werden diese Dateien nicht im 'data/' Verzeichnis, sondern im 'unreferenced_data/' Verzeichnis übergeben. Dies erlaubt den weitgehenden Erhalt der ursprünglichen IE in 'data/'.

Weitere Hinweise

Archivierungshistorie

Die DIPs enthalten keine Informationen zur Archivierungshistorie der angefragten IE, sie sind daher **nicht** für AIP-AIP-Transfers geeignet.

Dateipfade

Die Dateikodierung erfolgt auf Basis der [BagIt](https://tools.ietf.org/html/rfc8493) [https://tools.ietf.org/html/rfc8493]-Spezifikation. Es wird empfohlen ein modernes Dateisystem zu verwenden, welches mit folgenden Einstellungen umgehen kann:

- UTF-8 kodierte Dateipfade
- lange Dateinamen
- case sensitive Dateipfade

Das SLUBArchiv ist **bemüht** die Dateipfade möglichst portabel zu halten.

Textdateien

Alle im Rahmen der [BagIt](https://tools.ietf.org/html/rfc8493) [https://tools.ietf.org/html/rfc8493]-Struktur verwendeten Textdateien sind als UTF-8 (ohne BOM) kodiert. Als Zeilenendezeichen wird 'newline' verwendet.

Beispiel

Verzeichnisstruktur Bag

```
examplebag/  
├─ bag-info.txt  
├─ bagit.txt  
├─ data/ ①  
│   ├── 1.txt  
│   ├── 3.dat  
│   └─ subdir/  
│       ├── 2.png  
│       └─ 2.mdx  
├─ unreferenced_data/  
│   └─ 682448d2-d6a8-46f3-927b-d74c65609bca/  
│       └─ 5.unknown ②  
├─ manifest-md5.txt  
├─ manifest-sha512.txt  
├─ meta/  
│   └─ mods.xml  
├─ tagmanifest-md5.txt  
└─ tagmanifest-sha512.txt
```

① Eigentliche Intellektuelle Einheit (IE)

② Ursprünglich zur IE gehörende Datei, deren Pfad nicht mehr zuortbar ist

[1] Die genannten Dokumente sind auf der Webseite des SLUBArchivs unter <https://slubarchiv.slub-dresden.de/technische-standards-fuer-die-ablieferung-von-digitalen-dokumenten/> veröffentlicht und sind dort, technisch bedingt, spezifischer benannt.