SLUB
Wir führen Wissen.

RFC - Workflow Rosetta - Archivematica Migration

Andreas Romeyke

Version 0.11, 2024-03-20

Inhaltsverzeichnis

Status

Proposed (Entwurf)

Begriffe

siehe Dokument *SLUBArchiv_Glossar*.

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 Archivematica basieren.

Soll Stand

Die SLUB betreibt Archivematica 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 Vorrausetzungen:

- Der Produktivbetrieb von Archivematica 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*[es] gewährleistet.

Workflowszenarien

Modulübersicht

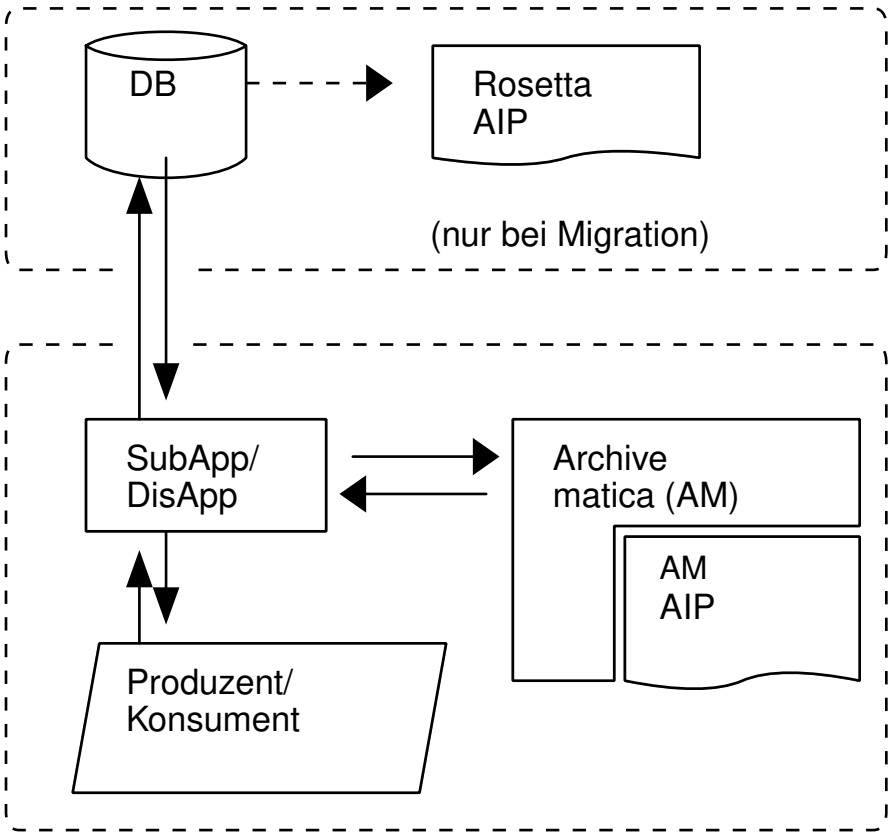


Abbildung 1. Zusammenwirken der Module

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

Im folgenden die Ablaufpläne innerhalb der SubApp.

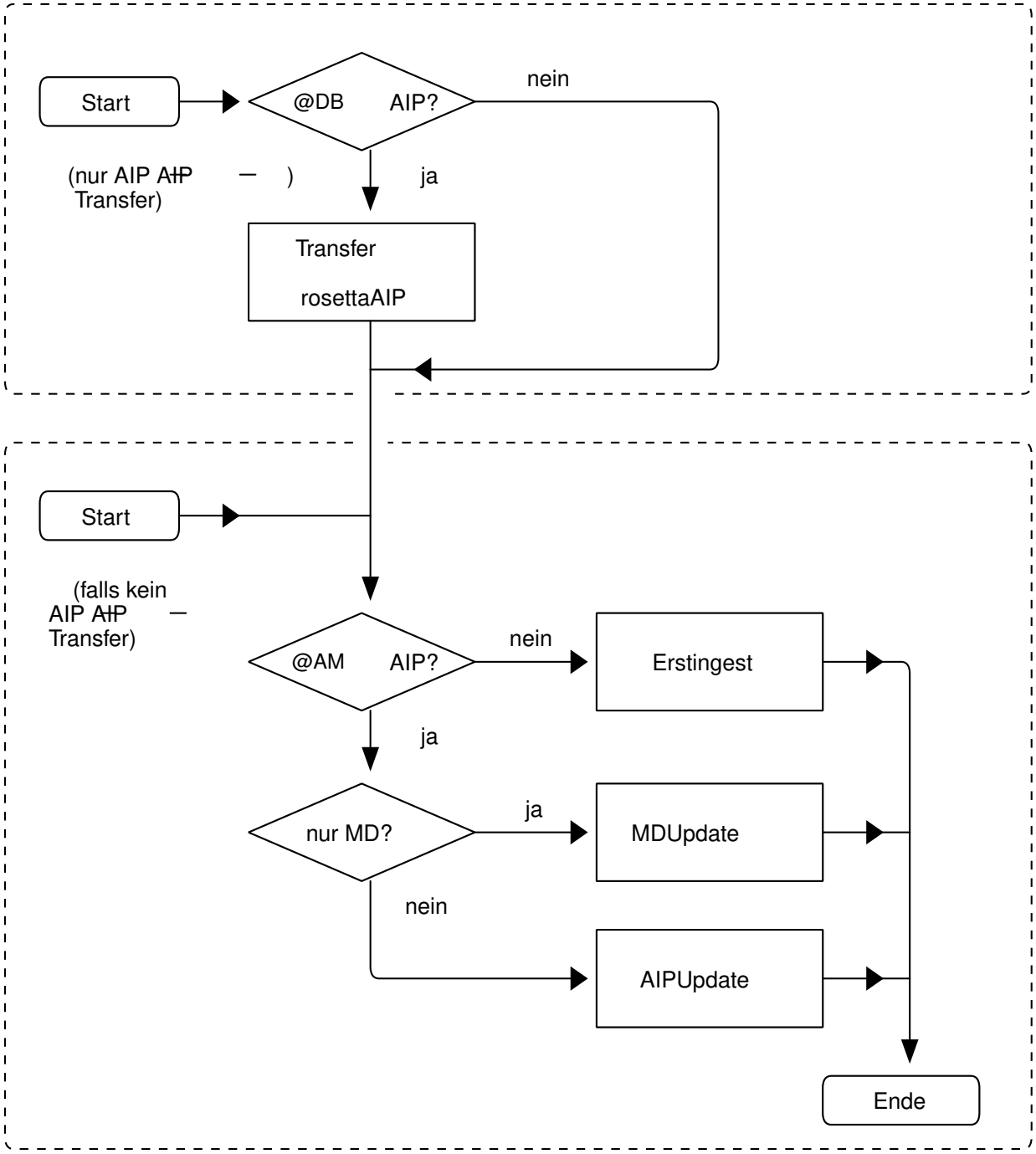


Abbildung 2. Ablaufplan Ingest

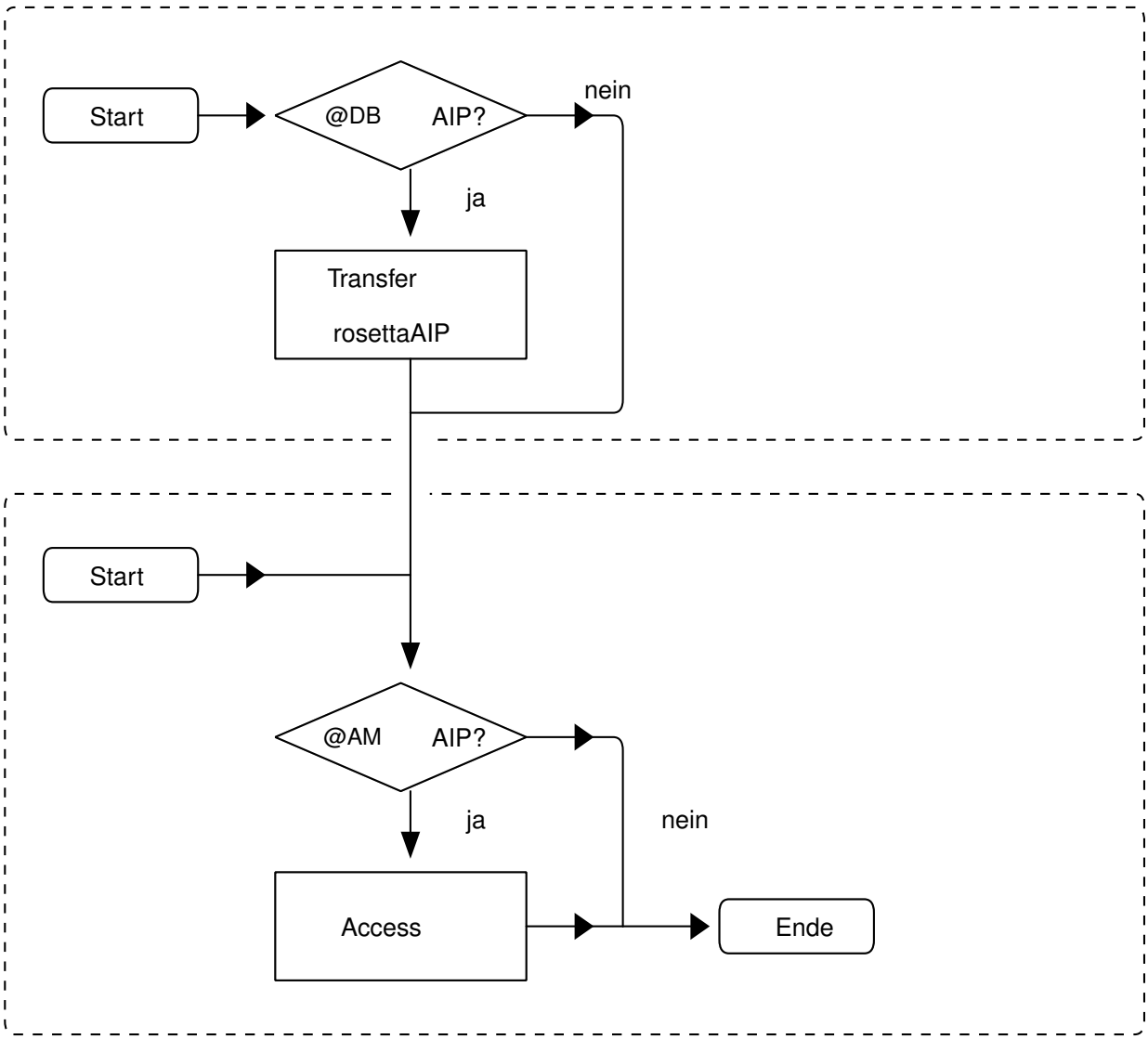


Abbildung 3. Ablaufplan Access

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 Archivematica spezifisches SIP
2. SubApp löst Ingestierung dieses SIPs aus

Nachbedingungen

- Es existiert in Archivematica ein AIP mit *externalID*

MDUpdate

Vorbedingungen im Falle der Rosetta-Archivematica-Migration

- SubApp fragt DB, ob Rosetta AIP zu *externalID* existiert
  - Wenn ja, dann siehe [Migration]

Vorbedingungen

- SubApp fragt Archivematica, 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 Archivematica 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 Archivematica aus

Nachbedingungen

- Es existiert in Archivematica ein AIP mit *externalID* und geänderten Metadaten.

AIPUpdate

Vorbedingungen im Falle der Rosetta-Archivematica-Migration

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

Vorbedingungen

- SubApp fragt Archivematica, 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 Archivematica ab
    - Wenn keine Unterschiede, dann siehe [MDUpdate]

Folgende Schritte sind notwendig

1. SubApp löst AIPUpdate in Archivematica aus

Nachbedingungen

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

Access

Vorbedingungen im Falle der Rosetta-Archivematica-Migration

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

Vorbedingungen

- Es existiert eine AIP zu *externalID* in Archivematica

Folgende Schritte sind notwendig

1. DisApp löst Access in Archivematica aus

Nachbedingungen

- Es existiert ein DIP mit *externalID*

Migration

Basis der Migration ist das Modell "AIP → SIP → AIP" Transfer[ar]. 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] wurde sich darauf verständigt, dass, im Falle des Nachweises aller rosetta-AIP-Versionen, diese alle vollständig als eine IE zusammengepackt nach Archivematica migriert werden.

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

Der Aufbau wird in [aips_in_archivematica] beschrieben.

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

Der Aufbau wird in [aips_in_archivematica] beschrieben.

Vorbedingungen

- Es existiert eine Rosetta AIP zu *externalID*
- Es existiert keine AIP zu *externalID* in Archivematica

Folgende Schritte sind notwendig

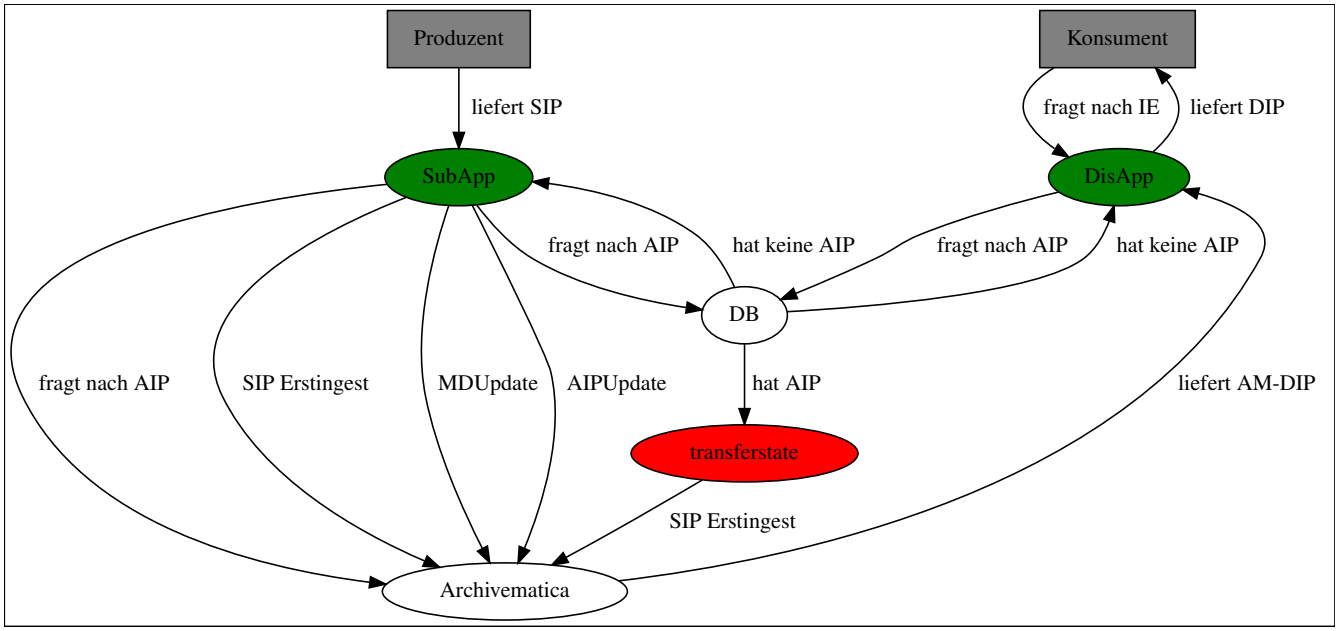
1. SubApp bereitet TransferSIP für AIP-AIP-Transfer vor
2. SubApp löst Erstingest in Archivematica aus
3. SubApp löscht AIP in DB

Nachbedingungen

- Es existiert ein AIP mit *externalID* in Archivematica
- 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.



Der Zustand "transferstate" ist ebenfalls Teil der SubApp.

Versionierung der AIPs in Archivematica

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

AIPs in Rosetta

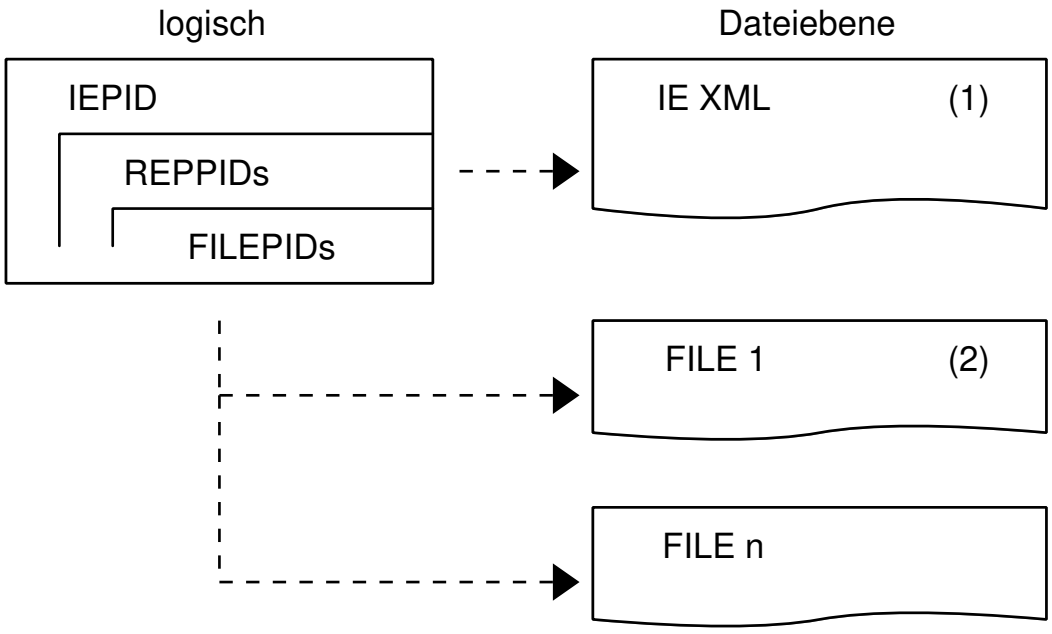


Abbildung 4. Aufbau AIP in Rosetta

(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 zB. bei einem AIPUpdate hinzukommen, dann erhalten diese je eine eigene FILEPID. Bereits bestehende Dateien werden nicht angetastet (siehe [exlibris]).

AIPs in Archivematica

In Archivematica werden AIPs in einer BagIt-Struktur ([bagit]) gespeichert. Dabei enthält das payload-Verzeichnis *data* die eigentliche IE.

Archivematica kennt dieses Konzept der Versionierung nicht.

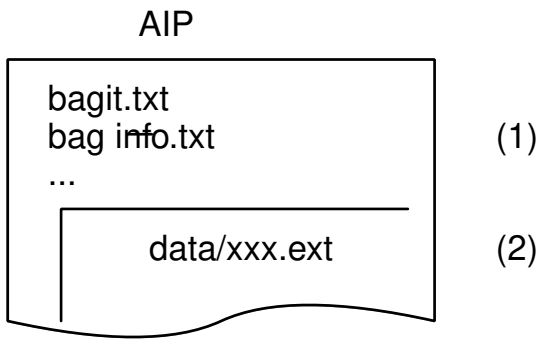


Abbildung 5. Aufbau eines AIP in Archivematica (stark vereinfacht)

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

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

Abbildung rosettaAIP in Archivematica 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\x{ff}data/test01.tif\x{ff}\x{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 gelöschtes 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)

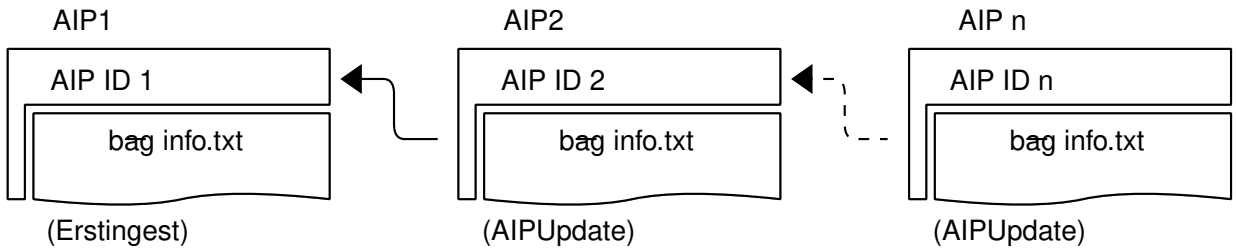


Abbildung 6. Versionsgraph

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 Archivematica, wie 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.

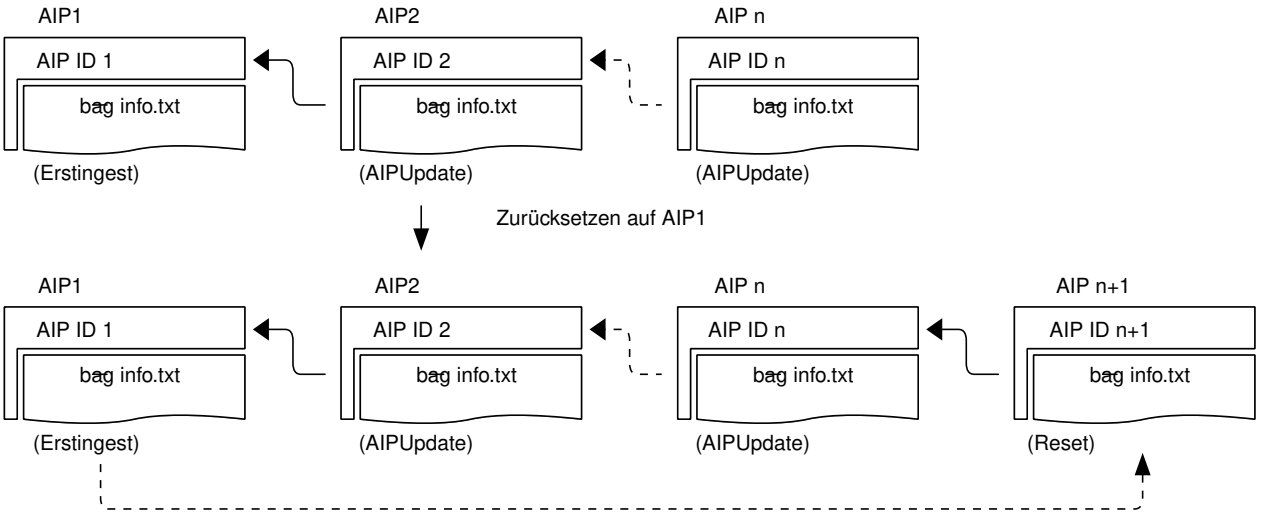


Abbildung 7. Versionsgraph nach Rücksetzen auf AIP1

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*

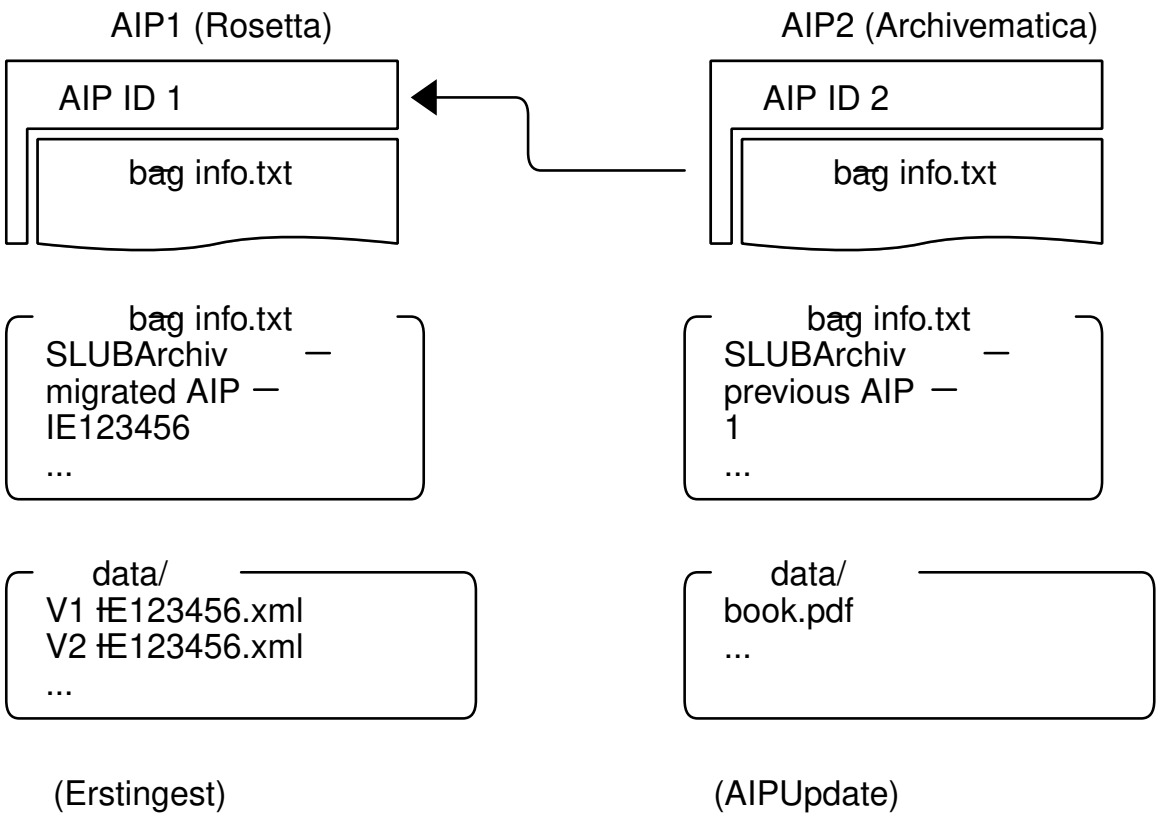


Abbildung 8. Versionsgraph nach getriggerter Migration / AIPUpdate

Beispiel 3. Beispiel Liste aller AIP Versionen

1. Frage Archivematica 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 Archivematica nach AIP, merke desses 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 Application und Rosetta

Die Migration erfolgt in zwei Schritten:

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

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 Submission Application einen eigenen Eventcallback, der regelmäßige Migrationen von AIPs anstößt. Auf ein separates Script bzw. eine Migrationsapplication wird verzichtet, da deren Vorteile den Implementierungsaufwand nicht rechtfertigen.

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 [slub1] und [ameventimport]

In der Submission Application müssen dazu neue EventCallbacks angelegt werden.

Zur Zeit 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 [amdownload]) 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 muss die Migration als PREMIS-Event erzeugt werden. Danach wird es wie Erstingest behandelt.

Im Erfolgsfall muss der Eintrag in DB gelöscht werden.

In der Submission Application 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
- Transferscript für parallele, dauerhaft laufende Migration

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:

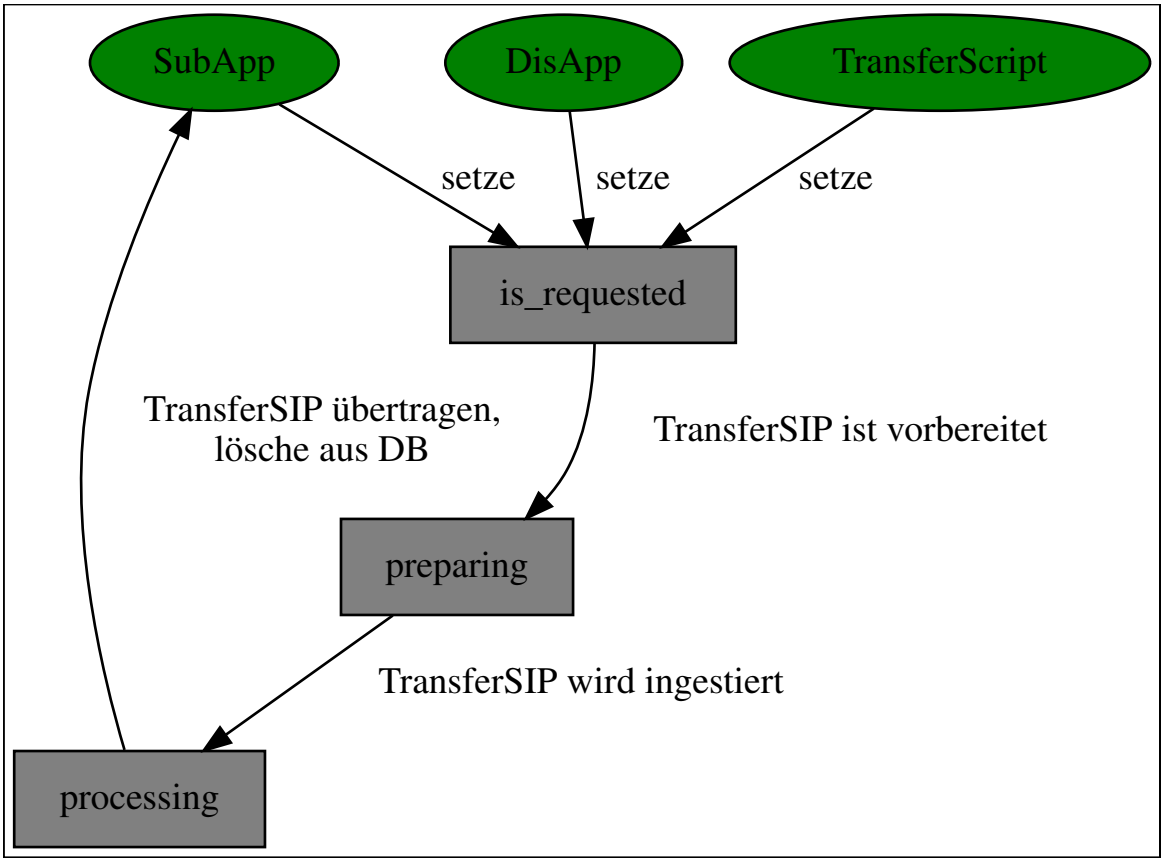
- is_requested → das AIP soll zeitnah transferiert werden
- preparing → das AIP wird als SIP vorbereitet
- processing → das AIP wird als SIP geingested

Tabelle 1. Beispiel Datenbanktabelle transferaip

iepid	**lzaid**	**transferstate**
IE129992	SLUB:LZA:Kitodo:kitodo:73273221	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 Submission Application für die Transistion des *transferstate* von *is_requested* → *preparing* immer erst Archivematica 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 geingested 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

Offene Fragen

- ~~Gibt es in Archivematica eine Möglichkeit alle AIPs zu bestimmen, die in *bag-info.txt* die gleiche *externalID* benutzen **und** diese nach Datum zu sortieren?~~ Antwort: ja, über Anfrage Elasticsearch würde man für die Frage nach AIP den neuesten Eintrag verwenden (und auf den nicht referenziert wird)
- ~~Gibt es Möglichkeiten in Archivematica beim Ingest bestimmte PREMIS Events mitzugeben? Wenn ja, könnte man die Versionsinformationen zusätzlich über Events in das entstehende AIP eintragen.~~ **Ja**, siehe → Siehe [ameventimport]
- An welchen Stellen ist PT Schätzung unrealistisch?

Quellen

- [ar27] Übertragbarkeit von Archivinformationspaketen zwischen Langzeitarchivsystemen als Teil der Exit-Strategie, Andreas Romeyke, 2016-10-17, http://andreas-romeyke.de/bkm/romeyke_masterarbeit2016.pdf, Definition 2.4.5, S.27
- [es] SlubArchiv.digital: Exit-Strategie Rosetta. , Techn. Ber., Sächsische Landesbibliothek – Staats- und Universitätsbibliothek Dresden (SLUB), v1.3.4, 2017-10-18., url: https://slubarchiv.slub-dresden.de/fileadmin/groups/slubsite/slubarchiv/SLUBArchiv_Exit_Strategie_v1.3.4.pdf
- [exlibris] Ex Libris Documentation Department (Hrsg.): Rosetta AIP Data Model, Englisch, Version 5.0, DNX Model auf Seite 43ff., ExLibris Group, 16. März 2016,url: https://knowledge.exlibrisgroup.com/@api/deki/files/39700/Rosetta_AIP_Data_Model.pdf
- AIP re-ingest, Englisch, in: Artefactual Systems Inc. (Hrsg.), 28. Apr. 2016, url: https://wiki.archivematica.org/AIP_re- ingest, ArchivematicaWiki.
- Transfer, Englisch, Version 1.5, in: Artefactual Systems Inc. (Hrsg.) 2016, url:https://www.archivematica.org/en/docs/archivematica-1.5/user-manual/transfer/transfer/[], Archivematica Dokumentation.
- [bagit] Kunze, J. u. a.: The BagIt File Packaging Format, Englisch, Techn. Ber., RFC 8493, Internet Engineering Task Force (IETF), 2020-01-21, url: https://datatracker.ietf.org/doc/rfc8493/
- [amsearch] Bestandsaufnahme existierender Suchfunktionen, Intranet der SLUB, 2021-06-14, https://intranet.slub-dresden.de/pages/viewpage.action?spaceKey=LZA&title=Bestandsaufnahme+existierender+Suchfunktionen
- [amreport] AM Test - Reporting, Intranet der SLUB, 2021-06-14, https://intranet.slub-dresden.de/display/LZA/AM+Test+-+Reporting
- [ameventimport] Importing event metadata with premis.xml, Artefactual Systems Inc. (Hrsg.), 2021-06-14, url: https://www.archivematica.org/en/docs/archivematica-1.12/user-manual/transfer/import-metadata/#premis-xml, ArchivematicaWiki.
- [amdownload] Download Package, , Artefactual Systems Inc. (Hrsg.), 2021-06-14, url: https://wiki.archivematica.org/Storage_Service_API#Download_package, ArchivematicaWiki.
- [slub1] Ausschreibung „Softwareentwicklung für das Archivinformationssystem Artefactual Archivematica – Erweiterung der Funktionalität für Metadaten-Import, Metadaten-Aktualisierung und Metadatensuche“, SLUB intern, 2021-03-11

[1] eventDescription=*IE has been deleted* in IE-XML
[2] Rows of String Values, Spezifikation unter https://github.com/Stenway/RSV-Specification