DIE ÖSTERREICHISCHE BIBLIOTHEKENVERBUND
UND SERVICE GMBH

obv sg

FROM GROUND TO CLOUD:

MIGRATION OF CONSORTIA CATALOG ENRICHMENT WORK FLOW EDOC FROM ALEPH TO ALMA

VICTOR BABITCHEV

DACHELA 2017
BERN, 22.-23. JUNI 2017

AGENDA

- ❑ eDOC current status
- ❑ eDOC workflow
- ❑ Getting closer to Alma
- ❑ Our (API) development approach
- ❑ Results
- ❑ Summary

eDOC current status – 1

In course of ALMA implementation OBVSG defined central services to be migrated from Aleph to Alma Network Zone

– eDOC is one of them, the production start is Q1 2018

We began to develop eDOC in 2004 – now it is a set of workflows for enrichment of Aleph central catalog with electronic contents

eDOC integrates central services and collects their electronic contents; it also extracts and stores texts from objects and passes them to Primo for full text indexing (mostly PDFs)

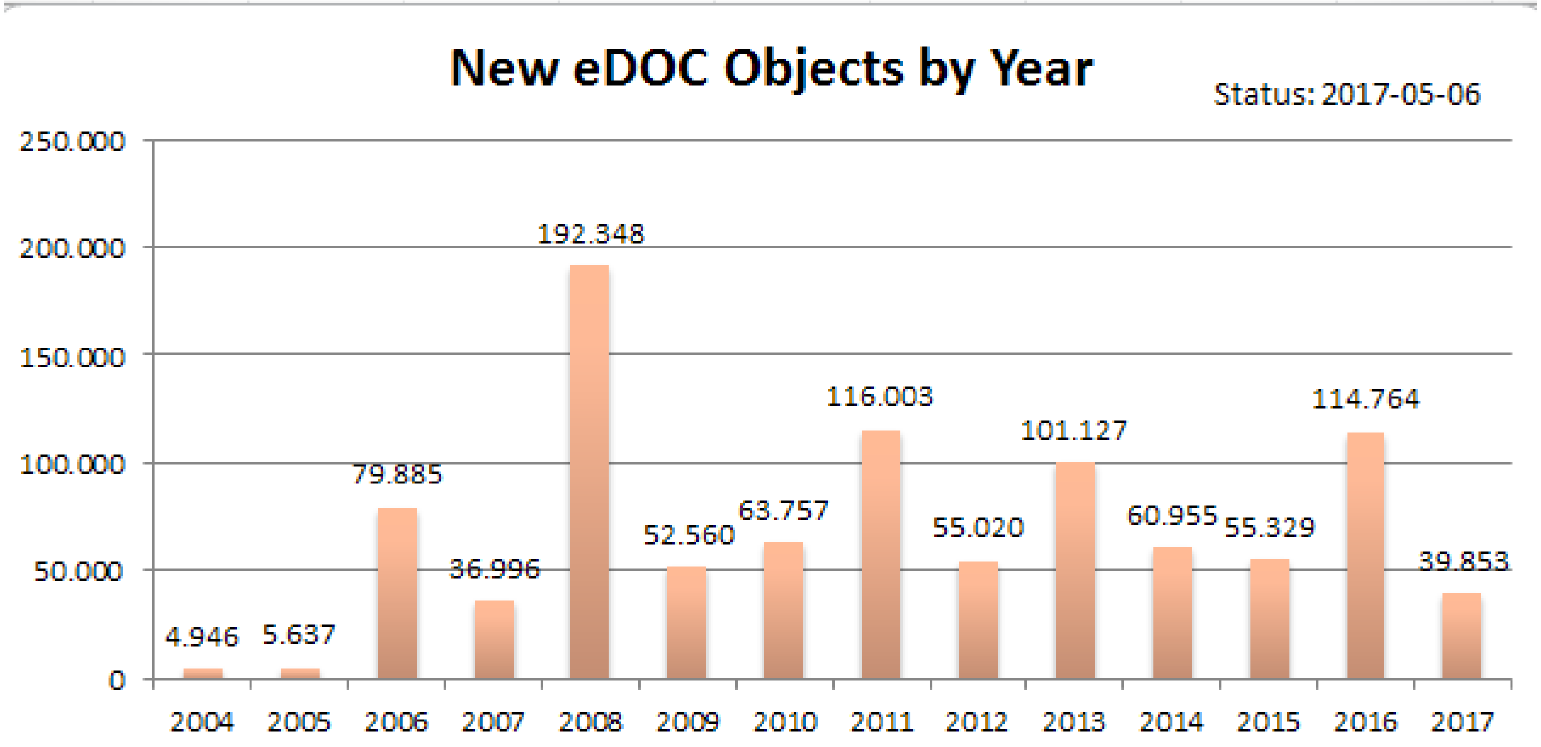
eDOC strong sides are an efficient and librarian-friendly work flow and reliable fully automated operations

… and we want to have in Alma too!

eDOC current status – 2

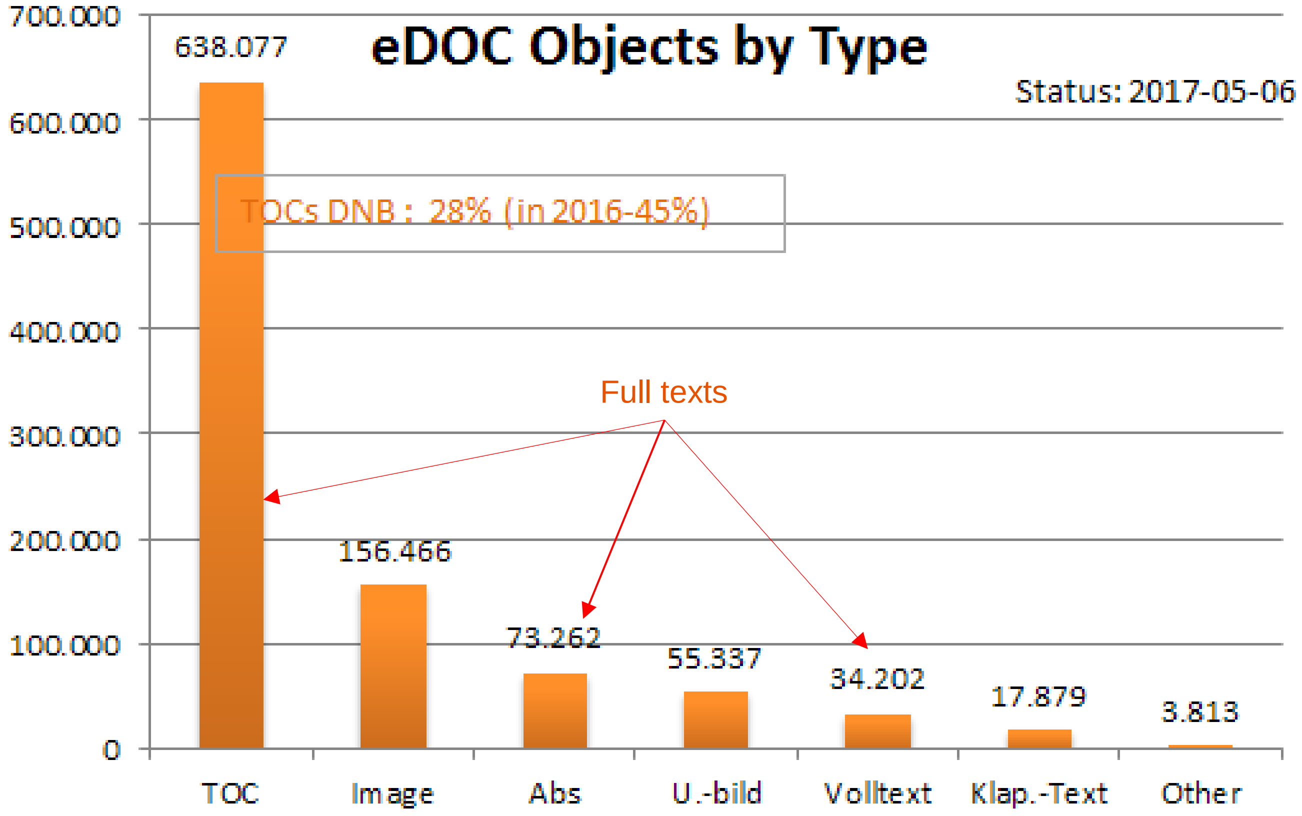
The eDOC objects repository is constantly growing

(2004-11: 5,000, 2017-05: 975,000 objects)



eDOC current status – 3

eDOC contains a considerable number of electronic objects containing full texts and they are searchable in Primo



From Aleph to Alma – eDOC workflow – 1

The eDOC workflow:

Collection of electronic objects from institutions and various services such as

- scans from libraries
- born digital FTs and abstracts – mainly theses and OA publications (from Visual Library and OPUS)
- valuable PDF TOCs from DNB (automatic "harvesting")

Objects ingest into eDOC repository

- each object file name contains ACNR of BIB record

Preparation of catalog update records (adding links to objects) and performing catalog update

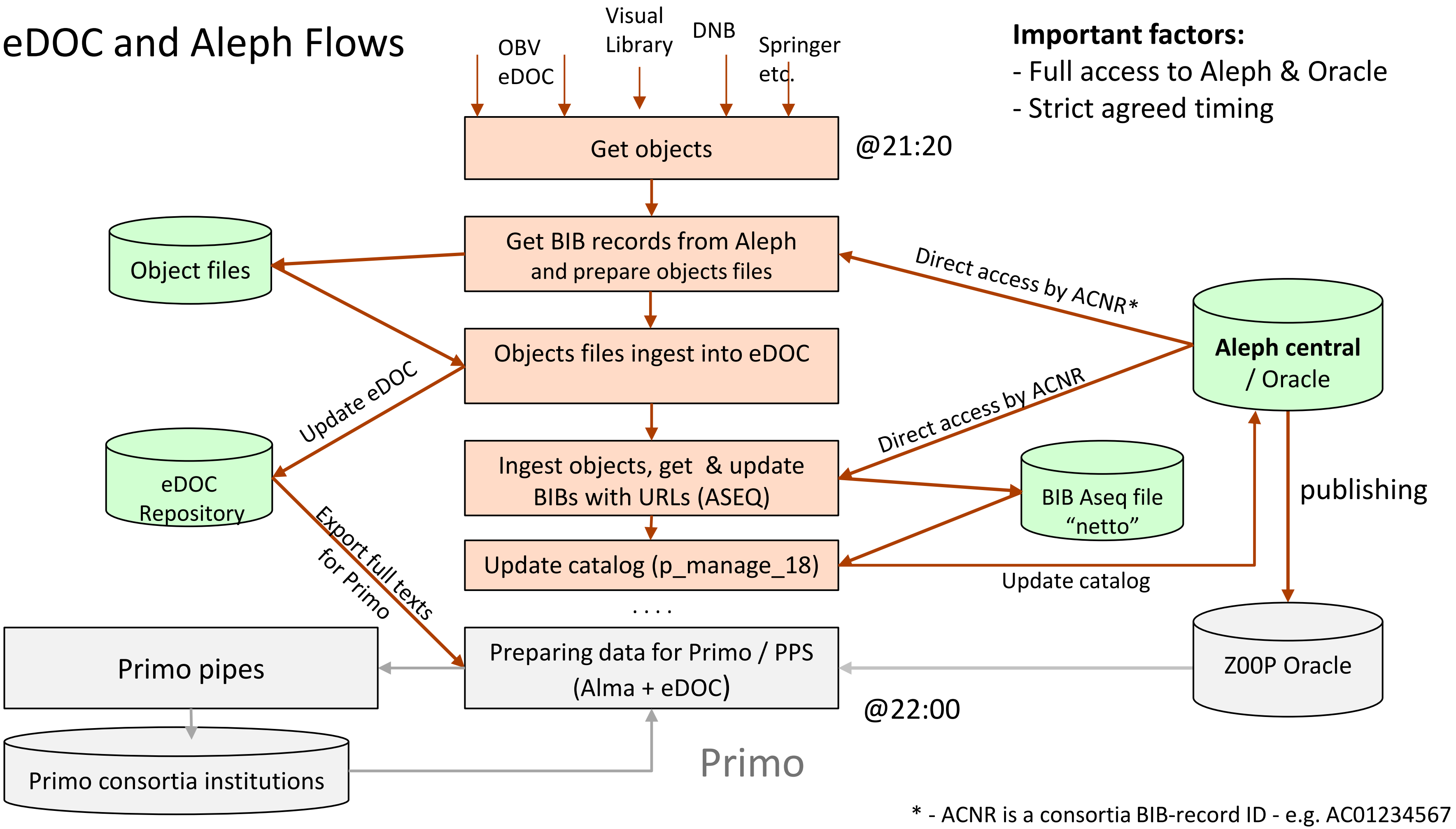
- links in eDOC lead to internally or externally stored objects

From Aleph to Alma – eDOC workflow – 2

eDOC and Aleph Flows

Important factors:

- Full access to Aleph & Oracle
- Strict agreed timing



OBV eDOC
Visual Library
DNB
Springer etc.
Get objects
@21:20
Get BIB records from Aleph and prepare objects files
Object files
Direct access by ACNR*
Aleph central / Oracle
Objects files ingest into eDOC
Update eDOC
eDOC Repository
Direct access by ACNR
Ingest objects, get & update BIBs with URLs (ASEQ)
BIB Aseq file “netto”
publishing
Update catalog (p_manage_18)
Update catalog
Export full texts for Primo
. . . .
Primo pipes
Preparing data for Primo / PPS (Alma + eDOC)
Z00P Oracle
@22:00
Primo consortia institutions
Primo

* - ACNR is a consortia BIB-record ID - e.g. AC01234567

From Aleph to Alma – eDOC workflow – 3

Important aspects of current solution

The full access to data and optimally agreed times are important prerequisites for a successful work flow

All work flow processes running automatically on several servers

→ the base for a good service in heterogeneous environment

Primo receives data and its processing and indexing should finish before 07:00 on next day! Thus latest bibliographic data and full texts are optimally presented and searchable

Can we find solutions for Alma where these specialties will be achieved/satisfied? How could it be done?

From Aleph to Alma – Getting closer to Alma – 1

Alma offers APIs and catalog maintenance jobs in Back Office (BO)

We will need:

Good performance and volumes scalability accessing and updating catalog via APIs and/or jobs

To be able to launch and monitor Alma jobs locally (not in BO!)

- … our workflows starting and running "below the cloud" …

To re-use resources by further using our tools for bibliographic data processing

- we need to adjust MAB2 processing to MARC21
- To process Alma data in the good old "ASEQ"- format (because librarians like it ;-)

→ we see great synergies and resources saving!

It seems we are going to process Alma data like in local Aleph!?

… almost … only where it is possible?

From Aleph to Alma – Getting closer to Alma – 2

Project conceptual design phase (Q1-Q2 2016)

During this (very useful) phase Exl showed interest in our use cases and we opened “tracks” to work out solutions for them

- They are being implemented in phases during 2016 - 2017

It was not easy to convince Exl to implement APIs enabling accessing BIBs by MARC system control numbers” (tag 035) - ACNR, ISBN etc.

- In Alma June 2017 release the 1st version of API supporting building of Alma sets from tag 035 was released

Exl implemented APIs enabling starting Alma jobs and monitor their execution status as we requested

- It means we can start Alma jobs locally and also from our cron files!

From Aleph to Alma – Getting closer to Alma – 3

In early 2016 we began working on Alma REST APIs

Our focus was only on methods accessing and updating BIBs

- access via ACNRs using SRU
- retrieve BIBs by MMS_ID (in 100 record blocks)
- update BIBs using API and import jobs
- export of BIBs using API and export job

Our Aleph performance allows us to update 20K catalog records under 15 min (word indexing ends however in 1.5 hours)

Our goal is to find out how many records can we process between 21:20 - 22:00 using Exl Alma tools optimally?

From Aleph to Alma – API – our development approach – 1

First of all we needed our own data in Alma sandbox

- Data in standard sandboxes were not suitable for our work – too much errors due to its quality (especially nerving were duplicate IDs)

- We selected 20K records from DNB* used in Verbundkatalog, adapted them and imported to Alma. The problem with the data quality was resolved

  → this became our main data pool for development work

* - in Oct. 2016 Mab to Marc converter was not available yet

From Aleph to Alma – API – our development approach – 2

From Aleph to Alma – API – our development approach – 3

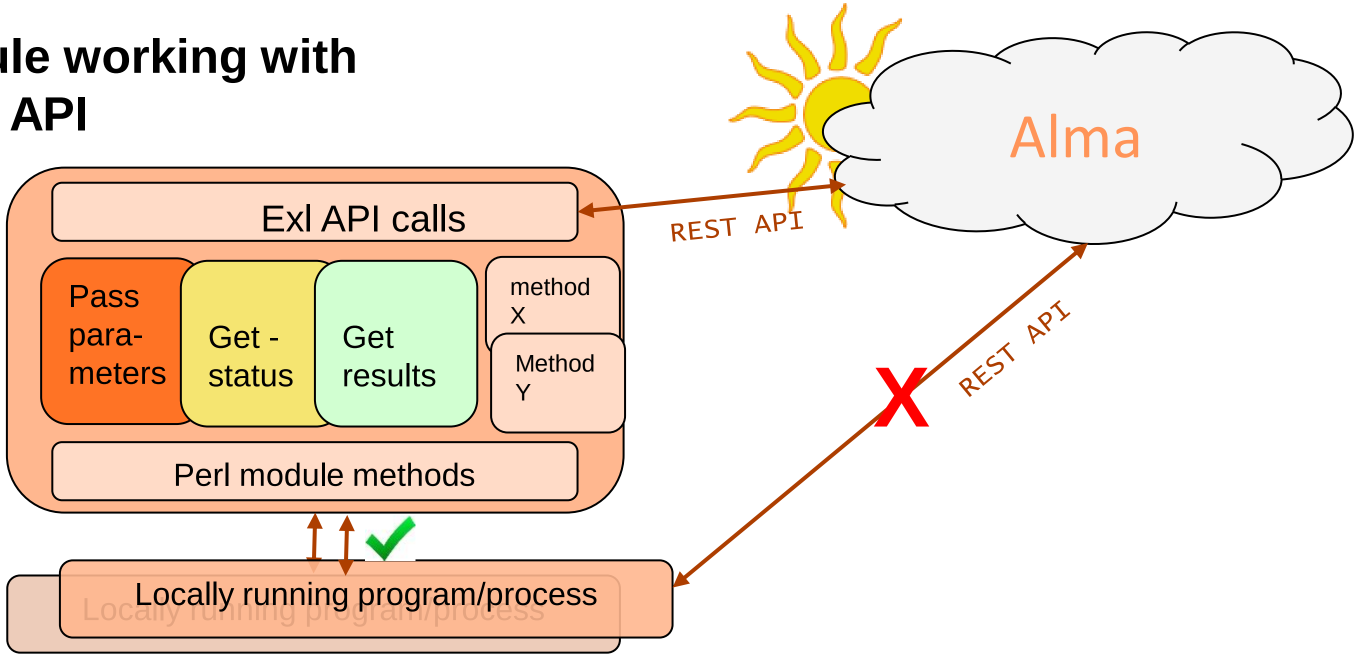
Second, we wanted to integrate Exl REST APIs into our existing “Perl processing“ traditions and make Perl modules for them

- Such modules hide complexity, do it best in getting from API optimal performance and reduce API calls e.g. by grouping data in blocks before calling API etc.
- It is not necessary to know API in details – one can use a simple Perl module call interface and concentrate herself on algorithms of the given task
- A program-caller selects a required method, passes data and gets results from the module which are clearly interpretable

This was another challenge for us to make out of Alma APIs a set of friendly local tools (here REST interface showed its advantages) – we call it also Alma API Framework (or a kind of “middleware”)

From Aleph to Alma – API – our development approach – 4

Example of module working with Alma API



Alma

Exl API calls

REST API

Pass para-meters

Get - status

Get results

method X

Method Y

Perl module methods

REST API

X

Locally running program/process

Example of Perl module call	*“I give you 20K MMS_IDs in array @ArOfMmsids – give me XML records back in the format considering parameters I provided …”*

Example of Perl module call

```
my ($RC, $status) =
    $ObjAPI->ProcBibs(
    \@ArOfMmsids,
    $outputFormat,
    $matchType,
    \%params);
```

“I give you 20K MMS_IDs in array @ArOfMmsids – give me XML records back in the format considering parameters I provided …”

“On return – I will check :
- The return code - $RC first !
- How many records found, how many not, where errors found etc.
- Then I get MARC21 XML records returned (using another method for that)… ..!

From Aleph to Alma – API – our development approach – 5

Having implemented APIs in Perl modules* and adjusted our existing program tools we got “universal” construction blocks which include:

- Export records by BIB ID lists (two approaches: SRU and API managing sets, lists up to 20K were tested)
- converter of MARC21XML to ASEQ
- converter of ASEQ format to MARC21 XML
- Alma records update (two approaches: import job, or single updates via API)
- API-mechanism starting and monitoring Alma jobs (import, export, publishing will come soon)

These blocks we can use in eDOC and also in other central services which we will adapt to Alma

→ they are easy combinable for building other work flows– like any other Perl modules we use

* - the modules are currently in Beta phase

From Aleph to Alma – and the results are …. – 1

How do these blocks work?

What is the performance?

Does Alma outperform Aleph?

From Aleph to Alma – and the results are …. – 2

How do these blocks work?

There were quite many iterations and improvements to implement our framework of APIs as a usable “Beta”

- we learned how to deal with various APIs and to isolate their REST API handling and other complexities into our modules
- very helpful were (and still are) monthly development WebEx meetings with ExI Development

We implemented what ExI delivered, sure, there were some issues and missing parts and we report them to ExI

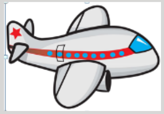
→ often changes coming in monthly releases – a good thing!

And this is our big construction enterprise – taking us over one year

- **eDOC has been adjusted to Alma in February 2017**

From Aleph to Alma – and the results are …. – 3

Performance comparison: 20,000 BIB-records processing

Transport	Export BIBs by ACNRs*	**Update BIBs** (with / without indexing)
	ALMA: 29 min A or 31 min B Var. A SRU ACNR -> MMS_ID (a): 15 min API retrieve BIBs (b): 14 min *Var. B* Build ACNR set: 25.7 min Export BIBs job: 5.3 min	**ALEPH 500: 155 min** Update + word indexing – 155min
		ALMA: 28.5 min Import job: 28.5 min Rec./Sec: 12.2 *Note. It seems indexing was done!?*
	**ALEPH 500 & OBVSG tools: 1.5 min** Resolve ACNR via SYSNR: 54 sec Export as ASEQ file : 30 sec	**ALEPH 500: 10.6 min** Update no index: 10.6 min Rec./sec: 34.6

Notes. „Export BIBs by ACNRs“ – (a) Exl APIs and jobs use MMS_IDs. Our „language“ is BIB ID or ACNr – therefore for certain tasks first we need to resolve ACNR with MMS_ID. (b) SRU BIB format is good but it is not exactly the same like e.g. API „Retrieve BIBs“.

The above performance values measured in Alma sandbox environment. In the future productive environment performance may be higher.

From Aleph to Alma – and the results are …. – 4

Best performance figures

Aleph

BIB IDs resolution plus export + update = 1.5 + 10.6 = **12.1 min**

Note. Longer indexing time is not relevant here.

Alma

BIB IDs resolution (SRU) plus API + update = 29 + 28.5 = **57.5 min**

➔ Alma slows our eDOC work flows by **45** minutes!

From Aleph to Alma – and the results are …. – 5

Performance figures interpretation

- We did not expect that Alma will be faster than our local Aleph!
- We have to sacrifice performance in some batch tasks by adjusting our workflows (in eDOC it was possible)

From Aleph to Alma – and the results are …. – 6

Performance figures interpretation

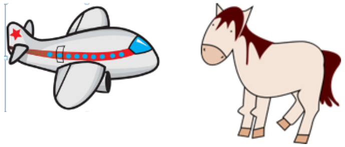
Our re-worked eDOC workflow – or what we’ve learned

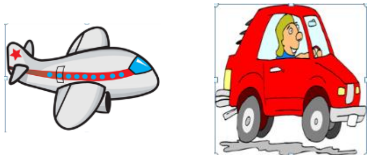
- We cannot afford doing multiple accesses within the workflow to the same record in different steps (was not a problem using SQL)
- We get first the whole scope of BIB records using the fastest way (in blocks) and then process them in all workflow steps
- We do catalog update using import job (started via API) – it provides the best performance
- Volumes scalability is possible – for very big records sets we can iterate through smaller portions still applying the optimal way for each data chunk
- We can easily replace modules providing the same results (e.g. take “build API set” instead of using SRU)

From Aleph to Alma – and the results are …. – 6

And how does eDOC work flow looks for Alma now?

Let’s compare now two diagrams for the current and the new one flows

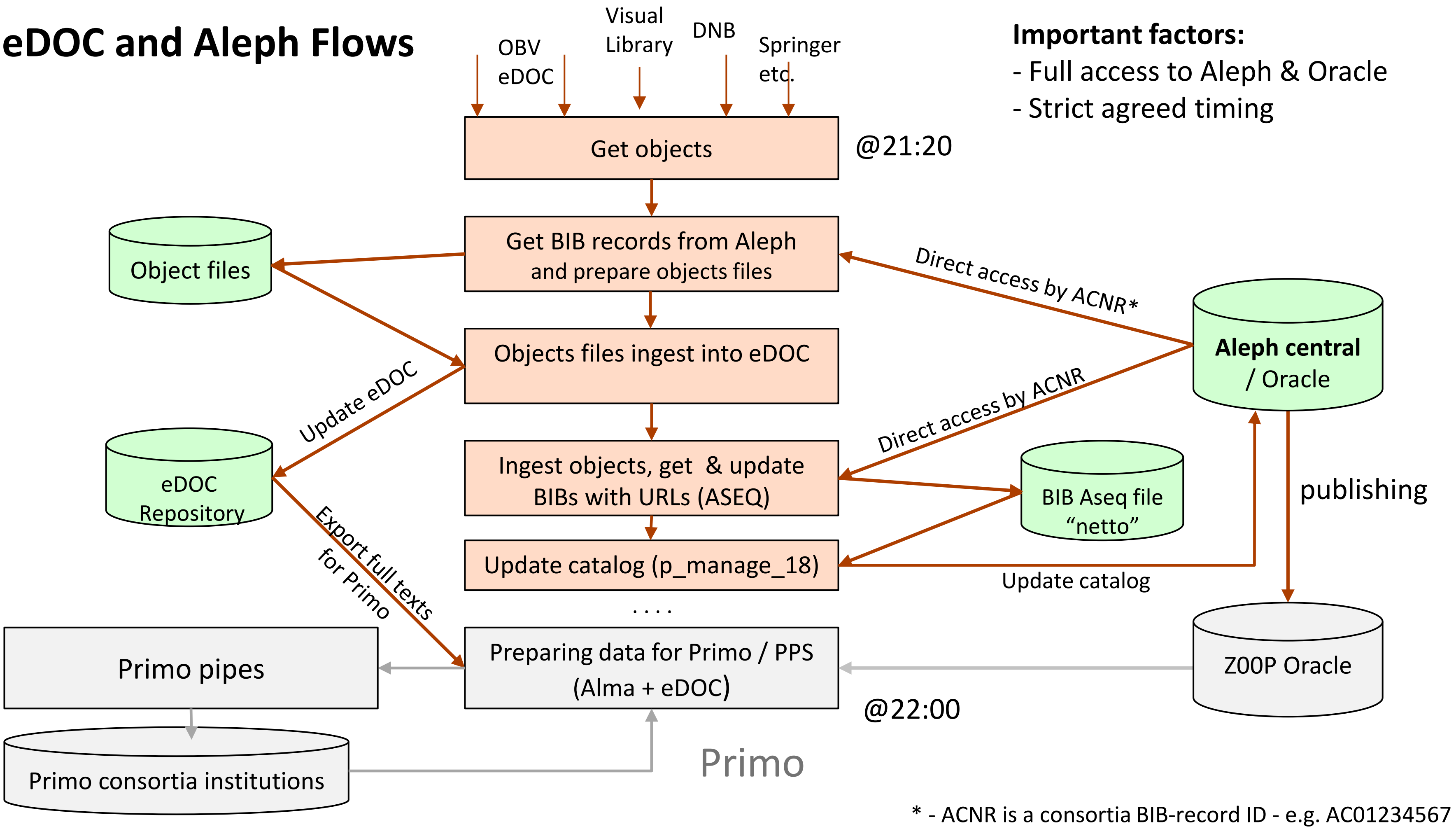




From Aleph to Alma – eDOC workflow Aleph

eDOC and Aleph Flows

Important factors:
- Full access to Aleph & Oracle
- Strict agreed timing

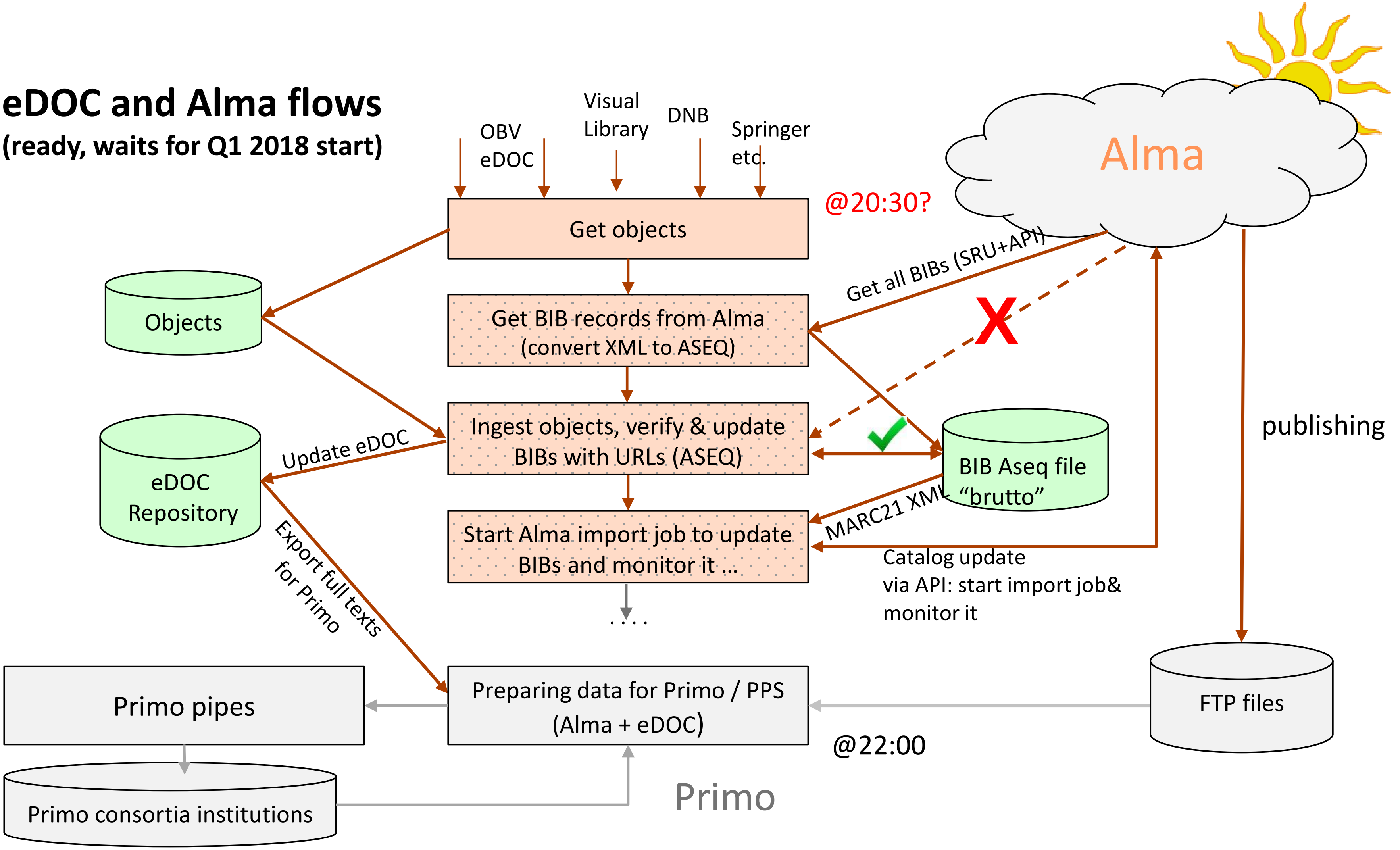

OBV eDOC
Visual Library
DNB
Springer etc.
Get objects
@21:20
Get BIB records from Aleph and prepare objects files
Object files
Direct access by ACNR*
Aleph central / Oracle
Objects files ingest into eDOC
Update eDOC
eDOC Repository
Direct access by ACNR
Ingest objects, get & update BIBs with URLs (ASEQ)
BIB Aseq file “netto”
publishing
Update catalog (p_manage_18)
Update catalog
Export full texts for Primo
. . . .
Primo pipes
Preparing data for Primo / PPS (Alma + eDOC)
Z00P Oracle
@22:00
Primo consortia institutions
Primo

* - ACNR is a consortia BIB-record ID - e.g. AC01234567

From Aleph to Alma – eDOC workflow Alma

eDOC and Alma flows

(ready, waits for Q1 2018 start)



Coming to the end ...

Summary – 1

Alma works differently and offers much more than Aleph. But we still need to better understand its potential and this has its price…

We adjust our ways of working with catalog data in cloud:

- We try to make it easy and still with maximum performance
- We re-use good existing program resources
- And last but not least – keep our good “ground” traditions

Our work with Exl enriched Alma functionalities with elements enabling us to work with workflows locally and control them as we need – with the maximum of automation

Alma API framework which we develop shows its advantages and we have good prospective for its future, even when the work is not fully finished yet

Summary – 2

Work with APIs in the consortia context requires a good organisation of tools, management of complex configurations and a solid expertise

For batch processing we see certain slowness of Alma comparing to Aleph. We do not consider it yet as a very critical but nevertheless we have plans to deal with a local mirror database….

The experience with Alma sandboxes makes us believe that our adjusted workflows will work for us in production as well

And we still have a couple of big central services projects to complete…

The production start in Q1 2018 will give us more information…

We hope for the good results!

Contributors

Petra Ollram – software developer, API tests and Perl modules development

Victor Babitchev – concepts and work with Exl, specifications, tests, development and eDOC

Stefan Majewski – API tests, communications and coordination with Exl, development (from Sep. 2016)

This is the end - thank you!

victor.babitchev@obvsg.at

VB/2017-06-23