FIAF Treasures Analysis

Evaluating Migration of the Legacy Database from Filemaker to a Browser-based OpenSource System (possibly EN15907 compatible)

29. April 2020

Peter Bubestinger-Steindl
(p.bubestinger@av-rd.com)

Thomas Schieder
(t.schieder@av-rd.com)

Table of Contents

Abstract

This document aims to compile information about the Filemaker-based FIAF Treasures database structure in relation to possibilities to migrate it to an open, preferrably web-based platform. If possible, mapping the content to a layout according to the EN15907 standard in this step.

Additionally, the manual labor currently required for ongoing so called “daily imports” shall be considered and possible reduced to support a fully-automated workflow as well as possible. This includes means of deduplification, handling controlled vocabulary as well as detecting and linking related entities.

Therefore, this document contains an analysis of the legacy Filemaker structure, as well as its contents from the point of view of machine readability in its current state and how to improve this, while trying to enable becoming as EN15907 compatible as possible.

In some cases, a CollectiveAccess “Providence” will be used throughout this document as an example platform to show mapping possibilities to the EN15907 standard or simply illustrate possible UI behavior after a migration. CA Providence was chosen, since it is currently the only Open Source collection management system the has this standard implemented and because it is the system that the authors of this document have the most practical experience with yet.

Data Structure Information

Original Source Structure

Although FIAF treasures is originally stored in a Filemaker database, the SQL (MariaDB) dump produced by Darren Mothersele was used for analysis, since it depicts the Filemaker data structure in a way that is faster and easier to deal with. The information is stored in tables and some indexes are used to speed up database operations. Below is an overview of the database tables and their content (tables are listed in alphabetical order). Tables that contain data relevant for a migration are marked as **bold**:

Table name	Description
access	This table contains information about available *access* copies of the Work.
alternate	Titles for the Work (including alternative titles).
archive2016	This seems to be a duplicate of “archive_names”. Maybe a copy/backup from 2016? Possibly obsolete.
archives	Information about Archives
archives_list	A list of stored archive names (archives.arc)
archive_names	Information about Archives (new table imported from FIAF). Seems to be intermediate data for updating/adding the “archives” table entries.
cast_	Cast associated to a Work
cast_list	A list of cast members names (=cast_.cx)
country	Countries assigned to Works
country_code	A list of country two-letter codes. These 1-/2-digit country codes do not seem to be any official standard codes.

Table name	Description
credit	People credited for taking part in a Work
credit_list	A list of people credited for a Work
diacriticcodes	A list of diacritic characters and their codes in XML and unicode, e.g. „&aacute U00E1 á“. It is not clear why and where this was used. It can be assumed that it was used to map certain characters during import/export. (unclear)
director	Directors associated to a Work
director_list	A list of Directors
non_access	This table contains information about available *non-access* copies of the Work.
photography	Photographers associated to a Work
photography_list	A list of Photographers
producer	Producers associated to a Work?
producer_list	A list of Producers
production	Production Companies associated to a Work
production_list	A list of Production Companies
series	Series the Work is a part of
series_list	List of all Series
treasures	This appears to be a meta-table, pretty much containing the information of all other tables in one.
writer	Writers associated to a Work
writer_list	A list of Writers
year	Date Information on creation, modification, …

Some information is not available to us at this time, as it is part of scripting in FileMaker, e.g. splitting of some fields when displaying in the FileMaker GUI or the relationship of fields on a technical level, e.g. displayed values constructed from several fields in FileMaker (like in “Full Film Title”).

Obsolete Legacy Tables

Some of the currently existing Filemaker tables are very likely to be rendered obsolete when migrating to a new system for different reasons: Some contain data that is outdated, replaced by standards (e.g. country codes) or simply superfluous for other reasons.

For example, some tables (*_list) are merely used for indexing and search functions, and therefore to be handled internally by any chosen target platform and/or database.

There should therefore be *no need* to include the following tables and their information in the data migration or future imports:

Table name	Stored information	Obsolescense Reason
archives_list	A list of stored Archives	Not part of the FIAF data, but for search/indexing purposes. Handled by target system.
cast_list	A list of cast members	
credit_list	A list of people credited for a Work	
director_list	A list of Directors	
photography_list	A list of Photographers	
producer_list	A list of Producers	
series_list	A list of Series	

Table name	Stored information	Obsolescense Reason
writer_list	A list of Writers	
archive_names		Seems to be intermediate (temporary) data for updating/ adding the “archives” table entries, based on external data provided by FIAF. It also seems to have been used to change the naming style, as well as updating a 3-letter (column “old_code”) code to 4-letters (column “major_key”).
archive2016		This seems to be a duplicate of “archive_names”. Maybe a copy/backup from 2016?
country_code		These country codes are non-standard and should be replaced by referring to an official standard (e.g. ISO 3166-2).
diacriticcodes	A list of different diacritic character encodings in different norms (unicode, HTML entities, …)	This information is available in official listings, and not required to be stored/migrated in the target database, since this mapping functionality should be handled by any data import/export method. For example the following listing on "key-shortcut.com" as a reference.
treasures		This table seems to *mostly* contain a merged duplicate copy of data stored in other tables. The only values that seem to be originally unique in this table are: the Film title (.tix), end user notes (.note) and NFPF notes (.nfpf). Even though it appears to hold all significant data of the other tables, it can *not be used* as a migration source, as it also contains concatenated information (data from multiple tables stored in one treasures.field, e.g. “Full Film Title”). However, it may be used as a reference to check if the information layout after a migration covers the legacy functionality, since the structure and contents of this table are used to export data to other platforms for web presentation.

Source Schema Information

For these core fields, information is known and can be used to map fields (values that appeared uniquely in the “treasures” table have been marked bold, since other values in the treasures table seem to be obsolete duplicates).

Fieldname (XML)	Fieldname (by Tables)	Description
an	*.an	ID number (“Accession Number”). 5 digits. This is the foreign key used to relate the data in all tables together.
record_id	*.record_id	ID number. Seems unrelated to AN.
fi	Combined field formatted as: “FT (FC, FD, FY)”	Full Film Title
ca	cast.cx	Cast Name
fw	writer.wx	Writer Name

Fieldname (XML)	Fieldname (by Tables)	Description
ph	photography.photx	Photographers Name
cr	credit.credx	Credits
se	series.serx	Series
ar	archives.arc	String contains multiple data: “Archive Name (Country) [XXXX]” Where XXXX is a 4 letter identifier (IMIS code?) Example (id_archive = 117): arc = “BFI National Archive (London) [GBLB]”
--	archives.arc_bu	Similar to “.arc”, but seems to contain an older, possibly outdated naming. It uses a 3-digit code, which seems to have been used before 2016 (see “archive2016” table) Example (id_archive = 117): arc_bu = “bfi/National Film and Television Archive (London) [GBB]”
--	archives.imis_code	Contains the same 4-letter identifier as present in “.arc” and “.arc_bu” (but without “[]”)
pc	production.pcox	Production Company Name
fp	producer.px	Producer Name
fd	director.dx	Director Name
ft	**treasures**.tix + alternate.atix	Film Title and alternative titles
fc	country.cpx_code	Film Country (2 characters)
fy	year.yrx	Film Year
ah	access.arc2	Access copy
nh	non_access.nhx	Non-access copy (previously called AO)
nt	**treasures**.note	End user note
nf	**treasures**.nfpf	NFPF Note (Contains a remark if fim has been preserved by the National Film Preservation Foundation)

There is a lot more information stored that has not yet been defined in detail for further use in mapping data on a new target platform. From spot checking the actual data in these fields, it seems that most of them are very likely superfluous. The reply from Platon Alexiades regarding these undocumented fields is as follows:

> *“In any case, the tables with ‘unknown fields or unknown tables’ (if any) should be disregarded and can be deleted. They may have been used in the past for some ad hoc import. They will not be used in the future.”*

Here is a list of these possible obsolete data fields:

ti2x	fc2	dxx2	wxx2	serxc2
tix_net	fy2	dx_txt_copy	wxc2	arcxx
tix_net2	dx_chk	fc2_copy	photxx	arcxx2
tixx	dx_imp	fy2_copy	photxx2	arcxc2
tixx2	dx_txt	d3x_copy	photoxc2	pcoxx
import_id	dx_inverted	tixc	credxx	pcoxx2
d3x	dx_inverted_cleaned	cxx	credxx2	pcoxc2
d3xx	dxc	cxx2	credxc2	pxx

d3xxx	dxc2	cxc2	serxx	pxx2
d3xxx2	dxx	wxx	serxx2	pxc2
atixx	arc2xx2	nfpfc2	cx_txt	arc2x
atixx2	arc2xx	nfpfxx	wx_txt	nhx
atixc2	nhxc2	nfpfxx2	photx_txt	arcx
cpx_code	nhxx2	gname	credx_txt	serx
cpxx2	nhxx	serx_txt	arc	filmsearch
cpxx	notec	pcox_txt	arc2	filmsearch2
yrxc2	notec2	cpx	nhx_txt	filmsearch3
yrxx2	notexx	yrx	ati_txt	filmsearch4
yrxx	notexx2	px_txt	atix	filmsearch5
arc2xc2	nfpfc	dx_txt_2	credx	tix_copy
tix_txt	test	archives_count	id	tix_check
tix_2w	limk			

However, there seems to be some syntax to these field names:

Description	**Examples**
The original fields often ended with “x”	Title = **tix**
For each copy, another “x” was added:	**tixx**
Some copies were marked with a “c”	**tixc**
Some copies received a numeric counter. Not to be mistaken with the suffix “_2w”, which means “The first 2 words only”.	**tixx2**, **atixc2**
Text variations prepared for public display were marked with “_txt”	**tix_txt**
dx_inverted	Director’s names reversed from “Lastname, Firstname” to “Firstname Lastname”

EN15907 – Basic Structure and Entities

The standard defines the ‘Cinematographic Work’ (EN15907, 4.1) as the topmost level of description. It already holds information, or links to, Agents, Content, Subjects, Variants, Manifestations and more. Further it can be in relation to other Entities as depicted in the very compressed description below.

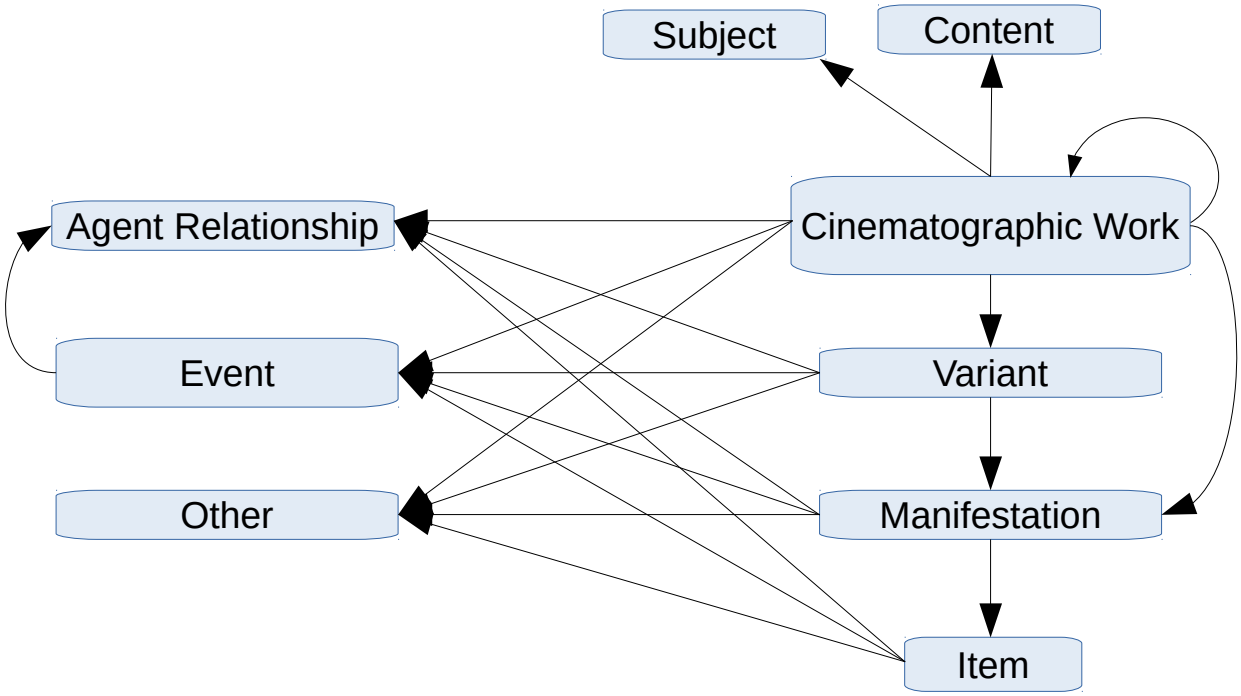


Figure 1: EN15907 Entities

Variant, Manifestation and Item Entities (EN15907, 4.2 – 4.4) are used to describe different versions of the same Cinematographic Work (Variant, descriptive level), the physical manifestations (Manifestation, a physical carrier or file) and finally complete, incomplete, defective, fragments or otherwise related specific (unique) physical carriers or files (Items)

Content Entities (EN15907, 4.5) are statements about the content of a Cinematographic Work and consist of the elements ‘Subject term’ (e.g. genre) and Content description (textual description of the content of a Work).

Entities of the type ‘Agent’ (EN15907, 5.1) are used to represent Persons, Corporate Bodies, Family and Person Groups that are related to the Cinematographic Work.

Event Entities (EN15907, 5.2) characterise occurrences in the life of a Cinematographic Work and can be linked to other Entities, such as Variant, Manifestation and Item. The currently defined Event types for Publications, Decisions, IPR registrations, Awards, Production and Preservation Events.

Mapping Treasures to EN15907

The structure defined in the EN15907 depicts a 4-level entity relationship: Work, Variant, Manifestation and Item. Some institutions who already use EN15907 for their daily work, have only implemented a 3-level layout: Work, Manifestation and Item. There is still discussion going on in the film archive domain about pros and cons of the different approaches.

The data stored in the Treasures database only depicts a flat, more classic structure (mainly with information from Work and Item) – for legacy reasons, since a majority of film archives have their cataloging data in a similar structure. In order to evolve this towards a 3-or-4 level structure (EN15907 compatible) and also to conform with the requirements of some fields (e.g.: mandatory, defined vocabulary, etc), some decisions

have to be made to perform an initial migration mapping. The conclusions and decisions drawn from this migration step can then be re-used for ongoing daily imports.

Data Mapping Table

The following table shows an overview for mapping known Treasures data to EN15907 data fields. Due to the structural differences, the mapping will be to entities and the corresponding field(s), where due to the old structure some data will have to be split into multiple fields or entries.

XML	Filemaker	Content	EN15907 Entity	EN15907 Field(s)	Comments
an	*.an	ID number (“Accession Number”). 5 digits. This is the foreign key used to relate the data in all tables together.	Work	Identifier	
record_id	*.record_id	ID number. Seems unrelated to AN.	Work	Identifier	
fi	Combined field formatted as: “FT (FC, FD, FY)”	Full Film Title	Work	Identifying Title	The original concatenation syntax of Treasures must be maintained by a migration, because it served (and was used) as identifier in e.g. other databases or the International Index to Film Periodicals, and possibly others.
ca	cast.cx	Cast Name	Agent [Person]	Name	High possibility for duplicate entries. Syntax is mostly: “Last Name, First Name”. It may be good to split the Name into separate fields (First name, middle name, last name, etc). Given the assumption that many archives may only have agent names as a single (non-split) text string, the question remains how a normalized syntax could be implemented. Same goes for optional name parts like: middle name, nickname or alias.
fw	writer.wx	Writer Name	Agent [Person]	Name	
ph	photography.photx	Photographers Name	Agent [Person]	Name	
cr	credit.credx	Names of credited persons	Agent [Person]	Name	
se	series.serx	Name of series, the work is part of.	Work	Name [descriptionLevel = s|c]?	There may be different options how to depict this. They should be discussed.
ar	archives.arc	“Archive Name (Country) [ID]”	Work	RecordSource [SourceName, SourceIdentifier]	String contains multiple data: “Archive Name (Country) [XXXX]” Where XXXX is a 4 letter “imis_code” of the archive.
pc	production.pcox	Production Company Name	Agent [Organization]	Name	
fp	producer.px	Producer Name	Agent [Person]	Name	Similar to other Agents listed above (Cast, Writer, etc)
fd	director.dx	Director Name	Agent [Person]	Name	
ft	treasures.tix	Film Title	Work /	Title	

	alternate.atix	Alternative titles	Manifestation		
fc	country.cpx_code	Film Country (2 characters)	Work	Country of Reference [Production]	
fy	year.yrx	Film Year	Work	Year of Reference [Production]	
ah	access.arc2	Access copy	Item / Manifestation	?	String contains multiple in inkonsistent formatting. Example: “16 mm acetate positive: MXMF”
nh	non_access.nhx	Non-access copy	Item / Manifestation	?	String contains multiple in inkonsistent formatting. Example: “35 mm acetate master positive: USWL”
nt	treasures.note	End user note	Work	Notes	Additional field not in EN15907
nf	treasures.nfpf	NFPF Note (Contains a remark if fim has been preserved by the National Film Preservation Foundation)	Work	NFPF Notes	Additional field not in EN15907

full_film_title: This field is a concatenation of the fields “film title”, “film country”, “film director” and “film year”. As those fields are also available stand-alone, the information should be imported from those. That way is a lot safer for automated processing compared to using the “full_film_title” field and split it back into single fields. The original concatenation syntax must be preserved (was used as reference identifier in other databases and film related writings)

Agent entities (person): Those are “cast”, “writer”, “photography”, “credits”, “producer” and “film director”. The default format, as also declared in information received from Platon Alexiades is “Last Name, First Name, Initial”. Having a closer look at values stored, it showed that the data contained is not always consistent with the described default format.

As an example, the “7th Heaven” Work already has two special cases that make automatic processing difficult. One is “`Stone, George E. (Georgie)`”, which contains additional information in the form of “E.” and “(Georgie)”. The other is “`Valentine, J. A.`”, where it is not clear how to map the initials. Any automated processing of non-standard field information would be guesswork.

Additionally, it is a realistic expectation that the same person can be found within the source data in multiple instances, e.g. “Stone, George”, “Stone, George E,”, “Stone, George E. (Georgie)” and “Stone, Georg”. To minimize this, a de-duplification process should be defined for the migration.

The proposed way to handle this and also reach a good level of catching duplicates is a semi-automatic approach. All fields of this type can be automatically parsed by a script to identify which entries are sticking strictly to common naming string-syntax occurences, such as:

- “Last Name, First Name”
- “Last Name, First Name, Initial.”
- “Last Name, First Name (Nickname)”
- “Last Name, First Name, Initial. (Nickname)”

Those can be split and mapped with a good chance that little false-positive mappings will occur. All entries that are identified that do not match this pattern should be inspected by human eyes and checked for validity and ways to correct the entry. Additionally, a search can list all entries, roughly grouped by how alike they

are, to check for possible typo-related duplicates (e.g. “Stone, George” vs. “Stone, Georg”). Cleaning as much of those entries prior to migration might help the process significantly.

Archive: There is an entry for each Archive in the format “Archive Name (Location) [Archive Identifier]”.

These values have to be split into single fields to populate the list of Archives on the target platform with the corresponding data while- or post-migration. For EN15907, an Archive is a “Record source” and only has values for the Archive name and Archive identifier, but not for the location. It is to be discussed if the Archive identifier is already specific enough (=unique) so the location information can be discarded, or alternatively, where to store this auxiliary information in the new schema.

Analogue to Agent entities, handling this prior to the migration and doing a human eye check on a generated and sorted list of available entries, might have significant positive impact on the data quality due to duplicate and typo detection.

Film title: Multiple titles for a Work are stored in a single string, using “§” as a delimiter. The current schema holds the Film title in different languages and probably a multitude of variants, so preprocessing is required.

Depending on the target platform, this string has to be dismantled and changed into single entries where it has to be clearly determined what the “main” Film title is, and how to treat the others. This is mainly as the decision has to be made how to treat different titles on the new platform upon migration. The automatic import can not decide of which type an additional title is, e.g. “Original Title”, “Alternative Title” or “Translation”, also it is not sure to assume that the first Title in the string is always the “Original Title”. A human check is necessary when preprocessing information of this type.

Access / non-access: Information about the physical film copies – either access or non-access – are stored in a string in the format “Format: [Archive Identifier]”. The Archive Identifiers here ideally are matching with those used for “Archive” entities (see above). As with Film titles, it has to be decided if information is used either to be stored the main work entry or consequently create an Item for every copy in existence, storing the information there.

For example, the list of access-copies on our 7th Heaven example contains this information: `“35 mm: USNM|16 mm: USWL”`, indicating two existing access copies in the USNM and USWL Archive, respectively. However, the note field of the same entry reads as following: `“USWL: 16 mm is Killiam reissue version”`.

Please refer to the chapter “Physical Film Copy Information” for more details on how to deal with this data.

Initial Migration Considerations

This chapter is about the initial migration of the Treasures database to an EN15907 conform data structure. It includes suggestions for cleaning the source data, as well as pre-processing requirements to support a clear conversion towards EN15907. Some of the mechanisms and requirements described here may be re-used for the regular data imports during normal operation in any new system.

Import Test in CollectiveAccess

A test import of the Treasures data in CollectiveAccess was tried in order to get more practical insights which might be overlooked or not-noticable by looking at data dumps alone.

This test not only confirmed some expected complexity of the legacy data layout, e.g. due to the many linked entities that are stored in flat text (separated by “|”, “§”, etc), but also that the import would have to split up into separate steps in order to be able to properly link the newly created entities to each other.

The default XML importer of CA might be able to be configured to handle these situations, but the complexity began to reach unpractical levels. Alternative means of importing the existing data to the new platform of choice, might be the better option.

The insights gained from this test are included throughout this document.

Data Access: XML vs SQL?

XML is great, but has its limitations if the data is not in the final layout. For case of migrating the Treasures data, the preferred approach is a direct, read-only database connection (e.g. ODBC) to Filemaker (as successfully done in the past by Darren Mothersele). This method allows to retrieve not only the individual data fields, but also whole data-sets generated by SQL queries. This would allow certain pre-processing, renaming or re-combination of fields to better suit the target data layout. SQL is also the preferred (and often faster/easier) way for dealing with accessing relational data (such as Treasures), since transformations and preprocessing, as well as possible renaming and re-contextualization of the source data is required.

For this use case, SQL access is usually easier and better to handle from within any programming language, rather than having to query across separate XML nodes or other file-based data formats, such as CSV for example.

Depending on the chosen target platform and how its database layout is structured, or if it provides an interface (API), one of the following 2 methods could be used for creating or modifying entries during an import:

1. Direct database access.
2. API access.

This also applies for ongoing future imports.

While direct database access allows raw access to handle the data and therefore have full control and decision options over the imported data, an API may provide certain built-in preprocessing, sanity checks, etc – that make sure that the created data is consistent with the target platform's internal workings. In comparison, using direct database access without having the full knowledge of the internal workings of the target system may create data conditions that may exhibit erroneous behavior (or worse) when accessed through the target system.

Therefore, if an API exists it is advised to prefer this method over direct database access.

Any import method is prone to generate duplicates in case of typing errors in the source data, but this can easily be resolved in some (semi-)manual, possibly script-assisted checks post-import.

The actual importer should then start at the highest level content (Cinematographic Work), create it, and link the now already existing entities to it. This would be a recursive operation down the description levels, so the next would be Manifestation (so it can be linked to the now imported Work), and Items. A possible alternative would be a minimalistic approach, which only uses Work and Item objects.

As an alternative, it is possible to do the same from a database dump. However, working read-only from the source database has the big bonus of not creating a time-related delta between the live and the imported dataset.

Controlled Vocabulary Terms

Some of the incoming data needs to be mapped to controlled vocabulary terms. The EN15907 contains controlled vocabulary fields, but does not define which vocabularies to use. Therefore it has to be decided which vocabulary terms to use for any data import. The choice made for an initial migration will per definition affect future imports as well.

As a starting point, we suggest to use the vocabulary lists defined and used by the "European Film Gateway" (EFG) project (http://www.europeanfilmgateway.eu/): The metadata schema for delivering data to EFG is defined based on EN15907 and supported, and therefore practically tested, by a significant number of film archives. The vocabularies referred to in the FIAF cataloging manual are also a good source, but seem a bit less elaborate than EFG's.

The current list of vocabularies defined in EFG can be found online as a spreadsheet: https://efgproject.eu/downloads/EFGVoc_Values_ElementTypes_public_110510.xls

The following source data needs to be mapped to controlled vocabulary terms:

- Any country: to 7.2 Region (ISO 3166-2, AFNOR XP Z44-002 or MARC)
- Any person: to 5.1 Agent (Agent.AgentType, HasAgent.Activity)

Whereas for the following data sources, mapping to controlled vocabulary terms is suggested, but not defined mandatory in the standard:

- Access/Non-Access copies: Format, Gauge, InstantiationType

Countries

The country codes used in the FIAF Treasures database follow a rather unknown standard encoding (International Index to Film Periodicals). A copy of this Index, weblink or a DOI/oai would be great to have as a source mapping reference. It is suggested to translate (=map) them to a more widely known standard. Best-practice would be to follow the definitions in EN15907 "7.2 Region":

- ‘ISO 3166-2’ for countries currently in existence.
- ‘AFNOR XP Z44-002’ for historical countries.

Agent Activities

Below are tables that contain information about people or institutions, which shall be mapped to “Agents” in EN15907. The terms for Agent activities are taken from “EFG.TypeOfActivity”. In cases where EFG offers more than one possibly matching term, the preferred term is listed as first option and marked in italic.

Source Table	EN15907 Mapping
cast_	Agent.**type** = Person Agent.**name** = cast_.cx HasAgent.**activity** = Actor / Actress [EFG.TypeOfActivity]
credit	Agent.**type** = Person / Corporate Body Agent.**name** = credit.credx HasAgent.**activity** = Honoured to [EFG.TypeOfActivity]
director	Agent.**type** = Person Agent.**name** = director.dx HasAgent.activity = Director [EFG.TypeOfActivity]
photography	Agent.**type** = Person Agent.**name** = photography.photx HasAgent.**activity** = *Director of Photography*; Photographer; Still Photography [EFG.TypeOfActivity]
producer	Agent.**type** = Person Agent.**name** = producer.px HasAgent.**activity** = *Producer*; Co-Producer; Executive Producer; Line Producer [EFG.TypeOfActivity]
writer	Agent.**type** = Person Agent.**name** = writer.wx HasAgent.**activity** = *Screenplay*; Author [EFG.TypeOfActivity]

Enriching of the available Agent types for higher granulation is possible, as EFG.TypeOfActivity holds a total of 173 definitions. This is probably irrelevant for an initial Treasures migration, but to be considered for future imports if additional information that involves Agents is desired in the future.

Mapping Issues / Decisions

Lack of Proper Target Field

In regards of storing information that has no defined field in EN15907 (or any other schema) by itself, the decision has to be made whether to “misuse” an existing field, or to add an additional (non-standard) field to at least retain the source data information. The first option is discouraged, as it may lead to interoperability issues, diminishing the purpose of adhering to a metadata standard.

An additional field would be outside of the EN15907 standard, therefore not impeding interoperability (if e.g. it is simply left out when exporting data for others). The information stored in these temporary fields could then (semi-)manually be moved (copy/paste) by a human operator later on. This approach would allow to complete an initial migration without data being left behind, even if not everything could be mapped seamlessly to a new data layout.

Physical Film Copy Information

Because Treasures stores information about the physical copies (access & non-access) as free text that contains separate information in a single field: Which format of the film copy and a 4-letter identifier (imis_code) of the archive holding it. This requires preprocessing to correctly split the text into individual fields.

The syntax of the tex in the Filemaker fields “access.arc2” and “non_access.nhx” seems to be: “FORMAT: HOLDING_INSTITUTION[, HOLDING_INSTITUTION]”

Examples are:

- “35 mm: USWL, FRPC”
- “16 mm: ITGC, USRG”
- “Format unspecified: GBLB”
- “35 mm acetate dupe negative: USWL”
- “16 mm master positive: USRG”

The information about the format of the film is usually the Gauge, but sometimes also contains the InstantiationType. Multiple Holding Institutions may be delimited by a comma. It can not be assumed that it is guaranteed that spelling and phrasing will be consitent and error-free throughout the data set. This may cause mapping issues/errors during initial migration.

The information about the format of physical copies is defined in EN15907 in “4.3 Manifestations”, Element “6.7 Format”, whereas the HoldingInstitution and InstantiationType is defined in “4.4 Item”.

Preprocessing Suggestion

The following pre-processing steps would be suggested in order to separate and normalize the source data:

1. **Separate Format from Holding Institution:**
   Iterate through all (non-)access table entries and split the string at the colon (:), storing the left side as Format and the right side as a list of Holding Institutions.
2. **Extract known vocabulary terms:**
   Split the extracted Format string by terms common in the dataset: like “16 mm”, “35 mm”, “Format unspecified” - as well as: “nitrate”, “positive”, “dupe”, etc.
3. **Refine vocabulary terms:**
   After having extracted (=removed) these strings from the text in the (non-)access tables, one can create a list of terms that weren’t matched.

4. **Spot mistakes/inconsistencies:**
   Use this list to spot typing mistakes/inconsistencies as well as to know which terms are actually used in the Treasures data.

This separately accessible data resulting from these pre-processing steps can now be used to clean and normalize the source dataset. It can be assumed that the number of distinct values of the extracted data fields (Gauge, InstantiationType, Holding Institution) will be rather low, which is good. NOTE: Depending on how the final target system is configured/implemented, Gauge and InstantiationType may require to be mapped to a controlled vocabulary.

The Note Field

The Treasures data contains a note field described as “End User can enter a note about the film”. The currently contained information suggests that this information has no corresponding field in EN15907. It therefore would make sense to store this information in a text field which is added regardless of the target data schema, and kept unmodified as-is during the initial migration import. It can then later be dealt with by a human operator who can pick apart and re-assign the contained information into its proper target fields.

In practice it contains mostly different comments about the manifestation/items of the film, but also other comments. Some examples are:

- “USWL: nitrate negative is incomplete (1 reel only)”
- “A short film made to promote Ripon, a Cathedral ci…”
- “AUCF: Fragment and chariot race sequence”
- “LUDC: Restored in 1999 with a new musical score”

EN15907 does not offer a generic notes field like this. Other existing field options – which are either not really defined for this purpose or it’s almost impossible to populate properly during an automated migration – can also not be used automatically, since the source note cannot clearly be distinguished to be of a certain type.

Information currently spotted in the Treasures’ note field, suggests the following target fields where an operator could manually copy/paste the text to where it properly belongs. Once all these note fields have been manually migrated, the temporary data field can be removed.

Changelog

Currently, the Filemaker structure includes fields per table that allow logging *who* and *when* a value has been created or modified in the database. The following fields currently exist in every Filemaker table:

Field name	**Stored information**
created_account	Name of the operator/user who created this data. These fields seem to contain the identical information.
created_name	
date_created	The date when this value was created
timestamp_created	The time when this value was created. In the SQL dump, this field contains both date and time, which might make “date_created” superfluous.
modified_account	Identical to the fields for creation, but updated on modification of values.
modified_name	
date_modified	
timestamp_modified	

It is noteworthy, that the current dataset seems *not to contain* creation information for most entries, and the “modified_account” = “modified_name” is “Platon Alexiades” or “fiaf” in almost all cases. It does contain the date/time of modifications however, but their identical timestamps suggest bulk-updates of several entries at the same time.

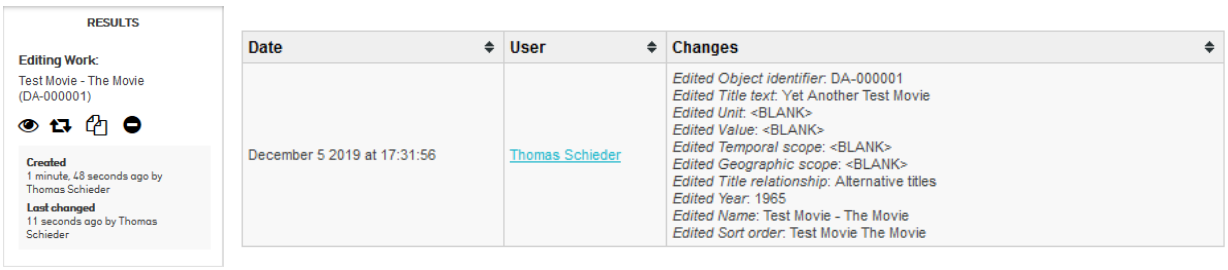
It is quite common for recent collection management systems to already keep track of “who has done what” by noting a timestamp and some information for most actions. In this document we refer to this as “Changelog”. Therefore this functionality should already be implemented in any new target system, and there should be no need to keep the log-fields mentioned above after an initial migration.

If desired though, this legacy Changelog-information could be imported if deemed important enough. Since this data is not content per-se, it is unclear where to store it on an initial migration. Possible options may be:

1. A separate (non-standard) textfield per entity.
2. In the Changelog tables/structure used by the target system.
3. Maybe as EN15907 preservation event, but this may be a bit overkill.

In any case, it might be wise to have a look at the actually stored information beforehand and decide which information to keep or if it is better to thin out and start with a clean “Changelog” and leave the old data behind.

To give an impression what future Changelog entries may look like, here an example of CollectiveAccess keeping track of changes to the entity “Work”, recording the User, date/time and type of changes:



Data Imports

Required logic/programming

Regardless which import method will be used in the end for the initial migration, it will require some logic (=programming) work in order to evolve the current data into a clearer structured form: better suitable for machine processing, reduce disambiguation guesswork and improve general interoperability.

The required programming logic addresses the following necessities:

- Splitting text fields that contain combined information.
  (e.g. Person/Institution names, physical copy information, etc)
- Check if to-be imported entities already exist or have to be created anew.
  (Especially Persons/Institutions)
- Map current free-text field values to controlled vocabularies.
- Normalize/translate source vocabulary terms to target terms.
- Handle (create/update) relationships between imported entities.

It was given as a requirement, that ongoing daily imports must not require progamming logic like this, but should be delivered as ingest-ready as possible.

Suggestions for Delivered Data

There are some steps that can be taken to reduce necessary disambiguation and other guesswork when receiving datasets for Treasures in the future. Some of them involve preprocessing the data received from the source archives, as well as suggestions for improving the data quality of the delivered data in the first place.

Some of these suggestions may even be used by source archives to clean and improve their data generally in their in-house catalogue, making it easier for any future data exchange with others.

Define and provide controlled vocabularies

It is mandatory that the FIAF Treasures have a clearly defined list of which vocabulary terms for which data fields to use. If some terms are known to have different, but common forms among the providing film archives, normalization/translation tables could be provided by FIAF.

Preferrably, source archives implement these vocabularies in their collection – or already perform the normalization/translation during export, if possible. This could even be set as a mandatory requirement by FIAF, if desired.

Examples:

- **Languages:**
  An archive may store language information in different forms, identifiable and consistent within their dataset, but not seamlessly interoperable: if it is just a text-string in their local language (like “German”, “English”, etc) - or a non-standard code (“GE” for German instead of “de”). It shall be output to ISO639 language codes.
- **Film gauges:**
  “35mm” vs “35 mm” vs “35 millimètre”, etc.
- **Instantiation types:**
  “duplicate” vs “dupe” vs “dup”, etc.

One information per field

If any field should contain more than a single kind of information per data field, it shall be split and declared as early and clearly as possible. Different source archives may have different separators or syntax policies defined, but these are most likely to be consistent at the source. When leaving this splitting to an importer at FIAF's side, variations (and possible errors) – and therefore programming and administration overhead – may accumulate.

Examples:

- **Item / Manifestation: Gauge and format as separate fields**
  YES: "35 mm" / "acetate dupe negative"
  NO: "35 mm acetate dupe negative"
- **Person: Each part of the name as individual field**
  (rather than 1 textfield with the whole name)
  YES: "John" / "Smith" / "Johnny"
  NO: "Smith, John 'Johnny'"
- **Holding institution: Archive name, Location and Identifier as separate fields**
  YES: Libary of Congress" / "Washington" / "USW"
  NO: "Libary of Congress (Washington) [USW]"

Provide identifiers for entities

A lot of work (and possible errors) during import is due to having to match the source data fields in a way to "guess" if that may depict a certain entity.

Examples:

- **Work:**
  Matching rule = Similar title + directory/producer/writer + dates (+/- 1 year)
- **Agent:**
  Matching rule = Similar name(s).
  Even if some source datasets may contain more than just the name, such as birthdate, location, etc – precise matching is neither trivial nor flawless.

These identifiers can be kept stored as additional identifiers for entities in the Treasures database. They can then be used to avoid duplicates or guesswork during future imports. It is preferred to refer to use external identifiers such as for example Wikidata, EIDR or ISAN. Identifiers only unique in the source archive may also be used, but in any case the schema in which the ID is valid/defined must be declared for each identifier stored.

A structure as defined in EN15907 "6.1 Identifier" is perfectly suitable for this, even if the data schema is not EN15907.

Satisfy minimum dataset requirements (EN 15744)

Since it cannot be assumedfor the time being, that in the near future the majority of works will come with a proper identifer, it may be helpful for the source data to contain a defined minimum set of fields. The definition in the EN 15744 ("Film identification - Minimum set of metadata for cinematographic works") was used as a reference here.

The current Treasures dataset covers almost all fields, except:

- Duration/Length
- Language
- Genre

Here is the list of metadata fields, as defined in EN 15744:

Required Field (EN 15744)	**Present in current Treasures dataset**	**Target Entity (EN15907)**
Title	Yes	Work
Series / Serial	Yes	Work
Cast	Yes	Work
Credits	Yes	Work
Country of Reference	Yes	Work
Original Format	Yes	Item
Original Length	**No**	Work, Variant, Item
Original Duration	**No**	Work, Variant, Item
Original Language	**No**	Work
Year of Reference	Yes	Work
Identifier	Yes	Work
Relationship	Yes	Work, Variant, Item
Source	Yes	Work, Variant, Item
Genre	**No**	Work

Example: CSV layout suggestion

Here we provide an example of a CSV column layout that should reduce the preprocessing, deduplification and other guesswork during ongoing daily imports. It must be noted here however, that due to the nature of CSV, having multiple files with different column-layouts would be necessary. If a single CSV file is desired to be used for the import, certain data improvements would lead to a significant increase of columns in order to reduce the complexity of finding and assigning related entities. One example would be support more than one agent per type (writer, producer, cast, …) per entry (=per line). If and how to address pros/cons of other ambiguities, like having agent names as a single text field versus individual fields (firstname, lastname, …) depends greatly on how important this case is considered, as it would greatly expand the size and complexity of a CSV. If this level of exactness is desired, an XML would be better – but may very likely increase the efforts on the data export side.

In this first CSV example, agent names are expected as single string, similar to the current Treasures data. The syntax in which the names are expected should be declared (and enforced) as strictly as possible. See the comment on agent names in “Source Schema Layout” for additional considerations.

Entries marked with “[CVoc]” indicate that this field must use terms from a controlled vocabulary. This example is intentionally hardcoding two external identifiers to avoid making this example too generic (e.g. avoid any external schema). It may be encouraged to offer more than one external identifier to avoid dependency on a single external entity, and to offer at least one publicly and freely available option. EIDR and Wikidata were chosen since they are already commonly used for linking filmographic entities in practice. Of course any other external identifier can be chosen by FIAF. “Full Film Title” was omitted on purpose, since it is to be generated internally by the database system or the importer to avoid parallel implementations of the same concatenation syntax.

Columns marked bold are considered mandatory fields. This is merely a suggestion. Which fields should actually be considered mandatory might better be reviewed by FIAF.

Column	**Comment**
Local ID	An identifier that must at least be unique in the context of the source institution providing it.

Column	Comment
Ext.ID1	For example: EIDR[1]
Ext.ID2	For example: Wikidata[2]
Film Title	Name of the film (work)
Alternative titles 1..n	N columns for alternative film titles
Film Country	Code for “country of reference” (production?) - [CVoc as defined in EN15907 – e.g. ISO 3166]
Film Year	To be declared: 1) Which year is meant: production? release? other? 2) Exact syntax for date ranges (1890-1927) and fuzzy dates like “189-” or “1927?”
Series title	Title of series
Series ID	Identifer used to find the right series (instead of by name). External identifier preferred.
Duration	This was added, due to being listed by EN15744. To be discussed: physical length of film item or playback duration or both?
Language	ISO 639. To be declared: 1) which subset: 639-2, 639-3, … ? 2) which language usage defined as default?
Cast 1..n	N columns for cast names
Cast 1..n ID	Identifier for each agent. External identifier preferred.
Writer 1..n	N columns for writer names
Writer 1..n ID	Identifier for each agent. External identifier preferred.
Photographer 1..n	N columns for photographer names
Photographer 1..n ID	Identifier for each agent. External identifier preferred.
Credit 1..n	N columns for cast names
Credit 1..n ID	Identifier for each agent. External identifier preferred.
Producer 1..n	N columns for producer names
Writer 1..n ID	Identifier for each agent. External identifier preferred.
Director 1..n	N columns for director names
Director 1..n ID	Identifier for each agent. External identifier preferred.
Production Company	Name of the production company
Production Company ID	Identifier for the agent. External identifier preferred.
Archive Name	Name of the holding institution / record source
Archive City	Name of the city where the archive is located [CVoc]
Archive ID	Currently, a so called “IMIS code” (4-letter code) is used. [CVoc] External identifer preferred.
Access Copy Gauge	[CVoc] Examples: 35mm, 16mm, 1/2” video, etc.
Access Copy InstantiationType	[CVoc] Examples: acetate positive, unspecified, DVD, etc.

1 "EIDR, or the Entertainment Identifier Registry, is a global unique identifier system for a broad array of audio visual objects, including motion pictures, television, and radio programs” (Quote: Wikipedia)

2 “Wikidata is a free and open knowledge base that can be read and edited by both humans and machines. Wikidata acts as central storage for the structured data […]” (Quote: Wikidata.org)

Column	Comment
Access Copy ID	Identifier of the item.
Non-Access Copy Gauge	[CVoc] Same as for Access Copy fields.
Non-Access Copy InstantiationType	
Non-Access Copy ID	Identifier of the item.
End user note	Freetext.

Handling Identifiers

In the CSV example, we've added identifier fields not only for the work, but also for all agents (director, producer, writer, …) as well as the actual items (access, non-access). The handling of identifiers as described here applies regardless of the exchange format used (CSV, XML, etc).

Only "Local ID" is defined mandatory, as every archive should have some identifier on this level – even if only locally unique, whereas any additional or external identifiers may not be present (yet) at the source.

Since the record source is known during import (or at least by "Archive ID"), it is possible to keep these local identifiers stored in parallel the created datasets (similar to adding additional identifiers in EN15907). This can be used to avoid deduplification guesswork upon subsequent imports – at least from the same institution, even allowing update existing data. For example, by providing the local identifier of the filmographic work, additional agents can be added or augmented at a later import, without even requiring to provide any information about the film (except for any identifier previously assigned).

This layout shall also encourage source archives to assign external identifiers (e.g. EIDR, Wikidata, etc) to their data sets, therefore allowing them to uniquely identify their data entries submitted not only to FIAF Treasures, but also when exchanging with others.

Of course the number of columns of this CSV is noticably greater than the ca. 20 fields of the previous export XML, but this is necessary to allow exact data assignment by reducing guesswork and concatenated fields.

External identifiers

For the fields "Ext.ID" (1 and 2) it is clear that they are referenced externally, but it has to be declared which sources are used by the import (e.g. EIDR, Wikidata, etc). It is advisable to keep the number of supported external identifiers to a minimum to avoid overcomplication and additional errors. For all other entities (e.g. agents, series, …) where external identifiers may be used, it has to be declared and defined in the importer code (possibly different per source archive) the schema/context in which these identifiers are declared/unique.

Software Requirements

The following is to be considered:

- The optimal software architecture
- Possible Candidates
- Check which valid candidates meet required functional and technical requirements
- Technical background of the candidates, installation/maintenance effort and 3rd party hosting options
- The estimated costs for data preparation, migration, eventual necessary customization, testing and quality assurance.

Software architecture

By its core, the archive sector should stay as independent as possible to avoid issues like “Vendor lock-in” or necessary hard- or software being declared “End-of-Life”.

One of the requirements is to avoid replacing proprietary Filemaker with another proprietary solution. Since properly managed and funded Open Source solutions have become an increasingly successful option in the archival domain (Archivematica, MediaArea, etc), an Open Source solution is the recommendation. To stay client-side as platform independent as possible, it is also recommended to use a system with a web frontend that works on a broad spectrum of browsers on different operating systems. Browser based catalogue systems have become one of the most common, if not the leading, option in recent years.

Candidates

Currently, there are some available solutions that meet many of the requirements as specified in the next sections. A comparison was done and the information aggregated in a table, checking which features are available as required and what functionality is missing or would have to be developed. Compatibility to, or the possibility to, implement EN15907 will also be held into account when analysing possible solutions for a long term archive system. Software that is not based on open-source is disqualified from the analysis to prevent issues that are typical with proprietary software, as described above.

AtoM / AtoM 2

“Access to Memory” (better known as AtoM) is an Open Source application for archival description and access. It utilizes well supported technologies as Nginx/Apache, MySQL, PHP and Elasticsearch. It also comes with some preconfigured standard templates like Dublin Core, ISAD(G), RAD, PREMIS and others. AtoM will be looked at in more details as a candidate.

Avalon Media System

The AMS Project is an Open Source application with a heavy focus on digital audio and video, including access management. While it does have capabilities for metadata enrichment, the main purpose is ingest and archival description of digital files with less focus on archive metadata and the needed detail. As the initial effort to adapt the system for archive use might be way above any meaningful level, it has been deemed unfit for the use case at hand and is not on the short list of candidates.

CollectiveAccess

Also Open Source, CA is specialized for managing archival collections. The speciality is the highly detailed description of archive material, following a configured standard. There are some standards already available as default templates. While the backend engine “Providence” is high in complexity, this also allows it to be configured in detail as required by the complexity of the matter at hand. There also is a first implementation

of EN15907 available and could be extended to specific needs if required. This candidate will be checked in more detail. CA already provides a separate web-frontend system called “Pawtucket”.

CollectionSpace

While primarily being used to manage museum collections, CS allows for cataloguing items using defined templates. While it might be possible to implement EN15907, it would require quite a development effort. CS is built on Nuxeo, a content management system, and based on Apache Tomcat, thus it relies heavily on Java and Javascript. While Tomcat is still actively developed, it requires more effort in installation and maintenance and is not the first choice for modern applications. Due to those reasons, CS is not a candidate for further analysis.

DSpace

Also written in Java, DSpace is a full stack web application with a focus on digital asset management and content delivery to end users. By default it comes with the Qualified Dublin Core metadata schema and allows for custom QDC-like schemas. With the next major release, some modern Web UI is also planned. It satisfies some of the key requirements, so it will be looked at in more detail to analyse the heavy focus on managing digital files, dissemination and the potentially rather high effort to implement the Filemaker Scheme / EN15907 to use for FIAF Treasures purposes.

Omeka / Omeka S

While primarily a web publishing platform for online exhibits, Omeka is possibly a light-weight option for handling archive collection metadata. Omeka also enables multi-site access to a larger set of stored data to be used by different projects.

It is, using the default installation, strongly configured to be used with Dublin Core metadata. Overall, the system is based a lot on the use of the right plug-ins for whatever purpose is desired, and an active community might have already developed those that are needed.

Further, it also offers various ways for data import and export and can be connected to other systems, e.g. DSpace, for data exchange. The overall flexibility makes it a candidate as a framework that might need – relatively - low effort in some plug-in development and customization – partly in code, partly using the administrative GUI – to meet the requirements for FIAF Treasures.

ResourceSpace

A feature-rich application with many options for configuration and tuning. It is focused on digital asset management and also feature a workflow engine and numerous options to interface with other systems. For the FIAF Treasures use-case, ResourceSpace might be vastly over-complex in initial setup and handling. It lines up with the other candidates to have taken a look at.

Samvera / Hyrax

A strong framework, primarily focused on digital content and with a focus on educational environments and their use cases. Hyrax is a frontend to extend the Samvera-backend. This software is optimized as repository platform for projects that are in cooperation between several institutions, e.g. research projects with several universities involved. For this use-case, it appears too complex and the initial hurdle to adapt it for the given requirements might be disproportionally high. That in combination with the overall different core focus disqualifies it as a candidate to inspect further.

Self-developed Solution

This approach has a very strong pro argument. By nature, it would be tailored exactly to the needs of a project and adding more features as they show to be practical in daily use, should be a lot faster compared to an externally developed software. However, the downsides of this approach – and a reason why framework solutions are popular for a long time already – are numerous.

The costs are often very high in comparison to other options and require at least one programmer to work on the software for a substantial amount of time, several months at least. The tool will have to be documented, nursed and patched as bugs show up and features are added, and might be stuck in a specific eco-system (e.g. JAVA, php, …) as favoured by the developer. While the latter is also true for third party software, the difference is in the point in time when the decision to use a language has to be made. This can be based on software in a high development stage or full fledged product, or has to be decided very early in an internal process, containing the risk to find out about certain restrictions or limitations later in the development, that can cause possible dead-ends or resource-heavy re-works.

Having many Open Source options in varying quality available, a self-developed solution might be the approach with the most resources required and an example of “re-inventing the wheel” where not necessary. Using the same resources to adapt an existing open-standards supporting solution would, with a high probability, serve all involved parties best and give the chosen software a better chance of surviving and flourishing in the wild, instead of being a singular solution that blooms once and then dies off quickly.

Functional Requirements

These are defined by the following overview, which gives an insight on necessary functions that the platform must be able to carry out:

User Roles	User management to configure user rights at different levels (User, Archive Contributor, Editor, Administrator,…). For non-admin users, only creating and augmenting entries should be permitted, but no delete / overwrite.
Search	Ability to search for: Film Title, Year of Release, Country, Director, Production Company, Producer, Cast, Archive
Advanced Search	Search combinations and/or ranges of existing entries
Upload Metadata	Enrich database by uploading a file (CSV/TSV, XML, JSON). Automatic mapping of data to fields and normalization of certain fields to a standard (e.g. ISO dates). Should include failure and error reporting.
Export Metadata	Extract a dataset from the database and save it into an exchange format, e.g. CSV/TSV, XML, JSON. Ideally, this should work for “all records modified in date range”, “all items per archive” and other parameters that could be specified.
Direct Data Enrichment	The manual entry of datasets must be possible, in case there is no information available to automatically import.
Available GUI	A GUI must be available to administer users and archives, delete/change/create/edit/[…] records, maintain lists and vocabularies, supervision of exports

Technical Requirements

Must-have

Norm fields in back-end, human readable on front-end	The Platform must be able to store a technical norm-value (e.g. ISO value as defined in the metadata standard), while showing human readable labels to the user
Hierarchical vocabulary	• Support hierarchical layers in controlled vocabulary sets.

	• Definable which nodes in the tree are selectable. • Definable which nodes in the tree are enabled/visible.
Support separate cataloguing sheets	Ability to configure different types of archive items with their own set of parameters
Data widgets	Different UI widgets for entering and presenting data that allow handling more complex data types, or data in better ways.
Performance	The system should be able to handle searches in a decent time frame and also allow entry/editing of new items with possibly dozens of entities added without long delays.

Optional

Support separate/parallel metadata schemas	For different object types (Cinematography vs Documents or Photos)
GUI Editor for vocabulary	Adding new vocabularies, editing, deleting, etc.
GUI Editor for cataloguing sheets	Configurable user interfaces to edit / view datasets
Separate cataloguing sheet UIs for input and view/listing	Available user interfaces for different purposes (e.g. edit data, view data, search data, ...)
Multi-language	• Vocabulary lists • Field values (even free text) • User Interface
Changelog	Automatically document if, when and by whom the content of a data field was changed.

Comparison Matrix

	Omeka (S)	**CollectiveAccess**	**ResourceSpace**	**Atom**	**DSpace**
EN15907 available or implementation possible	No/Possible	Yes	No/Possible	No/Hardly possible	No/Possible
User Administration	Yes	Yes	Yes	Yes	Yes
Different User Roles	Yes	Yes	Yes	Yes	Yes
Multi-Parameter Search	Yes	Yes	Yes	Yes	Yes
Advanced Search	Yes	Yes	Yes	Yes	Yes
Import Metadata	Possible [1]	Possible [1]	Possible [1]	Possible [1]	Possible [1]
Export Metadata	Yes	Yes	Yes	Yes	Yes
REST API	Yes	Yes [2]	Yes	Yes	Yes
Managing GUI	Yes, Web	Yes, Web	Yes, Web	Yes, Web	Yes, Web
Editing GUI	Yes, Web	Yes, Web	Yes, Web	Yes, Web	Yes, Web
Norm fields in back-end, human readable on front-end	Yes	Yes	Yes	Yes	Yes
Nested vocabulary	Requires plug-in development	Yes	Yes	Limited	Yes
Support separate cataloguing sheets	Requires plug-in	Yes	Yes	Limited	Yes

	Omeka (S)	CollectiveAccess	ResourceSpace	Atom	DSpace
	development				
Data widgets	Yes	Yes	Yes	Yes	Yes
Performance	Scalable	Scalable	Scalable	Scalable	Scalable
Support separate/parallel metadata schemas	Yes	Yes	Yes	No	Yes
GUI Editor for vocabulary	Yes	Yes	Yes	Yes	Yes
GUI Editor for cataloguing sheets	Yes	Yes	Yes	Limited	Yes
Separate cataloguing sheet UIs for input and view/listing	Requires plug-in development	Yes	Yes	Limited	Yes
Multi-language	Plugin available	Yes	Yes	Yes	Yes
Changelog	Requires plug-in development	Yes	Yes	Limited	Yes
Installation complexity	Medium	Medium	Low	Medium	Medium
Maintenance complexity	Low	Low	Medium	Medium	Medium
Third Party hosting available	Yes	Yes	Yes	Yes	Yes

[1]: Requires development for the rather complex use-case of linked-entity imports, e.g. if a Cinematographic Work is imported, it has to be checked if linked entities are already existing in the target database, and create them if not. Please see the remarks about direct data access vs API in “Data Access: XML vs SQL”.

[2]: Since CollectiveAccess’ Pawtucket part can use the data-backend database directly as-is, it is most likely able to cover the required features for public access without the need to give REST API access to a different web platform, as there is no programming required.

AtoM

While being a very good platform for already configured Templates and overall very customizable, those customizations have often to be done directly in the code, which requires Developer work and causes high future maintenance to keep such customizations working with platform updates.

It currently lacks an EN15907 implementation, which would require substantial work in configuring the system for all required entities, relations and controlled vocabularies, partially on code level.

Additionally, for the reason that it does not offer the required flexibility in configuring many necessary parts as requested, AtoM does not satisfy enough of the requirements and is thus not suggested as the platform of choice.

CollectiveAccess

A very strong and highly configurable framework with an almost complete implementation of EN15907 available. It consists of 2 individual systems: the “Providence” core, which is the back-end used to manage and edit metadata and other information and optionally “Pawtucket”: a public web-access front-end for users that is designed to work directly with the data entered in Providence as-is.

While the Providence core has a very high complexity, this also enables it to be configured for many use-cases. The GUIs for different descriptive levels (Work, Item,…) and entities (Agents, ...) can easily be modified to focus on required fields as seen fit by the archive. While not free of edges, it can become the top-tier choice for film archives with some development effort. It has the lowest initial effort as the standard is already implemented for the biggest part and only the enrichment of some controlled vocabulary fields and lists is required to start using it.

When not going with any of the already implemented standards, the effort to implement a custom tailored metadata set compares to other solutions.

Omeka

A slim system relying on plug-ins, specialized in providing digital data for public online exhibits. While it does have the basic functionality of a full cataloguing system, the effort to modify it to meet all requirements in a sufficient way may be unnecessary high. The default GUI is not ideal for film archives as it is focused to present the Item and shows other information as secondary in a side-bar. The default installation is also very optimized on the Dublin Core data fields.

Overall, with substantial investment into development of necessary plug-ins, Omeka could be used as a framework for the FIAF Treasures content, if no better candidate is found.

ResourceSpace

While being a strong platform with a lot of features, ResourceSpace is focused on digital asset management and less on metadata and use in a film-archive or professional preservation scenario. It comes with a lot of tools not needed for the use-case at hand, some of them pretty advanced, e.g. using AI for automatic tagging of objects or entities in pictures, and an advanced workflow engine for digital asset manipulation.

While being a nice DAM, it would probably be limited to staying a superficial, consumer use-case oriented system. Any experiences or efforts put into it would less likely be reusable outside of its context.

Example: A Treasures dataset depicted in EN15907 in CollectiveAccess

The following XML export example will be used to explain the data migration of this Cinematographic work into CollectiveAccess with a manual example. Please note that not all lists are populated fully yet, so some fields in the example might be incorrect.

Following here is the raw XML data for one dataset (called “row” in the export):

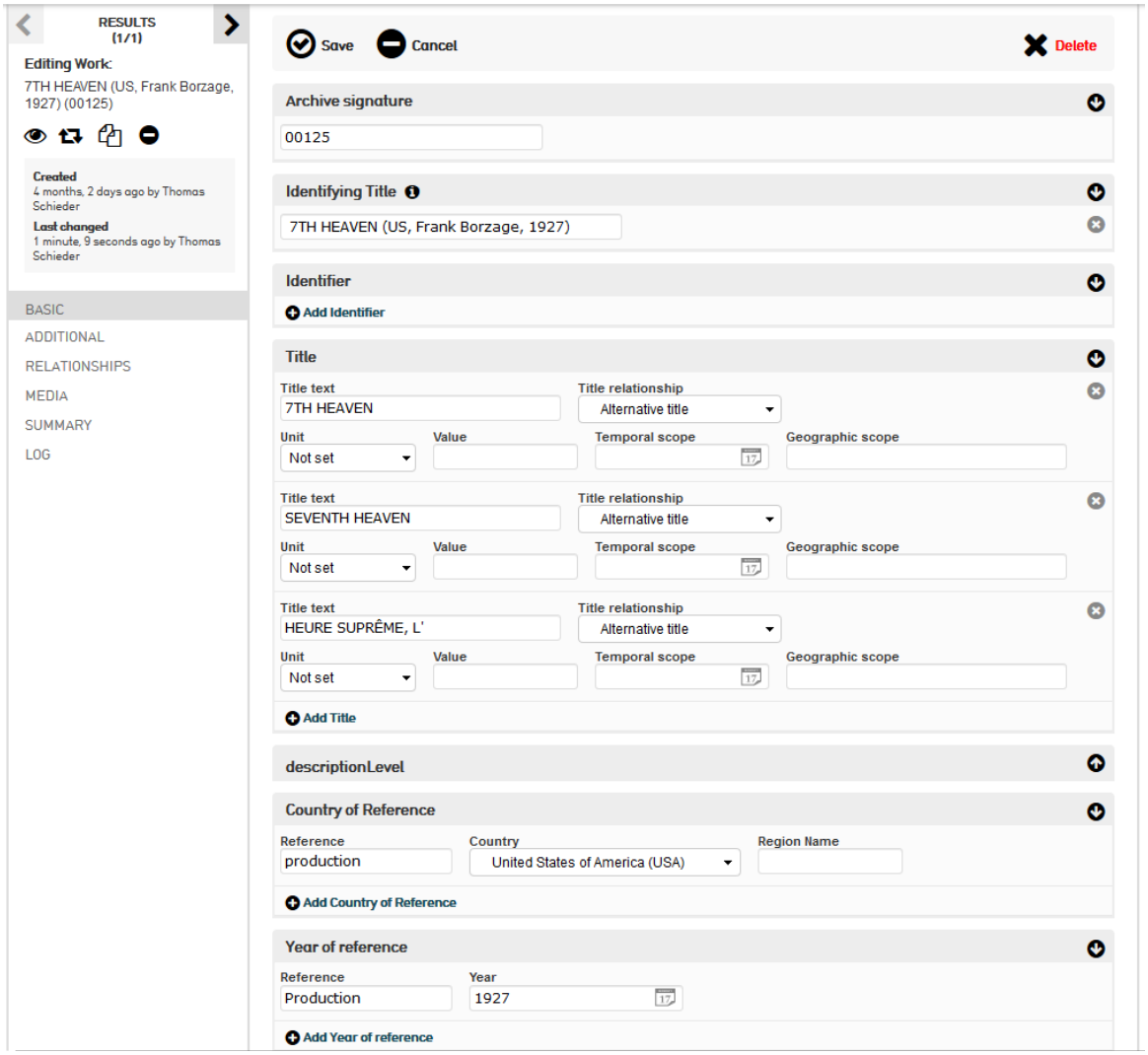
```
<ROW MODID="21" RECORDID="123">
  <AN> 125 </AN>
  <FI> 7TH HEAVEN (US, Frank Borzage, 1927) </FI>
  <FT> 7TH HEAVEN§SEVENTH HEAVEN§HEURE SUPRÊME, L&apos; </FT>
  <FC> US </FC>
  <FD> Borzage, Frank </FD>
  <FY> 1927 </FY>
  <SE> </SE>
  <PC> Fox Film Corp. </PC>
  <FP> Borzage, Frank </FP>
  <CA>
    Gaynor, Janet|Farrell, Charles|Bard, Ben|Butler, David|Mosquini, Marie|Gran,
Albert|Brockwell, Gladys|Chautard, Emile|Stone, George E. (Georgie)|Haslett, Jessie|
Hurst, Brandon|West, Lillian
  </CA>
  <FW> Glazer, Benjamin F. </FW>
  <PH> Palmer, Ernest|Valentine, J. A. </PH>
  <CR> Hilliker, Katharine|Caldwell, H. H.|Wolf, Barney </CR>
  <AR>
    Cinémathèque Royale de Belgique (Brussels) [BEBR]|Museum of Modern Art -
Department of Film (New York) [USNM]|UCLA Film &amp;amp; Television Archive (Los Angeles)
[USLU]|Academy Film Archive (Los Angeles) [USLA]|George Eastman Museum (Rochester)
[USRG]|Cinémathèque Française / Musée du Cinéma (Paris) [FRPF]|Library of Congress
(Washington) [USWL]|Lobster Films (Paris) [FRPL]
  </AR>
  <AH> 35 mm: USNM|16 mm: USWL </AH>
  <NH>
    35 mm nitrate positive: USLU|16 mm: USLA|Format unspecified: USRG, FRPF|35 mm
acetate positive: USWL
  </NH>
  <NT>
    USWL: 16 mm is Killiam reissue version{Inclusion of a title in this database does
not guarantee its availability nor completeness. Users should contact individual
archives for more information.
  </NT>
  <NF> </NF>
</ROW>
```

A table-representation of the same data:

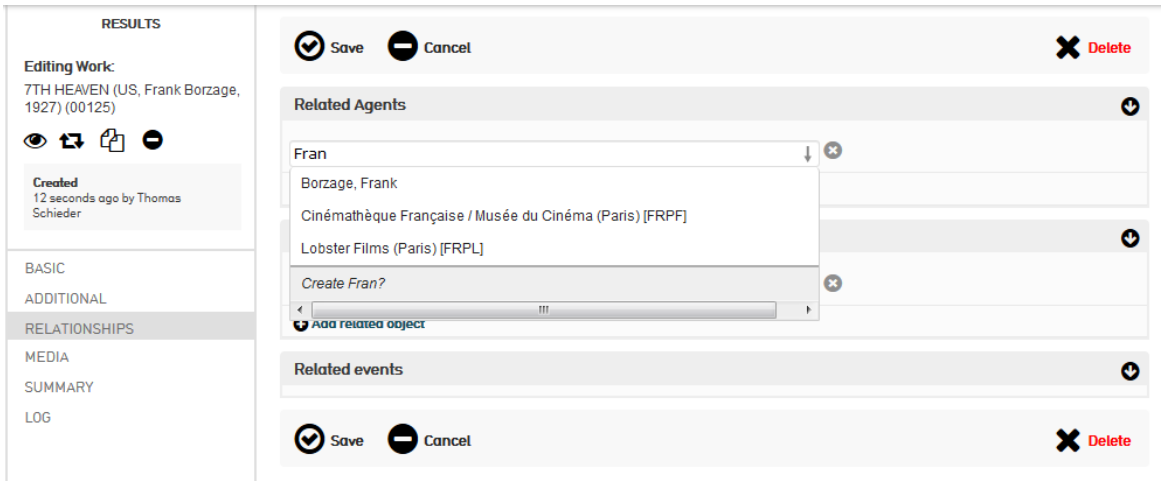
Identifier	Description	Data
AN	Accession number	123
FI	Full Film Title	“7TH HEAVEN (US, Frank Borzage, 1927)”
FT	Film Title	“7TH HEAVEN” “SEVENTH HEAVEN” “HEURE SUPRÊME, L”
FC	Film Country	“US”
FD	Film Director	“Borzage, Frank”
FY	Film Year	1927
SE	Series	NULL (=empty)

Identifier	Description	Data
PC	Production Company	“Fox Film Corp.”
FP	Producer	“Borzage, Frank”
CA	Cast	“Gaynor, Janet” “Farrell, Charles” “Bard, Ben” “Butler, David” “Mosquini, Marie” “Gran, Albert” “Brockwell, Gladys” “Chautard, Emile” “Stone, George E. (Georgie)” “Haslett, Jessie” “Hurst, Brandon” “West, Lillian”
FW	Writer	“Glazer, Benjamin F.”
PH	Photography	“Palmer, Ernest” “Valentine, J. A.”
CR	Credits	“Hilliker, Katharine” “Caldwell, H. H.” “Wolf, Barney”
AR	Archive	“Cinémathèque Royale de Belgique (Brussels) [BEBR]” “Museum of Modern Art - Department of Film (New York) [USNM]” “UCLA Film & Television Archive (Los Angeles) [USLU]” “Academy Film Archive (Los Angeles) [USLA]” “George Eastman Museum (Rochester) [USRG]” “Cinémathèque Française / Musée du Cinéma (Paris) [FRPF]” “Library of Congress (Washington) [USWL]” “Lobster Films (Paris) [FRPL]”
AH	Access	“35 mm: USNM” “16 mm: USWL”
NH	Non-Access	“35 mm nitrate positive: USLU” “16 mm: USLA “Format unspecified: USRG, FRPF” “35 mm acetate positive: USWL”
NT	Note	“USWL: 16 mm is Killiam reissue version” “Inclusion of a title in this database does not guarantee its availability nor completeness. Users should contact individual archives for more information.”
NF	NFPF	NULL (=empty)

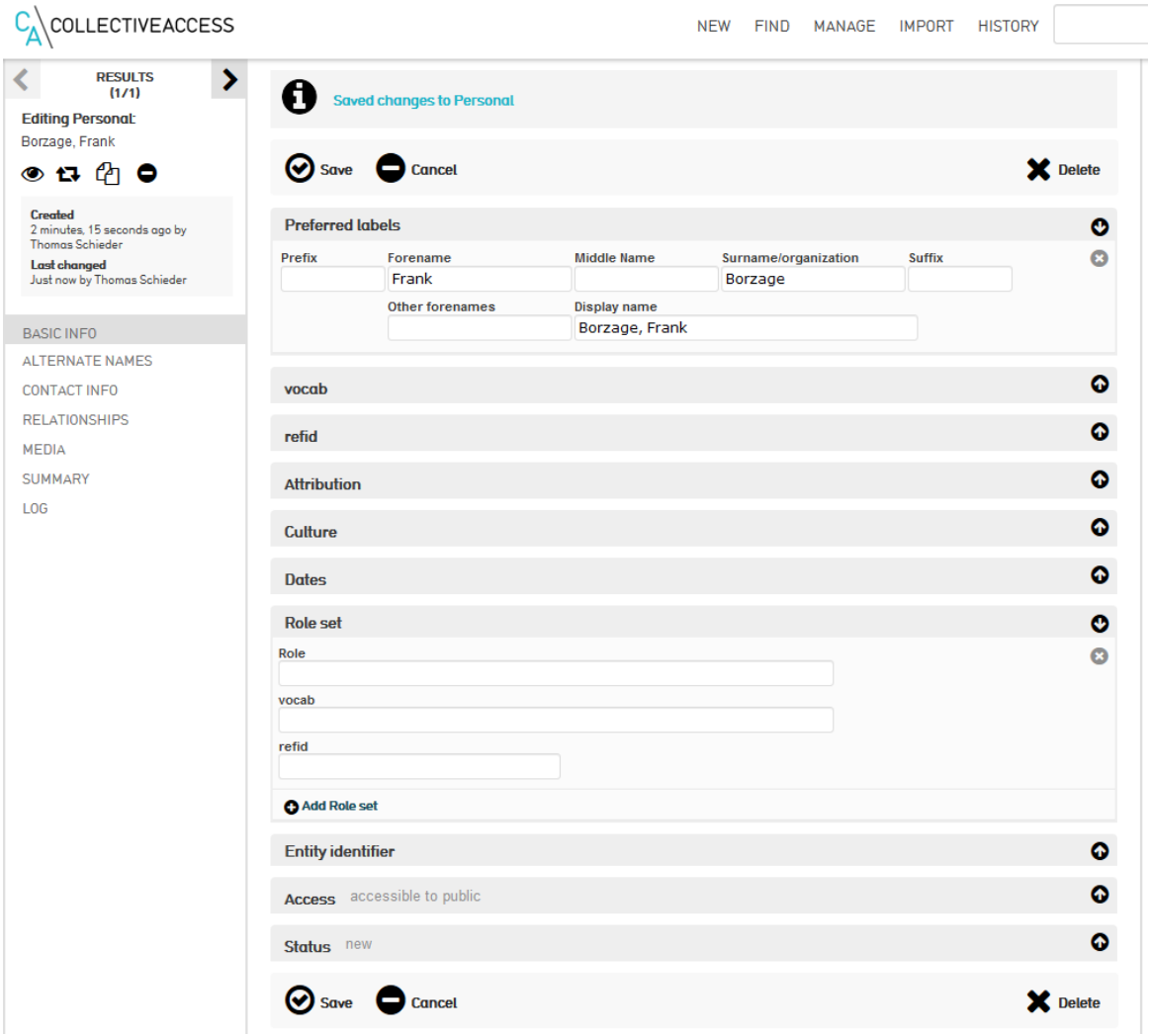
Creating a Work initially enables the user to add the information of the fields AN, FI, FT, FY and FC, forming the base for the new Work (or Item, …). The GUI supports the adding multiple entries for some fields, like the alternative titles and country/year of reference for key events in the Work's life.



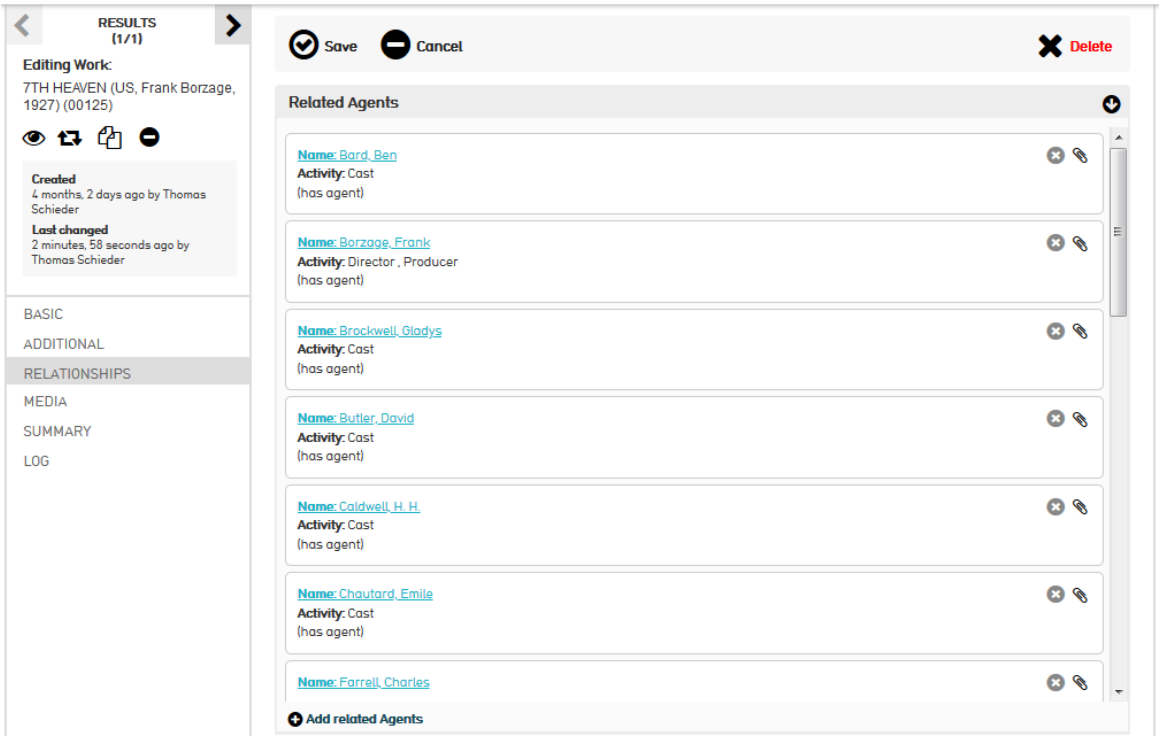
The relationships tab allows to enter Agent entities, which covers the fields FT, FP, PC, CA, FW, PH, CR and AR:



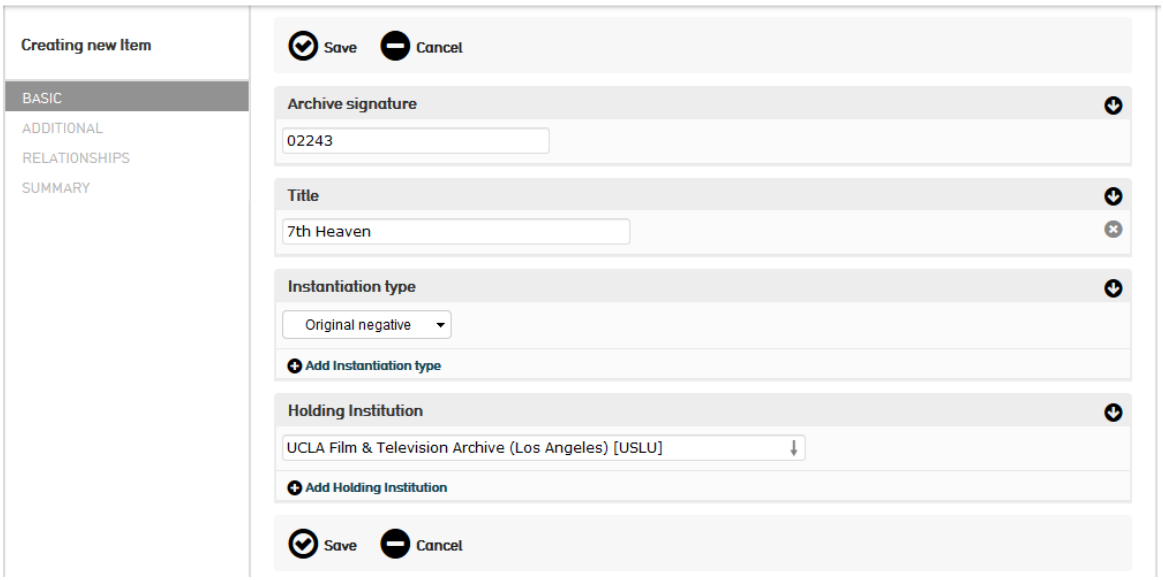
In case an Agent entity is manually entered but not found yet in the database (while typing, known entities that match will show), it can be created using the “Create <entered string>?” function. The example below shows the creation of an Agent of the type Person. The same GUI is used for Agents of type Organization/Corporation.



After the agent entities are added, their roles/relationship can be set at this point or alter in the workflow. The result is a populated list for relationships, as can be seen below:



Items represent physical manifestations or digital files of content in the catalogue and have typically data on the holding institution for the object. Creating an Item also supports pre-search for entities and quick-creation if they do not exist in the database yet, as can be seen on the following screenshot:

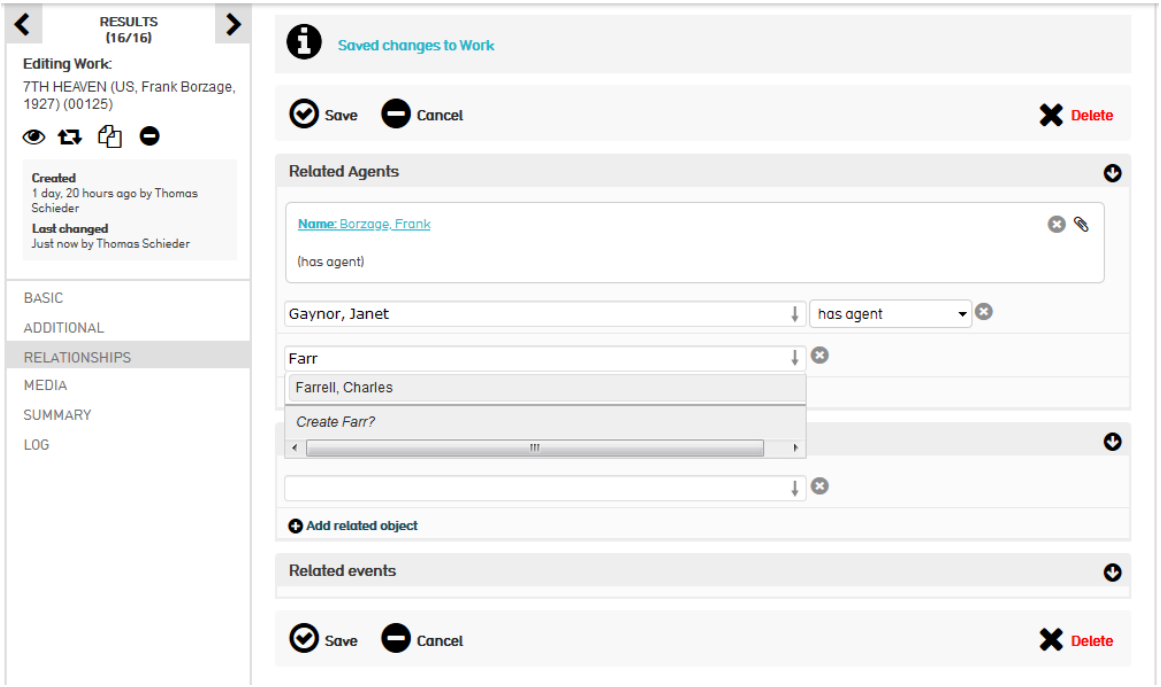


The Tab “Additional” can be enriched with information on the detailed type of the carrier Item, e.g. “35 mm nitrate positive”. The Relationships section will show links to the Cinematographic Work and Agent entities that are specifically related to this Item.

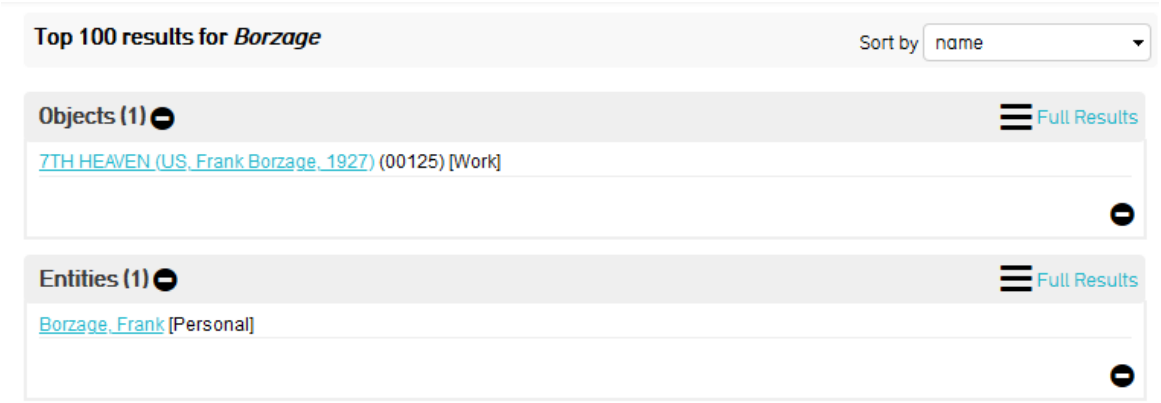
Other entities are created likewise with very similar GUIs containing the required and optional fields. It is possible to optimize the GUIs and disable unused fields for less cluttering and a better overview of wanted core fields.

The engine allows for dynamic linking of objects and entities as configured, using the “Relationship” tab. This is used to add Corporate and Person Entities: Directors, Cast, Producer, Production Company. The

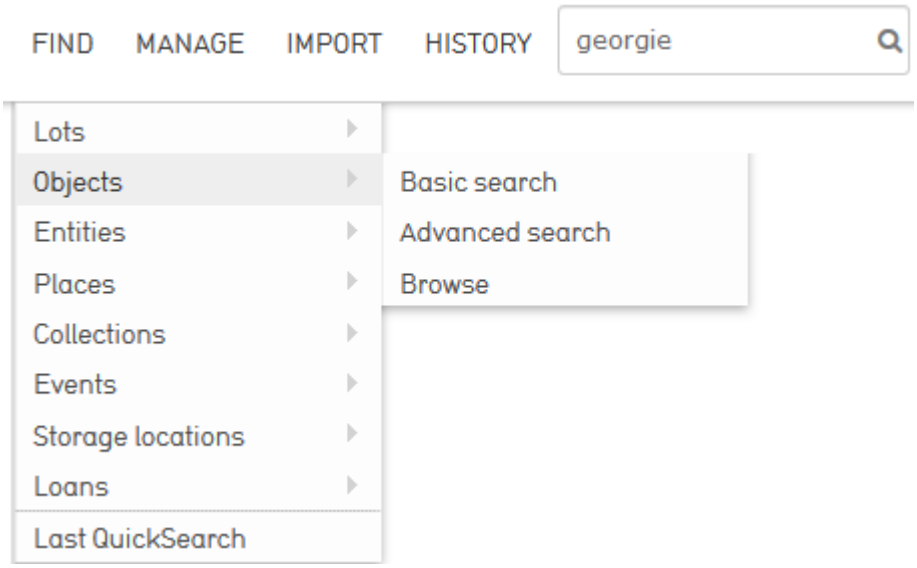
relationship types can be very freely configured bi-directional, so depending on the current active GUI, it might show “[Agent] is Director of [Work]” or “[Work] was directed by [Agent]”. Please note that in the following example screenshots, no relationship types are configured, so they only show “is agent of” relationships.



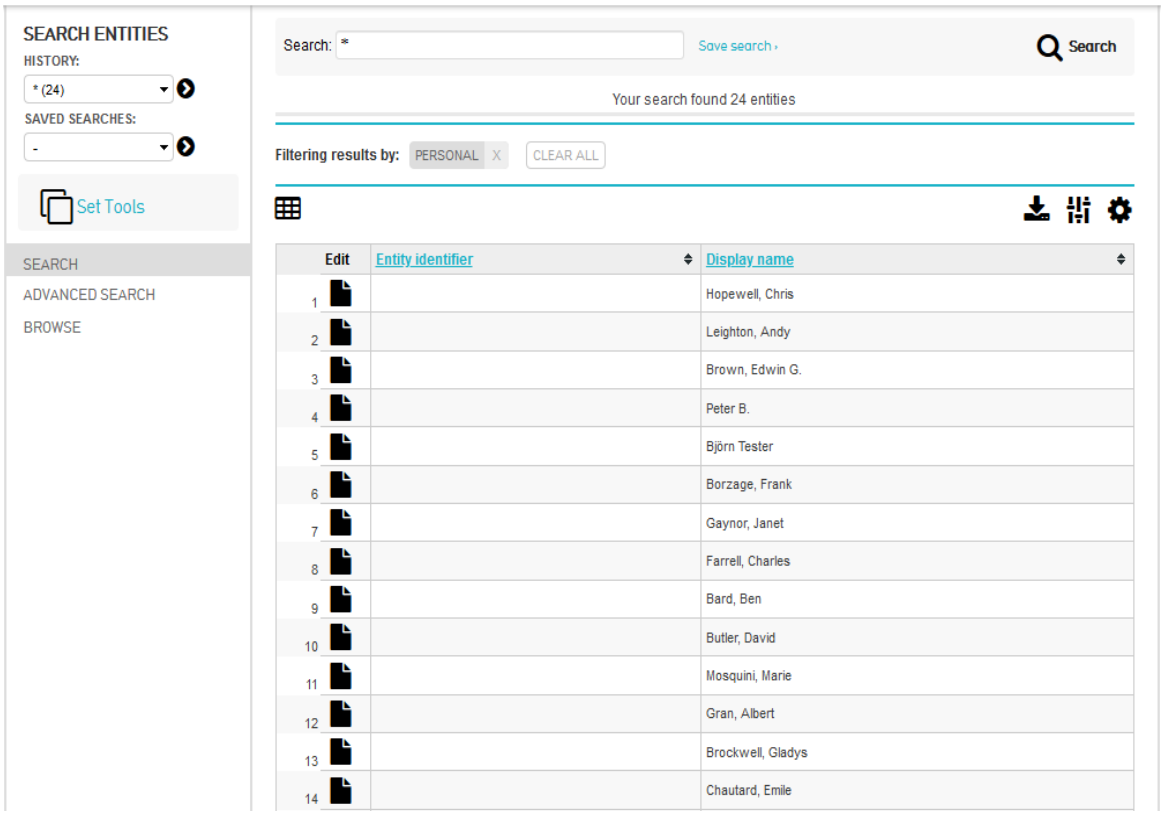
There are different methods to search the database content. The basic search will show all hits sorted by entity type, here an example that shows Frank Borzage as a Personal entity and a work having his name in the main identifier:



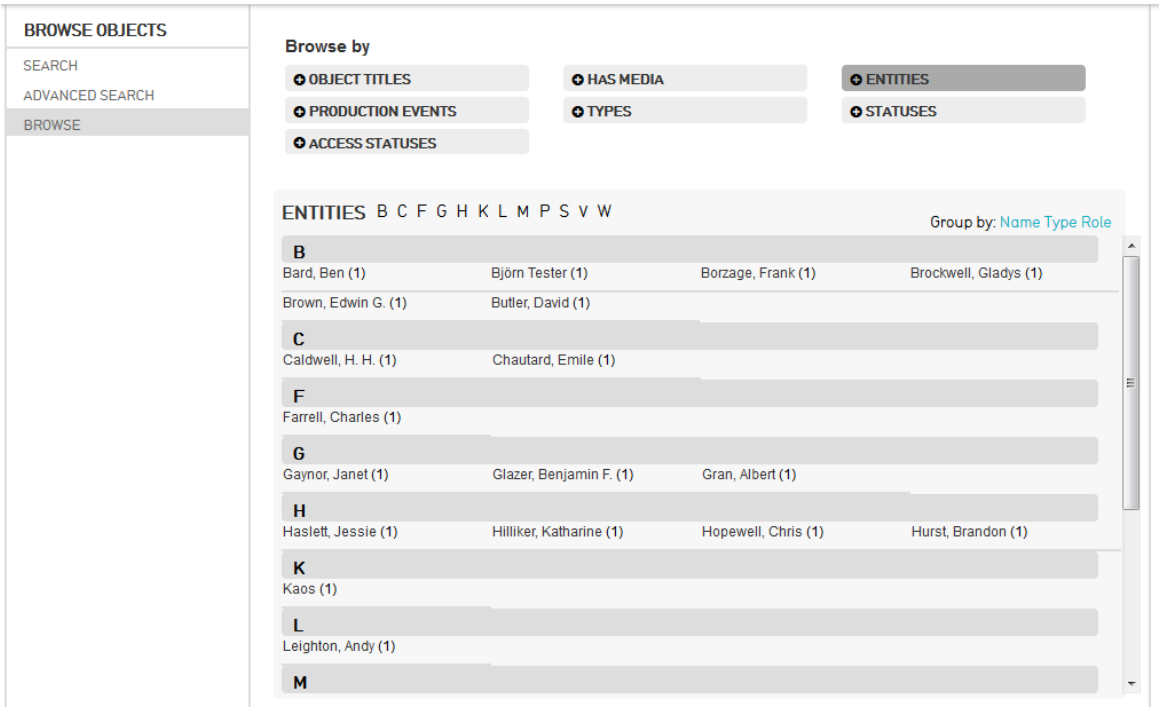
There is also the option for simple and advanced searching over all available fields that are configured in the extended search GUI.



This also allows to create complex searches and store them for later use:



Another method available by default is browsing by entity type, status, events and other options, which will present an alphabetically sorted list of available hits. This is useful if a name is known phonetically, but the exact spelling is unsure.


BROWSE OBJECTS
SEARCH
ADVANCED SEARCH
BROWSE
Browse by
OBJECT TITLES
HAS MEDIA
ENTITIES
PRODUCTION EVENTS
TYPES
STATUSES
ACCESS STATUSES
ENTITIES B C F G H K L M P S V W
Group by: Name Type Role
B
Bard, Ben (1)
Björn Tester (1)
Borzage, Frank (1)
Brockwell, Gladys (1)
Brown, Edwin G. (1)
Butler, David (1)
C
Caldwell, H. H. (1)
Chautard, Emile (1)
F
Farrell, Charles (1)
G
Gaynor, Janet (1)
Glazer, Benjamin F. (1)
Gran, Albert (1)
H
Haslett, Jessie (1)
Hilliker, Katharine (1)
Hopewell, Chris (1)
Hurst, Brandon (1)
K
Kaos (1)
L
Leighton, Andy (1)
M

Conclusion

The analysis shows that there is no golden solution and digging deeper into the existing structure revealed some additional issues that need to be addressed. The approach to sanitize the metadata and import it to a new platform using a new – preferable standardized – metadata scheme, all in one step, does not seem to be a realistic approach. The “jump width” is simply too far, so the process is ideally split into several steps instead:

- Milestone 1: Improve existing data quality
- Milestone 2: Define target metadata schema
  - Option A: Moving towards EN15907.
  - Option B: Moving towards the concept of Work / Item / Agent, but based on legacy Treasures field content.
  - Option C: Keep data layout and reproduce as-is.
- Milestone 3: Choose target system (backend).
  Includes defining import/export capabilities.
- Milestone 4: Attach web-based frontend
  - Option A: Keep using existing frontend with current XML export as-is.
  - Option B: Use the new web-based backend as frontend.
  - Option C: Choose and implement a new frontend.

Milestone 1: Improve existing data quality

When migrating the data from the Filemaker database to any new platform, any descriptive errors in it would also migrate. Thus, sanitizing the metadata by script support would be a good start to prevent this from happening.

Assessment: Which fields and manual or automated fix?

This step can be done in any mix: from completely manual to fully automated (scripted), solely depending on available staff/knowhow resources and time. It makes sense to:

- Cataloguers: Get input from cataloguers about which terms should and can be improved and make a list.
- Developer: Make a rough assessment which cases to expect and which make sense to fix in an automated way (=sufficiently large number of cases to detect and fix) and which cases to address manually (=number too small or difficult to detect for a machine).

Depending on the outcome of this assessment, you may need more cataloguers (for manually editing/fixing the data) – or programmers (for automated fix).

Deduplifying entities

The major issue is that once entries are imported into a system as entities, the theoretically same entity might show up several times with slight differences. When then linking those entities, for example a Director, to an Item or several Items, the user will have a hard time or even find it impossible to use the correct entity, if there is entries for “Richard Anderson”, “R. Andersson”, “Richard K. Anderson” and “Richard ‘Richi’ Anderson”, which are the same entity. While some sorting and fuzzy searches can look for similarities and provide a list, some decisions may still require checking by a human to solve de-duplification issues. The same is true for all entity types that might suffer from this issue. Due to the nature of batch-importing

metadata to a new target system, the linking of entities is something that must happen automatically, as manual linking is simply too time consuming and error-prone. For this to succeed, existing entities should be as unique as possible to be correctly linked on creation in the new target system.

Normalize existing terms

Additionally, data sanitizing is also required for fields that follow – or could follow – specific controlled vocabularies or standards. It is to be checked – again, with technical support using scripts etc – if all those fields do contain the data type they are supposed to hold, and especially if the format is exactly what it is supposed to be.

It would be okay to simply normalize the terms based on the existing contents, leaving the task of choosing and defining external vocabulary sources and terms postponed to Milestone 2 to quicker facilitate an initial migration.

Split people's names

Another discovery while analysing the data structure was the concatenation of data fields, especially names, which are stored as one-line-strings. If possible, it would be suggested to break this up into single fields, e.g. "First Name", "Last Name", "Middle Name" and so on, instead of "Louis C. Clark", for example. This allows for a clean handling of this data in the future and can be done together with the data sanitizing step to bring back together entities that have been falsely split into many, as described with the "Richard Anderson" example above. This may be the right time to assign (internal or external) identifiers to entities if possible.

Having clean data content would the first milestone to a successful migration of the FIAF Treasures to any new system.

Milestone 2: Define Target Metadata Schema

The consequently following step is the decision which standards to use on the new platform, both for the metadata (e.g. a self-specified metadata standard, EN15907 in pure form or adapted for FIAF use, …) and field/table specific entries (Country codes, date/time formats, …). The latter should be clear after the data scrubbing step in this Milestone 1, but the decision for the metadata format is more complex.

Be aware that defining the target metadata schema has great impact on the behavior, complexity and feature requirements of the target system. It would make great sense to apply the MoSCoW method during this step, so it is at least clear what the *must-have* changes/features for the Treasures database schema are.

If not done already, then this is also the step where it needs to be defined which vocabulary sources and terms shall be used.

The options here are ordered from most to least effort required.

Option A: Moving towards EN15907

While analysing, the data has shown to be close to a good mapping to the EN15907 standard. The current schema contains some fields that cannot be mapped automatically to EN15907, e.g. "Notes", so a decision to stick to the EN15907 standard as good as possible would thus also mean to accept temporarily "expanding" it with required fields. For interchange with other institutions, this should not pose a problem, as non-standard-conform fields could be excluded in a data export format.

Implementing EN15907 compatibility however would not only require to convert text-only fields such as people or archives into agents and creating the corresponding relationships, but also definitely require splitting and reformatting data from certain fields across separate entities which increases the complexity required from an importer tremendously.

Option B: Moving towards the concept of Work / Item / Agent, but based on legacy Treasures field content.

The other realistic option is to use a more minimalistic approach and import the now sanitized database from Milestone 1, where the data is also split into single fields where necessary, to a rather simple schema to keep the complexity at a more manageable level and within the available technical and personal resources: Already splitting it up in Work / Item / Agent entities, staying with the legacy Treasures field definitions where necessary, but already applying wording, vocabulary and intention of EN15907. This also means that not only the pure data should be copied, but linking it together will also have to happen, so the correct entities are linked with the corresponding Items and other objects.

The main difference to Option A would be that several shortcuts would be possible, since location and definition of which data goes where would not have to adhere to external Standard definitions. This would allow not getting stuck with difficult data mappings, while allowing to move a step closer to an interoperable and state of the art data model.

Option C: Keep data layout and reproduce as-is.

It would also be feasible to simply take the database tables and fields that are currently actually used, leave the obsolete and superfluous ones out and migrate the current data as-is. No need to split any person names or map to Work, Item or Agents: Simply reproducing the current status quo, but having moved from a proprietary, offline database to an Open Source system with browser-based web access.

Milestone 2 can thus be described as making the decision for the metadata structure, which subsequently will be a strong pointer to what platform will be used, and have the data imported to said platform.

With a check on the outcome and if the new dataset is correctly searchable on the new platform’s backend and creation of new entries (including linking entities) is working correctly within all specified norms and formats, this Milestone would be concluded.

Milestone 3: Choose Target System (Backend)

Very important for further operations is the ability to import and export datasets.

Define and implement data import

For this, a standard data exchange format (e.g. CSV/XML layout) can be defined, in which the data has to be presented to FIAF. Once this decision is made, a working importer has to be implemented.

Since this cannot be done without already having chosen a target platform, defining import/export options and selecting a target system need to happen in the same step.

It is always nice to be able to import from multiple formats, but specify a preferred structure as a default.

Define and implement data export

Milestone 3 thus consists of defining, creating and testing the means of importing new data and exporting existing data. Exports either need to satisfy whatever the desired web frontend requires. This can be the current XML layout or something else, depending on the choice for dealing with the web frontend (Milestone 4).

Keep in mind to establish a working backup and restore solution for the database. It is important to at least also try the restore operation once, as too often only the backup is tested and after a real data loss, issues with the restoration format or process are found too late.

Having a clean dataset on a working platform for normal operations (Milestones 1-3) should be the target for the current planning stage. For the future, it is then possible to move on from the chosen platform and metadata format in a much more orderly manner and with significantly less effort.

Milestone 4: Attach Web-based Frontend

A web-based frontend where the Treasures data is made accessible to users to browse and search entries needs to be connected to the data stored in the backend in some way. Currently this is done via a simplified XML export format produced by Filemaker. The current web-frontends that use this XML are: the FIAF Website, Ovid Portal and Proquest.

It is important to be aware that even if the backend is successfully migrated to a web-based engine, the data still needs to be made available to a frontend in some way. A question that was left open at the moment is, which web portals FIAF intends or requires to deliver the Treasures data to in the future, or if it shall generally be converged in a single access on the FIAF website.

The options here are ordered from least to most effort required.

Option A: Keep using existing frontend with current XML export as-is.

This one is the fastest and easiest option, which would allow seamless replacement of the backend as early as possible, but limit access features to the currently existing functionality as-is.

For this option it would only be required to reproduce the current XML schema and field content layout with the export function of the new target system and make sure that the XML files are uploaded to wherever the existing frontends expect them already. It is very likely that there is no change needed on the frontend sides at all. The only downside is that all shortcomings of merged data fields, agents (people and archives) as just-text fields, etc that have been explained in this document already will still be present.

This approach is only suggested as an interim solution until a new frontend has been implemented, or if the current behavior is considered sufficient.

Option B: Use the new web-based backend as frontend.

Although not all browser-based collection management systems are intended for public access, it is usually possible to define less-priviledged user accounts which may use the cataloging backend for read-only research purposes.

This option may require little or almost no additional implementation overhead, but merely administrational efforts to create and manage the user accounts who would have access - depending on the number of accounts. Possible downsides of this option is that it is highly depending on the chosen target system how this interface would “look & feel”, as UI and features would be designed for editor/cataloguer use – which may differ from what researching users would expect or desire.

Making the backend available like this may also impose security concerns, since handing out logins to the same system may increase chances of unintended user/code exploits or priviledge escalation.

Except for that, this option could provide a compromise between Option A and Option C: being able to already offering new options in terms of data quality (e.g. agents and relationships instead of text-only), but not requiring to implement and administrate a separate web frontend.

Option C: Choose and implement a new frontend.

This may and probably will be a project in its own, depending on the chosen target system and the requested new features from a web-frontend update. Depending if and which web-access options the target backend already offers, the efforts for setting up a new frontend can be from merely changing a few config files to requiring to implement a REST API data exchange between two completely different vendors – ranging from being ready in one week or half a year.

It therefore does not make sense to estimate anything here at this point, before decisions about how to implement Milestones 1 to 3 have been made.

Time Estimations

All time estimations here focus on the parts of the tasks that would require someone with IT skills that would probably need to go beyond an average web-administrator: of sorts that the person is not only familiar with Linux systems, relational database structures and SQL statements, but also some medium to advanced level of scripting in a programming language other than SQL, such as Python or PHP for example. Shell scripting knowhow may be of help in some cases, but would not be sufficient alone.

The estimations are based on the assumption that most of the given task could be completed to a status where no more blocking use-cases are present and one could move on to the next milestone, even though the theoretical goal was not reached 100%.

The required developer time can be reduced by having made as many decisions as possible on the content / cataloging side beforehand (choosing and defining vocabulary terms, or which fields to keep as-is for the moment, etc).

One week is calculated with 5 working days, each with 8 working hours. So 2 weeks would be 10 days.

Given the expected durations of these milestones, it does not seem pay a developer on a per-hour basis, but rather negotiate a price per milestone or for the whole project, with the possibility to re-evaluate certain requirements in case unforeseen obstacles are encountered during the implementation.

Task	Expertise Required	Estimated Human Time
Milestone 1: Improve existing data quality		
Assessment: Which fields and manual or automated fix?	Cataloguing, Programming + Database Ratio depends on outcome of Assessment step.	
Deduplifying entities		3-5 weeks
Normalize vocabularies		2-4 weeks
Split people's names		2-3 weeks

Milestone 2: Define Target Metadata Schema	Cataloguing and familiarity with technical metadata format consequences. In dialog with programmers familiar with cataloging engines.	2-3 weeks. Regardless which option, the time to decide and define the schema is approximately the same. BUT: it completely changes the time and efforts in Milestone 3: Import, Export and Mapping the data.

Milestone 3: Choose Target System (Backend)		
Option A: Moving towards EN15907.	Programming + Database. Have option to have sysadmin for webserver (unix-like) for dialog/consulting.	8-24 weeks
Option B: Moving towards the concept of Work / Item / Agent, but based on legacy Treasures field content.		4-16 weeks
Option C: Keep data layout and reproduce as-is.		2-4 week