FeM XXC3 encoding pipeline

The core of this pipeline consists of tickets in different states. To hold the state of a ticket and handle all state transitions, a state maschine is implemented in the so called ticket tracker. This tracker provides a GUI (in form of a website) and a API for scripts.

All nodes set up for the pipeline have access to the raw (video) material through a shared mount of a network filesystem. The data resides on a fast storage file server.

Tickets

There are 3 types of tickets available. „Recording tickets“, „Encoding tickets“ and „miscellaneous tickets“.

All lectures of the event are compiled into a schedule, so called „Fahrplan“ with information about location, time, language, title of lecture, category, lecturer, length, etc and a FahrplanID.

For each lecture of the event a „recording ticket“ with the according FahrplanID is created. For each recording tickets a number of child tickets, the „encoding ticket“s are created. For additional tasks, like check user feedback on glitches or errors in released files, a „miscellaneous ticket“ can be created.

Each type of tickets has its own states and transitions. From every state, any other other state can be reached, although each state should have a predefined successor and predecessor for convenience.

The states are designed to represent processing states (recording, enconding) and static states (recorded, encoded, released). Both types are only distinguished through their names and not considered by the tracker.

Each ticket has got an additional flag „failed“ to enforce human interaction and prevent script execution.

Recording tickets

States: (in order of predefined transitions)

Name	Type	to by	from by	progress	description
locked	STATIC	human	human	0.0 %	lecture canceled, asked not to record, clearification needed…
scheduled	STATIC	importer	script A	0.0 %	lecture planed, event is in future
recording	PROCESS	script A	script A	12.5 %	event is currently being recorded
recorded	STATIC	script A, human	script B	25.0 %	tickets in this state are to be fetched by a script
merging	PROCESS	script B	script B	37.5 %	virtual dv file is created/mounted by script
merged	STATIC	script B, human	human	50.0 %	tickets are to be cut by hand

Name	Type	to by	from by	progress	description
cutting	PROCESS	human	human	62.5 %	human cuts and checks output
cut	STATIC	human	script C	75.0 %	ticket ist ready for copying
copying	PROCESS	script C	script C	87.5 %	make a backup copy of source video (needed?) and copy cutting metadata to tracker
copied	STATIC	script C	human	100.0 %	source video is ready to be processed by encoders

additional states:

fixing	PROCESS	human	human	50.0%	human video repair team prepares new source video

(next state: merged)

Encoding tickets

States: (in order of predefined transitions)

Name	Type	to by	from by	progress	description
material needed	STATIC	tracker, human	tracker	0.0 %	encoding ticket created, but source material not ready
ready to encode	STATIC	tracker, human	encoder	10.0 %	encoding with assigned profile ready to encode
encoding	PROCESS	encoder	encoder	20.0 %	encoding
encoded	STATIC	encoder	human	65.0 %	encoding done, quality assurance next
checking	PROCESS	human	human	70.0 %	human QA, maybe recoding, maybe change state of parent ticket
checked	STATIC	human	script D	75.0 %	encoding got through QA
postprocessing	PROCESS	script D	script D	80.0 %	post processing is done (copying, generate torrents, checksums, ...)
postprocessed	STATIC	script D	human	85.0 %	human QA for all files prepared to be released
ready to release	STATIC	human	human	90.0 %	release it into wild!
releasing	PROCESS	script E	script E	95.0 %	releasing (move to public folder), announce release, wiki update
released	STATIC	script E	human	100.0 %	done, human can revert release

additional states:

fixing	PROCESS	human	human	50.0 %	encoding fixing team prepares correctly encoded files

(next state: encoded)

Miscellaneous tickets

States: (in order of predefined transitions)

Name	Type	progress
open	STATIC	0 %
inprogress	PROCESS	50 %
resolved	STATIC	100 %
wontfix	STATIC	100 %

Scripts

Explanation of scripts used with states above.

Recording scheduler (Script A)

Sets state of recording tickets according to difference between scheduled time and current time.

Not polling through master.pl! Executed by cronjob.

- get all tickets in state „scheduled“ & NOT FAILED
- check start time with current time
- if overlapping: set state to recording

- get all tickets in state „recording“ & NOT FAILED
- check end time with current time
- if overlapping: set state to recorded

Difficulty: Easy

Fuse-Helper Mount4Cut(Script B)

Grabs tickets in recorded state, mounts fuse-vdv for cutting.

- get next ticket in state recorded (status will be „merging“)
- mount fuse-vdv
  - if mount exists, unmount it
  - read Fahrplan.Date and Fahrplan.Start and set Record.Starttime := Fahrplan.Date . '-' . Fahrplan.Start
  - read Fahrplan.Duration and Record.EndPadding and set Record.Stoptime := Record.Starttime + Fahrplan.Duration + Record.EndPadding
    - unit of Record.EndPadding is seconds
    - if Record.EndPadding is not present, use a default value and store it in tracker
    - syntax of Record.Starttime and Record.Stoptime is YYYY-MM-DD-hh-mm-ss
    - extract an integer from Fahrplan.Room and store in Record.Room
    - use Record.Starttime - StartPadding (unimportant constant, currently 5 min.) as start for fuse-mount
    - use Record.Stoptime - Record.Starttime + StartPadding as length for fuse-mount
    - derive capture prefix from Record.Room
    - derive intro filename from Fahrplan.ID (Fix Me! introduce separate ID for filesystem related stuff? sth. like „path slug“?)

        - derive target path from Fahrplan.ID
- set state to merged
- if error occures: set state to recorded & FAILED

Difficulty: Medium

Cutting postprocessor (Script C)

Post processing hook for recording tickets. For now only used to make a backup copy of cut material as rawdv and to store cutmarks as ticket properties in the tracker.

- get next ticket in state „cut“ (status will be „copying“)
- make backup copy of virtual cut file
- check if the cutmarks have been recognised by fuse-vdv correctly
- set cutmark metadata in tracker
    - Record.Cutin : parameter „inframe“ from fuse, unit is # of frames from beginning of virtual uncut DV file
    - Record.Cutout : parameter „outframe“ from fuse, unit is # of frames from beginning of virtual uncut DV file
    - Record.Cutintime : absolute reality-wall-clock-time of beginning of cut DV file
    - Record.Cutouttime : absolute reality-wall-clock-time of end of cut DV file
- set state to „copied“
- if error occures: set state to cut & FAILED

Encoding postprocessor (Script D)

Post processing hook for encoding tickets. Used for generating checksums files, torrent files, upload to hidden mirror folder, etc.

- get next ticket in „checked“ (status will be „postprocessing“)
- post process….
- set state to „postprocessed“
- if error occures: set state to checked & FAILED

Release script (Script E)

Finally releasing videos!

- get next ticket in state „ready to release“ (status will be „releasing“)
- move to public folder
- set flag/trigger to update wiki page
- announce in jabber muc, irc, …
- set state to „released“
- add property „Release.Datetime“, value is the current time in the format YYYY.MM.DD_HH:MM:SS
- add property „Release.Count“ with a value of 1 unless the property exists, in this case increase the value by 1

Fahrplan checker

Checks periodically for fahrplan updates and informs muc.

cron job...

Fahrplan importer (by hand)

Click page in gui...

Properties overview

Section	Name	set on	Content	created/written by	consumed by
Fahrplan	XML	Project	link to fahrplan xml	Tracker	fahrplan importer
Meta	Album	Project	album meta data for encoded files, like event title	Tracker	encoder scripts
~~Meta~~	~~Comment~~	~~Project~~	~~comment meta data, e.g. deeplink to lecture page~~		
Meta	License	Project	license meta data, e.g. CC-BY-NC-ND	Tracker	encoder scripts
Meta	Year	Project	year, e.g. for copyright notice		encoder scripts
Fahrplan	ID	Recording ticket	attribute extracted from Fahrplan XML	fahrplan importer	script B
Fahrplan	Date	Recording ticket	attribute extracted from Fahrplan XML	fahrplan importer	script B
Fahrplan	Start	Recording ticket	attribute extracted from Fahrplan XML	fahrplan importer	script B
Fahrplan	Day	Recording ticket	attribute extracted from Fahrplan XML	fahrplan importer	script B
Fahrplan	Duration	Recording ticket	attribute extracted from Fahrplan XML	fahrplan importer	script B
Fahrplan	Room	Recording ticket	attribute extracted from Fahrplan XML	fahrplan importer	script B
Fahrplan	Slug	Recording ticket	attribute extracted from Fahrplan XML	fahrplan importer	script B
Fahrplan	Title	Recording ticket	attribute extracted from Fahrplan XML	fahrplan importer	script B
Fahrplan	Subtitle	Recording ticket	attribute extracted from Fahrplan XML	fahrplan importer	script B
Fahrplan	Track	Recording ticket	attribute extracted from Fahrplan XML	fahrplan importer	script B
Fahrplan	Language	Recording ticket	attribute extracted from Fahrplan XML	fahrplan importer	script B
Fahrplan	Abstract	Recording ticket	attribute extracted from Fahrplan XML	fahrplan importer	script B
Fahrplan	Person_list	Recording ticket	attribute extracted from Fahrplan XML	fahrplan importer	script B
Record	Starttime	Recording ticket	`Fahrplan.Date . '-' . Fahrplan.Start`, Syntax is `YYYY-MM-DD-hh-mm-ss`	script B	
Record	Stoptime	Recording ticket	`Record.Starttime + Fahrplan.Duration + Record.EndPadding`, Syntax is `YYYY-MM-DD-hh-mm-ss`	script B	
Record	EndPadding	Recording ticket	seconds to add to the end when creating fuse mount	script B or web GUI (cutting wizard)	script B
Record	Room	Recording ticket	integer extracted from Fahrplan.Room	script B	
Record	Cutin	Recording ticket	`inframe` from fuse, unit is # of frames from beginning of virtual uncut DV file	script C	
Record	Cutout	Recording ticket	`outframe` from fuse, unit is # of frames from beginning of virtual uncut DV file	script C	
Record	Cutintime	Recording ticket	absolute reality-wall-clock-time of beginning of cut DV file	script C	
Record	Cutouttime	Recording ticket	absolute reality-wall-clock-time of end of cut DV file	script C	
Record	DurationFrames	Recording ticket	Fix Me!	Fix Me!	
Record	DurationSeconds	Recording ticket	Fix Me!	Fix Me!	

Section	Name	set on	Content	created/written by	consumed by
Record	Language	Recording ticket	Language that is selected by cutter in web GUI	web GUI (cutting wizard)	
Record	SourceReplacement	Recording ticket	name of a raw DV file (without path and **without file suffix**) that should be used instead of recorded AV data	web GUI (fixing wizard) Fix Me! to be implemented	script B
Record	AVDelay	Recording ticket	number of milliseconds of AV delay, used as parameter for encoding	script B	
Project	Slug	(virtual)	Slug of project, may be used as filename prefix	Tracker	
Encoding	Basename	(virtual)	`<project-slug>-<Fahrplan.ID>-<Record.Language>-<Fahrplan.Slug>`	Tracker	script E
EncodingProfile	Basename	(virtual)	<Encoding.Basename>[-<encodingprofile-slug>]	Tracker	
EncodingProfile	Extension	(virtual)	<encodingprofile-extension>	Tracker	
EncodingProfile	Slug	(virtual)	Slug-Property of the encoding profile associated with the encoding ticket	Tracker	
Release	Datetime	Encoding ticket	time of release in the format YYYY.MM.DD_HH:MM:SS	script E	
Release	Count	Encoding ticket	increasing # of releases of the asociated file, starting with 1	script E	
Processing	Path.Raw	Project	path to raw files, e.g. /c3mnt/fuse		
Processing	Path.Tmp	Project	path to tmp files, e.g. /c3mnt/tmp		
Processing	Path.Output	Project	path to output files, e.g. /c3mnt/encoded		
Processing	Video.AspectRatio	Project	ascpect ratio of source video		

C3TT - C3 Ticket Tracker

GUI

The GUI is a website for human interaction with the ticket state maschine.

Users authenticate with username and password. The interface is designed to be intuitive and should explain itselfs.

API general information

The API is designed as XML-RPC interface through a HTTP connection to the tracker.

RPC stands for Remote process call, which consists of the method name and a stack of parameters. The return value can be a scalar or an array depending on the method implementation.

Authentication

Scripts (resp. „clients“) using the API authenticate themselves with an unique login hash („hash“), which is an MD5 hash of the concatination of the client's hostname („FQDN“) and a preshared secret. The secret is a **global** property of the tracker.

```
hash := MD5 ( FQDN . secret )
```

The FQDN should resolve via DNS to the IP address doing the current RPC call since the tracker is allowed to check this for security reasons.

API Version 3.0

New API version adapeted for cleaned up state maschine. Some methods were removed, some new ones introduced, new URL scheme, multi-project compatibility…

URL: http://tracker/rpc/$hash/$FQDN[/$project]

The variable `project` is a placeholder for the „project slug“, see method call `getProjects`.

Following methods don't require a project slug in URL:

- getVersion()
  - gets API version
- getServices()
  - gets list of states related to services, e.g. encoding
- getProjects([$read_only = false, $list_encoding_profiles = true])
  - gets array of project data, optionally with related encoding profiles
- ping([$ticket_id, [$log_message_delta, [$progress]]])
  - $ticket_id - optional ticket id of ticket currently working on
  - $log_message_delta - service output since the last ping method call
  - $progress - information about processing progress in percent, if available
  - return value:
    - when called **with** parameters
      - *„OK“* - everything is fine, keep working
      - *„Ticket lost“* - kill all process, start all over
    - when called **without** parameters
      - *„OK“* - everything is fine, come back later
      - *„Reboot“* - perform a hardware reboot
      - *„Shutdown“* - perform a hardware shutdown

The following methods require a project slug in URL:

- getTicketProperty($ticket_id, $name)
  - property of ticket's parent is returned if the given ticket does not have the requested property
- getTicketProperties($ticket_id, $pattern = null)
  - ticket's properties are merged with the parent ticket's properties if there is a parent ticket
  - ticket property names are of postgresql type „ltree“ and the pattern is used with the „~“ operator see documentation
- setTicketProperty($ticket_id, $name, $value)
- setTicketProperties($ticket_id, $properties)
- getJobfile($ticket_id)
  - for encoding tickets: returns rendered jobfile from encoding profile XML template
- getUnassignedTicketsInState($state)
  - returns only unassigned tickets in state $state
- assignNextUnassignedForState($state)
  - gets next unassigned ticket ready to be in state $state after transition
- setTicketDone($ticket_id[, $log_message])

    - unassigns ticket and set state to according state after procressing by service
    - $log_message should be only applied if it contains any relevant information (no „success“ message)
- setTicketFailed($ticket_id[, $log_message])
    - unassigns ticket and set „failed“ flag
- setTicketNextState($ticket_id, $current_state[, $log_message])
    - sets the state of a **unassigned** ticket to the next state in the normal workflow
- getEncodingProfiles($encoding_profile_id = null)
    - gets list of encoding profiles (with details) of current project
    - get details of single profile if id provided
- addLog($ticket_id, $message)
    - method to add a (short!) log message to a ticket

TODO

Moved to C3TT Trac.