



SLUB

Wir führen Wissen.

RFC - Workflow Rosetta - Archivematica Migration

Andreas Romeyke

Version 0.13, 2024-10-15

Inhaltsverzeichnis

Status	1
Begriffe	1
Ist Stand	1
Soll Stand	1
Migrationsvoraussetzungen	1
Workflowszenarien	2
Modulübersicht	2
Erstingest	4
MDUpdate	5
AIPUpdate	5
Access	6
Migration	6
Zustandsdiagramm	7
Versionierung der AIPs in Archivemata	8
AIPs in Rosetta	8
AIPs in Archivemata	9
Algorithmus zur Versionierung	10
Zusammenwirken von Datenbank, Submission Applications und Rosetta	14
Abschätzung Implementierung	14
Erstingest	14
MDUpdate	15
AIP-Update	15
Access	15
AIP-SIP-Transfer	15
Steuerung AIP-SIP-Transfer	15
Quellen	18

Status

Proposed (Entwurf)

Begriffe

siehe Dokument 'SLUBArchiv_Glossar'.

Ergänzend dazu wird im folgenden der Begriff "Transfer Application" oder kurz "TransferApp" verwendet. Dies ist eine speziell konfigurierte Submission Application, die sich ausschliesslich um den eigentlichen AIP-AIP-Transfer kümmert.

Ist Stand

Die SLUB betreibt Rosetta von Exlibris Group als Archivinformationssystem. Die Submission Application ist in der Lage

- SIPs als BagIt einzulesen und diese in Rosetta SIPs zu transformieren
- Rosetta DIPs einzulesen und diese in BagIt basierte DIPs auszugeben
- mithilfe der Rosetta API festzustellen, ob Vorgänge bereits im Archiv sind
- zwischen Erstingest, reinem Metadatenupdate und AIPUpdate zu unterscheiden

Es existieren produktive Workflows für Kitodo und Mediathek, sowie für das LfULG und die UBL. Der Workflow für die Fotothek wird nicht auf Rosetta, sondern auf Archivemata basieren.

Soll Stand

Die SLUB betreibt Archivemata von Artefactual als Archivinformationssystem. Das Archivinformationssystem Rosetta von Exlibris Group ist abgeschaltet. Die AIPs aus diesem System sind per Verzeichnispfad erreichbar.

Es existieren produktive Workflows für Kitodo, Mediathek und Fotothek, sowie für das LfULG und die UBL.

Migrationsvoraussetzungen

Es gelten folgende Voraussetzungen:

- Der Produktivbetrieb von Archivemata ist gesichert.
- Die bestehende Submission Application wird weiterhin verwendet.
- Es existiert eine Datenbank, die alle AIPs des Rosetta-Systems kennt und Abfragen darüber möglich macht. Dies ist via 'Exit-Strategie' ([[SLUB2017exit](#)]) gewährleistet.
- AIPs aus Rosetta müssen, um einen reibungslosen Ablauf zu gewährleisten, ohne Validierung

nach Archivemata überführt werden

- Alle anderen AIPs werden im Ingest validiert.

Workflowszenarien

Modulübersicht

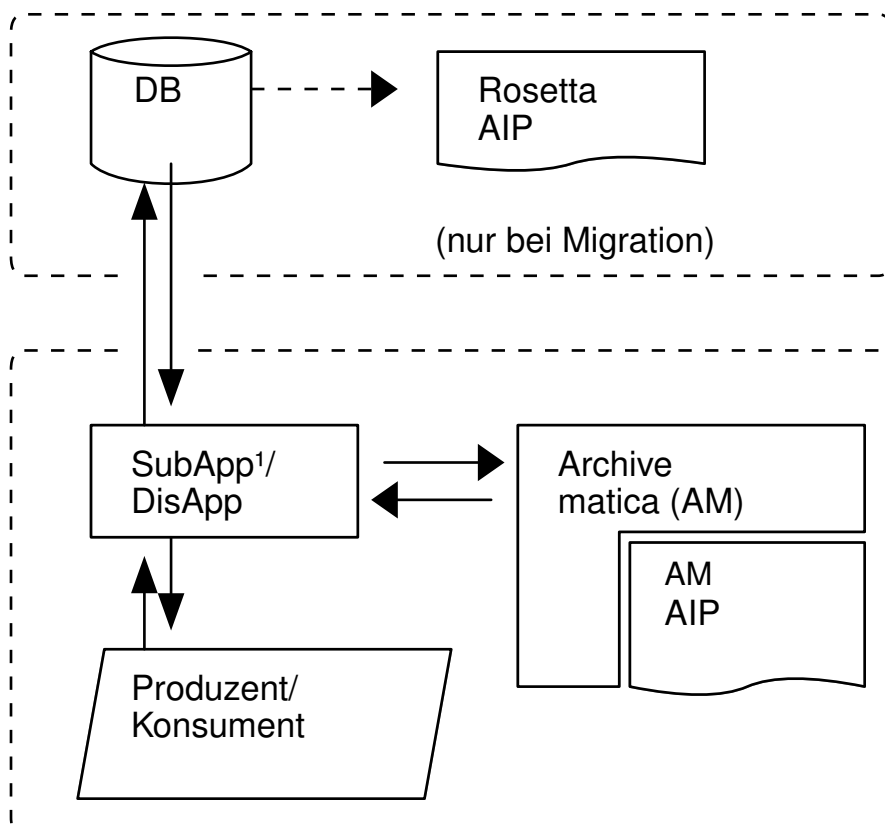


Abbildung 1. Zusammenwirken der Module

¹ SubApp umfasst sowohl die eigentliche Submission Application, die die Produzenten bedient, wie auch die Submission Application, die nur für den AIP-AIP-Transfer zuständig ist (TransferApp).



Für den Workflow Fotothek entfällt der Zugriff auf die DB für Rosetta-AIP.

Im Folgenden die Ablaufpläne innerhalb der SubApp.

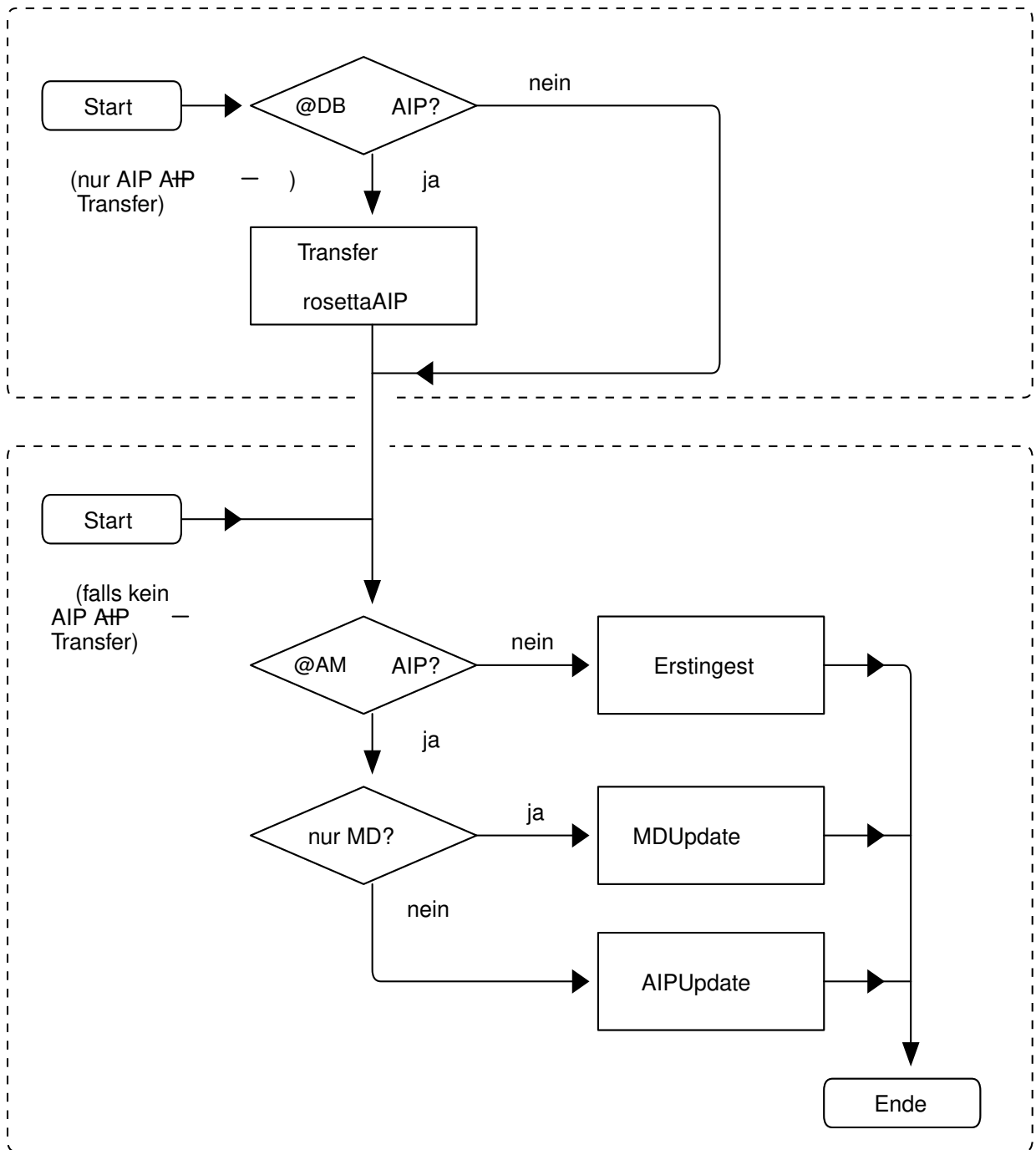


Abbildung 2. Ablaufplan Ingest

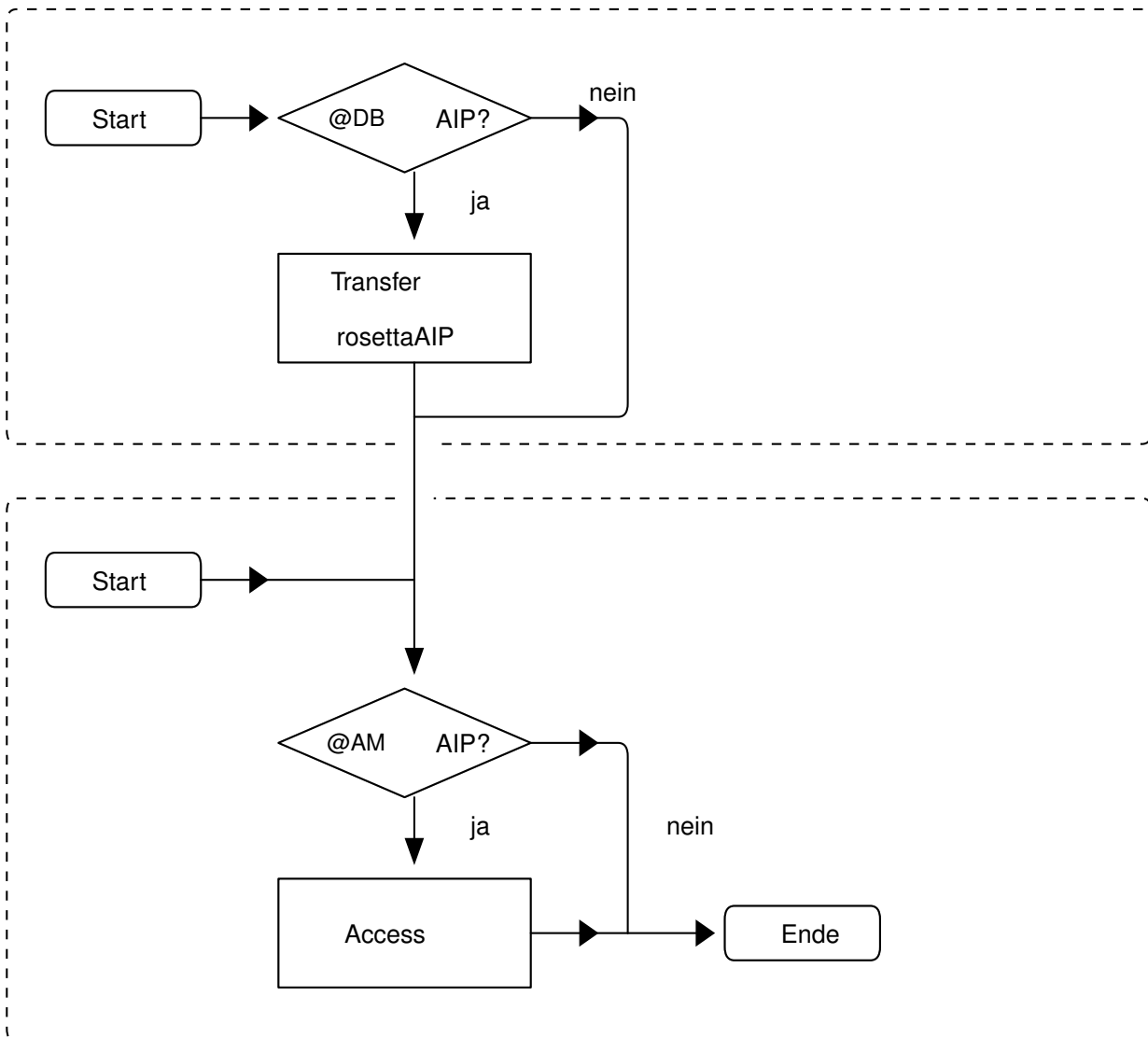


Abbildung 3. Ablaufplan Access

Erstingest

Vorbedingungen im Falle der Rosetta-Archivematica-Migration

- SubApp fragt DB, ob Rosetta AIP zu 'externalID' existiert
 - Wenn ja, dann siehe [Migration](#), danach weiter mit 1.

Vorbedingungen

- SubApp fragt Archivematica, ob AIP zu 'externalID' existiert
 - Wenn ja, dann siehe [MDUpdate](#) oder [AIPUpdate](#)
 - Wenn nein, es existiert kein AIP zu 'externalID'

Folgende Schritte sind notwendig

1. SubApp erstellt Archivemata spezifisches SIP
2. SubApp löst Ingestierung dieses SIPs aus

Nachbedingungen

- Es existiert in Archivemata ein AIP mit 'externalID'

MDUpdate

Vorbedingungen im Falle der Rosetta-Archivemata-Migration

- SubApp fragt DB, ob Rosetta AIP zu 'externalID' existiert
 - Wenn ja, dann siehe [Migration](#)

Vorbedingungen

- SubApp fragt Archivemata, ob AIP zu 'externalID' existiert
 - Wenn nein, dann siehe [Erstingest](#)
- SubApp bestimmt die letzte Version des AIP
- SubApp gleicht Dateien der IE im SIP mit dieser AIP in Archivemata ab
 - Wenn Unterschiede, dann siehe [AIPUpdate](#)
- Es existieren keine Unterschiede in Dateien des IEs der SIP

Folgende Schritte sind notwendig

1. SubApp löst MDUpdate in Archivemata aus

Nachbedingungen

- Es existiert in Archivemata ein AIP mit 'externalID' und geänderten Metadaten.

AIPUpdate

Vorbedingungen im Falle der Rosetta-Archivemata-Migration

- SubApp fragt DB, ob Rosetta AIP zu 'externalID' existiert, wenn ja
- Erstingest alter AIP, siehe [Migration](#)
- anschliessend AIPUpdate via SIP

Vorbedingungen

- SubApp fragt Archivemata, ob AIP zu 'externalID' existiert

- Wenn nein, dann siehe [Erstingest](#)
- SubApp bestimmt die letzte Version des AIP
- SubApp gleicht Dateien der IE im SIP mit dieser AIP in Archivemata ab
 - Wenn keine Unterschiede, dann siehe [MDUpdate](#)

Folgende Schritte sind notwendig

1. SubApp löst AIPUpdate in Archivemata aus

Nachbedingungen

- Es existiert in Archivemata ein weiteres AIP mit 'externalID'.
- Dieses AIP enthält einen Verweis auf seinen Vorgänger

Access

Vorbedingungen im Falle der Rosetta-Archivemata-Migration

- DisApp fragt DB, ob Rosetta AIP existiert
 - Wenn ja, dann siehe [Migration](#)

Vorbedingungen

- Es existiert eine AIP zu 'externalID' in Archivemata

Folgende Schritte sind notwendig

1. DisApp löst Access in Archivemata aus

Nachbedingungen

- Es existiert ein DIP mit 'externalID'

Migration

Basis der Migration ist das Modell "AIP → SIP → AIP" Transfer ([\[Romeyke2016\]](#), Definition 2.4.5, S.27).
Dazu werden alle relevanten Teile der rosetta-AIP als IE betrachtet:

IE mit Dateien der Rosetta AIP, Variante "gepackt"

```
|— V1-FL194620.tif  
|— V1-FL194621.tif  
|— V1-FL194622.tif  
|— V1-FL194623.tif  
|— V1-FL194624.tif  
|— V1-FL194625.xml  
|— V1-FL194626.xml
```

```
├─ V1-FL194627.xml
├─ V1-FL195611.tif
├─ V1-FL195612.tif
├─ V1-FL195613.tif
├─ V1-IE194618.xml ①
├─ V2-IE194618.xml
├─ V3-IE194618.xml
└─ V4-IE194618.xml ②
```

① Erste Version des Rosetta AIPs

② Vierte Version des Rosetta AIPs



In der Fachsitzung vom [2021-07-28](https://intranet.slub-dresden.de/display/LZA/LZA+Fachsitzung+2021-07-28) [https://intranet.slub-dresden.de/display/LZA/LZA+Fachsitzung+2021-07-28] wurde sich darauf verständigt, dass, im Falle des Nachweises aller rosetta-AIP-Versionen, diese alle vollständig als eine IE zusammengepackt nach Archivemata migriert werden.

Für die Variante "Zusammenpacken" spricht die mögliche Deduplizierung von Dateien.

Der Aufbau wird in [AIPs in Archivemata](#) beschrieben.



Sind in Rosetta gelöschte AIPs vorhanden ^[1] so werden nur deren IE-XML-Dateien zusammengepackt.

Der Aufbau wird in [AIPs in Archivemata](#) beschrieben.

Vorbedingungen

- Es existiert eine Rosetta AIP zu 'externalID'
- Es existiert keine AIP zu 'externalID' in Archivemata

Folgende Schritte sind notwendig

1. SubApp bereitet TransferSIP für AIP-AIP-Transfer vor
2. SubApp löst Erstingest in Archivemata aus
3. SubApp löscht AIP in DB

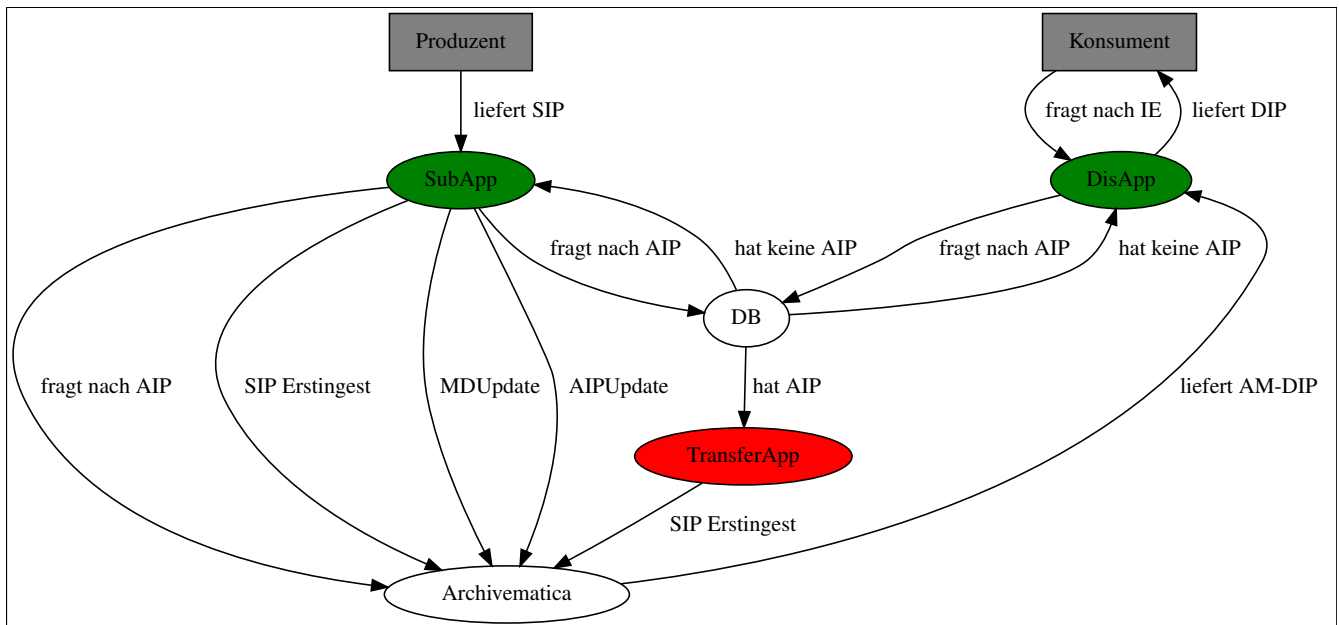
Nachbedingungen

- Es existiert ein AIP mit 'externalID' in Archivemata
- Dieses AIP enthält einen Verweis auf seinen AIP-AIP-Transfer
- Es gibt in DB keinen Eintrag mehr für AIP mit external ID

Zustandsdiagramm



Das Zustandsdiagramm ist ohne Fehlerzustände dargestellt.



TransferApp ist eine Submission Application, die sich nur um den AIP-AIP-Transfer kümmert.

Versionierung der AIPs in Archivemata

Archivemata verfolgt ein anderes Konzept als Rosetta, was die Speicherung von AIPs betrifft.

AIPs in Rosetta

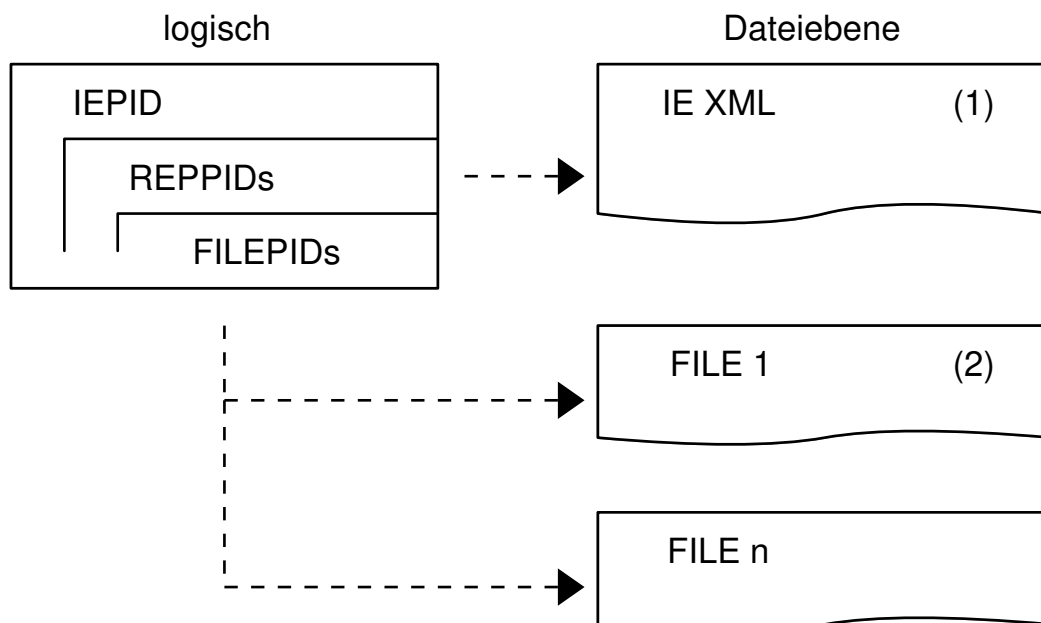


Abbildung 4. Aufbau AIP in Rosetta

(1) Versionsinformation und IE PID nach Schema "Vxx-IEyyyyyy.xml", mit xx - Version, yyyyyy - PID

(2) Versionsinformation und File PID nach Schema "Vxx-FLyyyyyy.zzz", mit xx - Version, yyyyyy - PID, zzz - Dateiendung

Rosetta speichert eine IE-XML, die mehrere Repräsentationen enthalten kann. Jede Repräsentation verweist dabei auf mehrere Dateien. Diese entsprechen dem eigentlichen digitalen Objekt. Das IE-XML bestimmt dabei, was zum AIP gehört.

Die Versionierung erfolgt in Rosetta so, dass jede neue Version des AIP eine neue IE-XML Datei erhält. Nur wenn neue Dateien z. B. bei einem AIPUpdate hinzukommen, dann erhalten diese je eine eigene FILEPID. Bereits bestehende Dateien werden nicht angetastet (siehe ([Exl2016]), DNX Model auf Seite 43ff.).

AIPs in Archivemata

In Archivemata werden AIPs in einer BagIt-Struktur [RFC8493] gespeichert. Dabei enthält das payload-Verzeichnis 'data' die eigentliche IE.

Archivemata kennt dieses Konzept der Versionierung nicht.

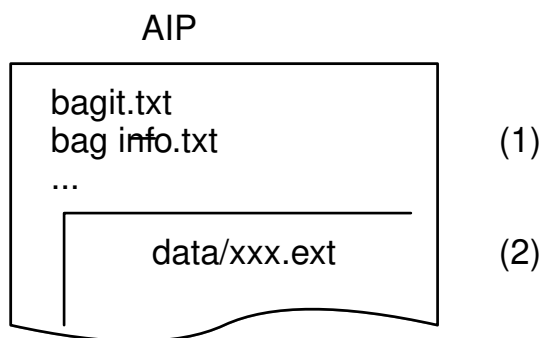


Abbildung 5. Aufbau eines AIP in Archivemata (stark vereinfacht)

(1) 'bag-info.txt' enthält Metadaten über das Bag

(2) der Ordner 'data' enthält die eigentliche IE

Abbildung rosettaAIP in Archivemata AIP

Das Payloadverzeichnis 'data' erhält zusätzlich die Verzeichnisse:

- '.rosetta-aip' - welches die Dateien des Rosetta IE beinhaltet
- '.aip-aip-transfer' - welches Informationen zur Migration enthält
 - 'mapping-rule.rsv' - Datei, enthält die Mapping-Vorschrift "'quellpfad' 'zielpfad'" jeweils bezogen auf BagIt-Hauptverzeichnis RSV ^[2] kodiert
 - 'about.txt' - Datei, die beschreibt, was dieses Verzeichnis bedeutet

Beispiel 'data/.aip-aip-transfer/mapping-rule.rsv'

```
data/.rosetta-aip/V1-FL20642.tif\{ff}data/test01.tif\{ff}\{fd}
```

Beispiel rosettaAIP vor der Migration

```
IE20640/  
├─ V1-FL20642.tif  
└─ V1-IE20640.xml
```

Beispiel AM AIP nach der Migration

```
├─ bag-info.txt  
├─ bagit.txt  
├─ data/  
│   ├── .rosetta-aip/  
│   │   ├── V1-FL20642.tif  
│   │   └─ V1-IE20640.xml  
│   └─ .aip-aip-transfer/  
│       ├── about.txt  
│       ├── mapping-rule.rsv  
│       └─ Rosetta_AIP_model_2024.pdf  
├─ manifest-md5.txt  
├─ manifest-sha512.txt  
├─ meta  
│   └─ rights.xml  
├─ tagmanifest-md5.txt  
└─ tagmanifest-sha512.txt
```



Bei der Ausspielung des DIP wird die Mappingvorschrift in 'data/.aip-aip-transfer/mapping-rule.rsv' angewandt.

Punkt-Verzeichnisse werden im DIP nicht ausgespielt.

Handelt es sich bei dem zu migrierenden AIP um ein gelöschttes AIP, ist ein leeres Mapping in 'data/.aip-aip-transfer/mapping-rule.rsv' zu hinterlegen.

Algorithmus zur Versionierung

Verwendete Schlüssel in 'bag-info.txt'

- 'SLUBArchiv-previous-AIP' - enthält AIP-ID des aktuellen AIS (hier: UUID)
- 'SLUBArchiv-migrated-AIP' - enthält AIP-ID des vorherigen AIS (hier: IE-PID)
- 'SLUBArchiv-origin-AIS' - enthält Name des vorherigen AIS (hier: Rosetta)

Im Fall von gelöschten AIP ist zusätzlich folgender Schlüssel zu verwenden: 'SLUBArchiv-deleted-AIP' mit dem Wert 'true'

Variante verkettete Liste

Als Ansatz wird eine verkettete Liste gewählt.

Es gilt:

- hat ein AIP keinen Vorgänger, so liegt AIP in Version 1 vor (ErstIngest)
- hat AIP einen Vorgänger, so liegt AIP in Version n+1 vor (AIPUpdate)

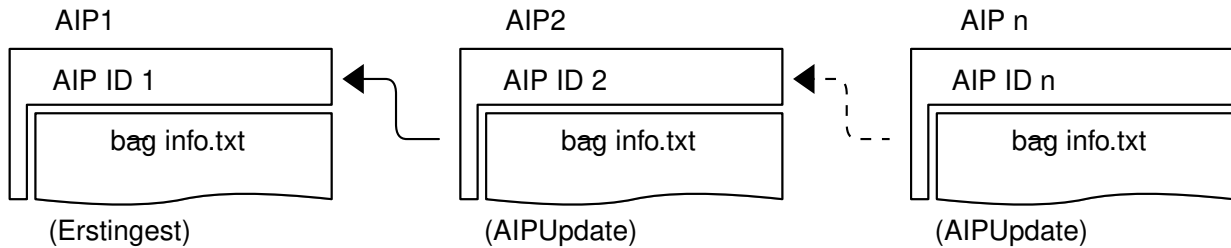


Abbildung 6. Versionsgraph

Die Vorteile liegen darin, dass keine Metadatenupdates für bereits existierende AIPs notwendig werden. Es sind nur wenige Metadaten für die Versionsverwaltung notwendig

Dieses Prinzip kann sowohl für AIPUpdates in Archivemata, als auch zur Abbildung der Migration aus Rosetta benutzt werden.

Dazu werden in 'bag-info.txt' die Schlüssel

- 'SLUBArchiv-previous-AIP' für AIPUpdates
- 'SLUBArchiv-migrated-AIP' und 'SLUBArchiv-origin-AIS' für AIP-AIP-Transfers (Migration)

genutzt.

Da in der Regel nur auf die letzte Version zurückgegriffen wird, reicht obige Angabe aus.

In seltenen Fällen kann es nötig sein, AIPs auf alte Versionen zurückzusetzen. In dem Fall wird eine neue Version erzeugt.

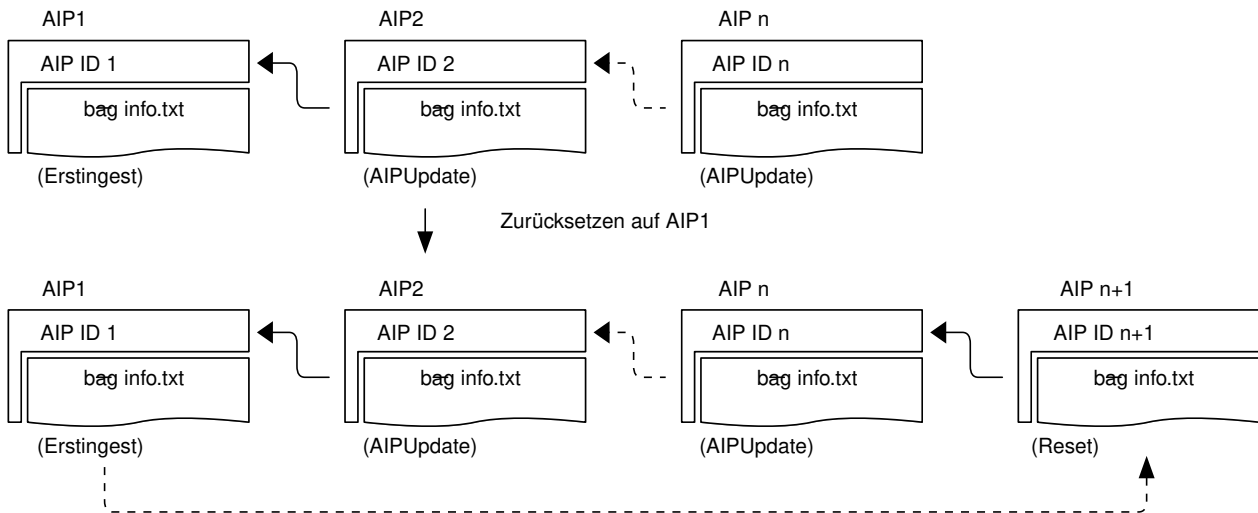


Abbildung 7. Versionsgraph nach Rücksetzen auf AIP1

Beispiel 1. Beispiel AIP Update

1. Frage Archivematica nach AIP mit spezifischer 'externalID'
2. Wenn AIP existiert, verwende dessen AIP-ID als Wert für 'SLUBArchiv-previous-AIP' in 'bag-info.txt'.

Beispiel 2. Beispiel AIP-AIP-Transfer

1. Frage DB nach (Rosetta) AIP mit spezifischer 'externalID'
2. Wenn AIP existiert,
 - verwende dessen IE-PID als Wert für 'SLUBArchiv-migrated-AIP'
 - setze 'SLUBArchiv-origin-AIS' auf 'Rosetta'

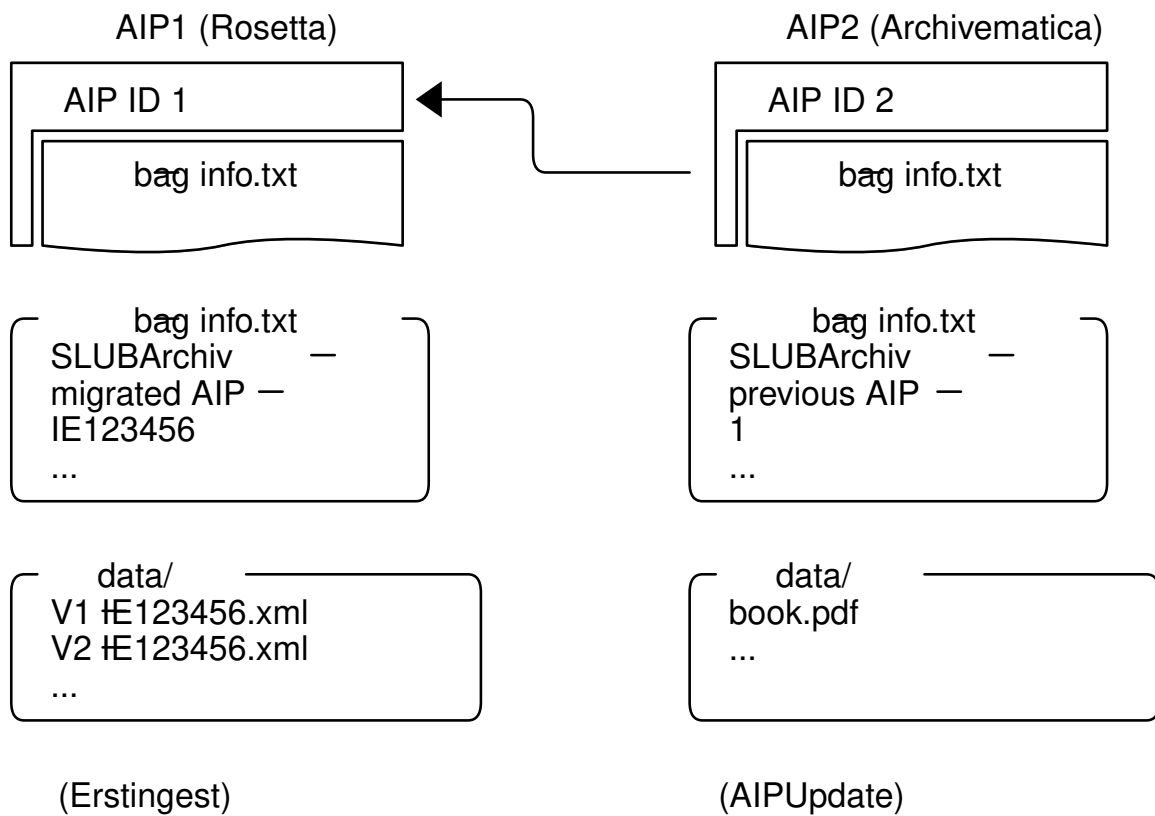


Abbildung 8. Versionsgraph nach getriggertem Migration / AIPUpdate

Beispiel 3. Beispiel Liste aller AIP Versionen

1. Frage Archivemata nach AIPs mit spezifischer 'externalID'
2. Wenn AIP nicht existiert, Abbruch
3. Bestimme die AIP ohne 'previous-AIP' in 'bag-info.txt' → '\$prev'
4. Wiederhole rekursiv bis leere Liste:
 - Bestimme AIP mit 'SLUBArchiv-previous-AIP' == '\$prev'

Beispiel Liste aller AIP Versionen (Perl)

```

sub get_all_aip_versions ($external_id) {
  my @matches = get_aip_ids_matching( $externalid );
  return if 1 == @matches;
  return sort { $a->{'previous-AIP'} eq $b->{'AIPid'} } @matches;
}

```

Beispiel 4. Beispiel Setze auf n-te AIP Version zurück

1. Frage Archivemata nach AIP, merke dessen AIP-ID
2. Aus Liste der AIPs, bestimme AIP auf das zurückgesetzt werden soll
3. Hole AIP

4. Verwende die gemerkte ID als Wert für 'SLUBArchiv-previous-AIP' in 'bag-info.txt'

Zusammenwirken von Datenbank, Submission Applications und Rosetta

Die Migration erfolgt in zwei Schritten:

1. Stetige Migration von AIPs via TransferApp pro Submission Application Workflow
2. Migration der restlichen AIPs durch neu zu erstellende Submission Application Workflows

Rosetta wird abgeschaltet und das zugehörige Speichersystem read-only gesetzt.

Via Script zur Exitstrategie wird aus Rosetta eine SQLite-Datenbank erzeugt, die für jeden Submission Application Workflow gespiegelt wird.

Dieses Vorgehen bietet mehrere Vorteile:

1. Es kann auf eine regelmäßige Sicherung der Datenbanken verzichtet werden. Die Datenbanken dienen nur dazu die Abfrage, ob ein AIP aus Rosetta migriert werden muss, schnell zu beantworten. Im Fehlerfall werden die Datenbanken neu aufgesetzt.
2. Auf eine Synchronisierung der Zugriffe auf die Datenbanken kann verzichtet werden, da nicht mehrere SubApps auf die Datenbank zugreifen (müssen). Dadurch sinkt der Implementierungs- und Administrationsaufwand, insbesondere beim Hoch-/Runterfahren von Diensten.

Um den administrativen Aufwand gering zu halten, erhält die TransferApplication einen eigenen Eventcallback, der regelmäßige Migrationen von AIPs anstößt.

Abschätzung Implementierung

Erstingest

Abfrage DB

Die Abfrage der DB ist trivial. Die DB steht als SQLite-DB zur Verfügung. Es wird nur auf Existenz der AIP durch Abfrage von LZaid gefragt.

Für den Workflow Fotothek entfällt dieser Schritt. Er wird implementierungstechnisch nicht extra behandelt werden.

Aufwand geschätzt: 1 PT Entwicklung, 1PT Test

Abfrage Archivematica, ob Erstingest

Dies könnte über eine Such-Abfrage erfolgen [\[amsearch\]](#).

Aufwand geschätzt, 1 PT Entwicklung, 1PT Test

Ingestierung in Archivematica

Es muss ein AM-spezifisches BagIt gebaut werden. Dieses Bag wird durch Aufruf Webservice Archivematica übermittelt. In der Submission Application müssen dazu neue EventCallbacks angelegt werden

Aufwand geschätzt, 5 PT Entwicklung, 8PT Test

MDUpdate

Siehe ([\[SLUB2021amcontract\]](#)) und ([\[AM2021eventimport\]](#))

In der Submission Application müssen dazu neue EventCallbacks angelegt werden.

Zurzeit keine Schätzung möglich.

AIP-Update

Es muss die Archivematica-ID des AIP ermittelt werden, dann wird ein neues Bag für Archivematica erzeugt und die notwendigen Ergänzungen in der 'bag-info.txt' vorgenommen. Es müssen in der Submission Application neue EventCallbacks angelegt werden.

Aufwand geschätzt, 5 PT Entwicklung, 8PT Test

Access

Es muss die Archivematica-ID des AIP ermittelt werden, dann wird über WebAPI der Access in AM ausgelöst (siehe [citenp\[AM2021download\]](#)) und das DIP gebaut.

In der Submission Application müssen dazu neue EventCallbacks angelegt werden.

Aufwand geschätzt, 3 PT Entwicklung, 4PT Test

AIP-SIP-Transfer

Es muss das AIP vom Dateisystem von Rosetta geholt und als Bag für Archivematica gebaut werden (entspricht AIP-SIP-Transfer).

Es werden extra Einträge in 'bag-info.txt' erzeugt (siehe [Algorithmus zur Versionierung](#)). Danach wird es wie Erstingest behandelt.

Im Erfolgsfall muss der Eintrag in DB gelöscht werden.

In der TransferApplication müssen dazu neue EventCallbacks angelegt werden.

Aufwand geschätzt, 5 PT Entwicklung, 8PT Test

Steuerung AIP-SIP-Transfer

Folgende Komponenten können den AIP-SIP-Transfer triggern:

- Submission Application im Falle eines MDUpdate
- Submission Application im Falle eines AIPUpdate
- Dissemination Application im Falle einer DIP-Anfrage und nicht-Vorhandensein in Archivematica
- Die TransferApp übernimmt dabei parallel die dauerhaft laufende Migration ^[3].

Um die Steuerung zu vereinfachen, wird die Datenbank (aus Exitstrategie) mitgenutzt. Dazu erhält sie eine Tabelle 'transferaip' mit der Zuordnung:

'iepid' → 'lzaid' → 'transferstate'.

'transferstate' ist einer der folgenden Werte:

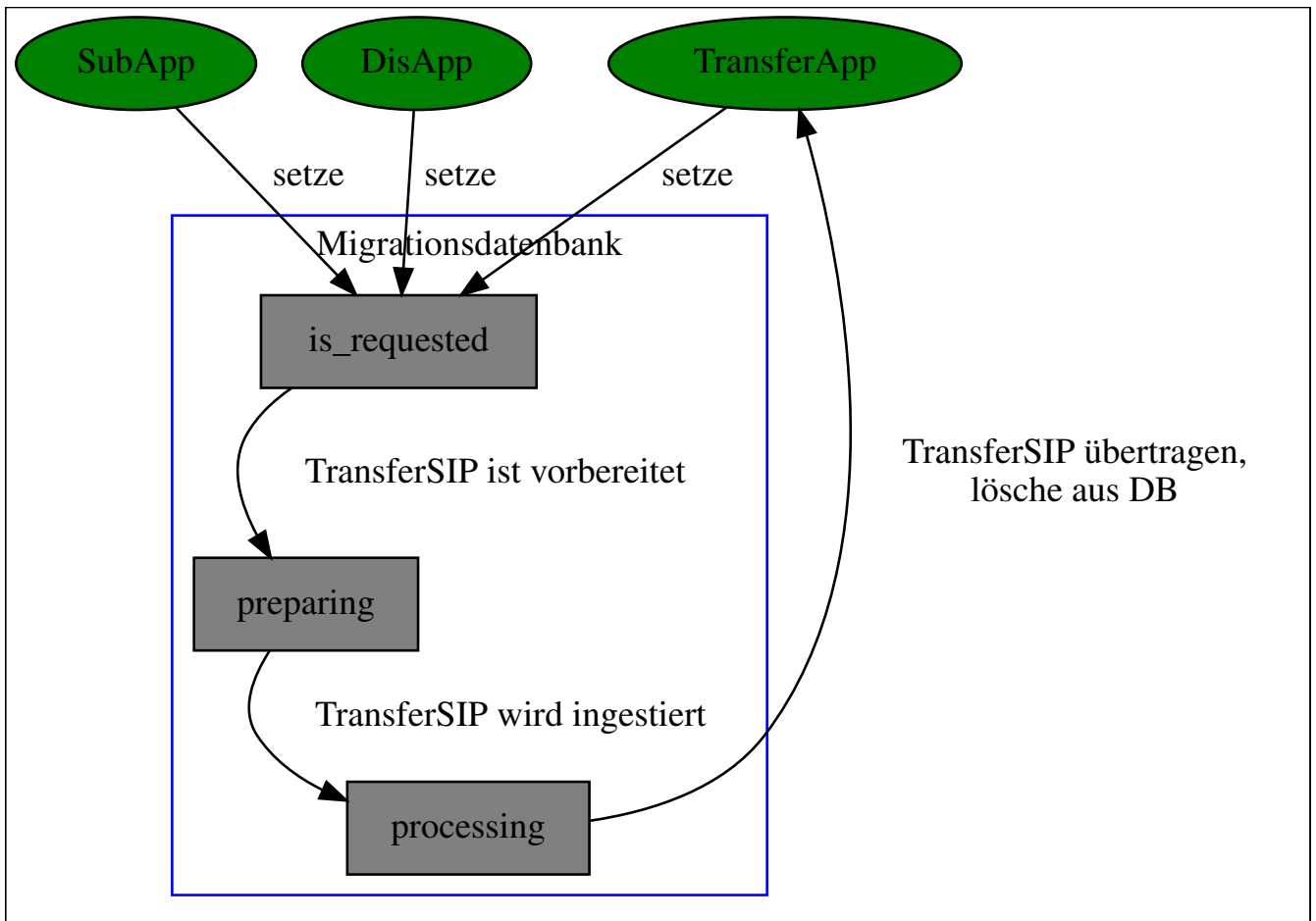
- is_requested → das AIP soll zeitnah transferiert werden
- preparing → das AIP wird als SIP vorbereitet
- processing → das AIP wird als SIP eingested

Tabelle 1. Beispiel Datenbanktabelle transferaip

iepid	lzaid	transferstate
IE129992	SLUB:LZA:Kitodo:kitodo:732732 21	is_requested

Solange ein zugehöriges AIP sich im 'transferstate' (Tabelle 'transferaip') befindet, ist die Verarbeitung des darauf basierenden SIPs oder DIPs in SubApp oder DisApp blockiert.

Wenn der Transfer erfolgreich war, wird das AIP komplett aus DB entfernt und die eigentliche Verarbeitung des darauf basierenden SIPs oder DIPs in SubApp oder DisApp wird fortgeführt.



Die Tabelle sollte schrittweise gefüllt werden, um ein Verhungern von Anfragen zu vermeiden.

Um im Falle eines Neuaufbaus unnötiges Processing zu vermeiden, muss die Transfer Application für die Transition des 'transferstate' von 'is_requested' → 'preparing' immer erst Archivemata anfragen, ob die 'externalID' noch nicht dort existiert. Andernfalls wird das AIP ebenfalls aus der DB entfernt, da eine Migration ja schon stattgefunden hat.



Die 'externalID' steht nicht für jedes AIP in Rosetta verfügbar (wenn das AIP manuell eingesteuert wurde). In dem Fall **muss** durch die SubApp eine 'externalID' erzeugt werden.

Die 'externalID' kann daher **nicht** als Primärschlüssel in der Datenbank verwendet werden!

Aufwand geschätzt, 3 PT Entwicklung, 3PT Test

Quellen

- [AM2021eventimport] Archivemata Dokumentation (Hrsg.): **Importing event metadata with premis.xml**. Artefactual Systems Inc., 2021, <https://www.archivemata.org/en/docs/archivemata-1.12/user-manual/transfer/import-metadata/#premis-xml> besucht: 2021-06-14
- [Exl2016] Ex Libris Documentation Department (Hrsg.): **Rosetta AIP Data Model**. ExLibris Group, 2016, S.43 ff., https://knowledge.exlibrisgroup.com/@api/deki/files/39700/Rosetta_AIP_Data_Model.pdf besucht: 2016-03-16
- [RFC8493] RFC8493, 2020: **RFC8493 - The BagIt File Packaging Format**. Internet Engineering Task Force (IETF), 2020, <https://datatracker.ietf.org/doc/rfc8493/>
- [Romeyke2016] Romeyke, Andreas: **Übertragbarkeit von Archivinformationspaketen zwischen Langzeitarchivsystemen als Teil der Exit-Strategie**. Hochschule für Technik, Wirtschaft und Kultur Leipzig, 2016, http://andreas-romeyke.de/bkm/romeyke_masterarbeit2016.pdf
- [SLUB2017exit] **SlubArchiv.digital: Exit-Strategie Rosetta**. Sächsische Landesbibliothek – Staats- und Universitätsbibliothek Dresden (2017), Technischer Bericht, https://slubarchiv.slub-dresden.de/fileadmin/groups/slubsite/slubarchiv/SLUBArchiv_Exit_Strategie_v1.3.4.pdf
- [SLUB2021amcontract] SLUB Intranet (Hrsg.): **Ausschreibung „Softwareentwicklung für das Archivinformationssystem Artefactual Archivemata – Erweiterung der Funktionalität für Metadaten-Import, Metadaten-Aktualisierung und Metadatenuche“**. SLUBArchiv.digital, 2021

[1] eventDescription='IE has been deleted' in IE-XML

[2] Rows of String Values, Spezifikation unter <https://github.com/Stenway/RSV-Specification>

[3] Falls eine Steuerung der Abarbeitung/Neupriorisierung der zu migrierenden AIPs notwendig ist, kann dies über eine Anpassung der Datenbanktabelle 'transferaip' erfolgen.