Åpen video på nett, det siste års utvikling

1. august 2010

Innhold

1 Introduksjon

I dette dokument vil det gjennomgåes det siste års utvikling innenfor åpen video på nett, og verktøyer for distribuering av dette. Det legges vekt på nye løsninger, tjenester og standarder som er kommet, men også eksisterende som er blitt forbedret.

2 Standarder

I løpet av de siste 12 månedene har det foregått nevneverdig utvikling innen både HTML5-video, underteksting så vel som generell håndtering av video. Et nytt åpent format med mange store aktører i ryggen har tredd frem, og støtte for visning av video gjennom HTML5 video-taggen er blitt betydelig bedre. Her skal vi ta for oss betydelige nyheter innenfor de nevnte feltene.

2.1 HTML5

Firefox lanserte 30. juni den første nettleseren med støtte for HTML5 video-taggen [1] . Siden den tid har de fleste største nettlesere kommet med en generell støtte, med unntak av den største, Internet Explorer, som vil gi støtte med neste utgivelse, versjon 9. Noen lanseringsdato er foreløpig ikke fastsatt.

Det er vedvarende ingen enighet om hva slags format som skal brukes på selve transportstrømme, hverken innkapsling, eller formater innad i innkapslingen. Safari og den kommende utgaven av Internet Explorer, støtter utelukkende innkapslingsformatet MP4, med h.264 video- og AAC som lydkodeks. Firefox og Opera støtter kun Ogg som innkapslingsformat, med Theora som video- og Vorbis som lydkodeks. Chrome på den andre siden støtter begge formater.

I tillegg til utbredelsen av Ogg- og MP4-innkapslingsformatet, er det kommet et tredje alternativ til de to eksisterende formatene. Formatet er utviklet av Google og går under navnet WebM [2] . Formatet baserer seg på VP8 som video- og Vorbis som lydkodeks, hvor disse kodeksene er faste. WebM er gjort tilgjengelig under en åpen lisens som tillater fritt kommersielt bruk. En mer grundig gjennomgang av WebM gjøres i et underkapittel funnet lenger ned.

Format	IE	Firefox	Safari	Chrome	Opera	iPhone	Android
HTML5 video	v9.0+	v3.5+	v3.0+				
Ogg(Theora+Vorbis)		v3.5+		v5.0+	v10.5+		
MP4(h.262+AAC)			v3.0+	v5.0+		v3.0+	v2.0+
WebM(VP8+Vorbis)	v9.0+ [3]	v4.0+	[4]	v6.0+	v10.0+		

- støttet i siste utgivelse
- støtte kommer i gitt fremtidig utgivelse
- støtte kommer i ubestemt fremtidig utgivelse
- ingen støtte, hverken nå eller i kjent fremtid

[1] http://www.engadget.com/2009/06/30/firefox-3-5-arrives/
[2] http://www.webmproject.org/
[3] brukeren må installere en plugin
[4] brukeren må installere en 3.parts plugin for QuickTime

2.2 WebM

19. mai, 2010 lanserte Google det åpne innkapslingsformatet WebM, sammen med videoformatet VP8. Innkapslingen bruker kodeksen Vorbis for lyd og VP8 for video, dermed er alle ledd basert på åpne standarder.

Flere fordeler ligger til grunne for at dette formatet kan bli svært utbredt:

- åpent format
- formatene brukt for video- og lydstrømmer er strengt definerte
- mange store aktører stiller seg bak formatet
- jobber med maskinvareakselerasjon

Formatet er fullstendig lisensfritt og åpent. Det gjør det enklere å integrere både utviklingsmessig og økonomisk, da det ikke er noen lisensieringsavgift.

Formatene brukt for lyd- og videostrømmer er fastsatte, for video brukes VP8 og for lyd Vorbis. I praksis betyr det at alle produkter som støtter WebM vil støtte enhver WebM-fil. Dette er ikke et faktum for de fleste andre innkapslingsformater, hvor man kan velge mellom forskjellige kodekser [5] .

Mange store og viktige aktører stiller seg bak formatet, både program- og maskinvareleverandører. Herunder finner man blant annet Firefox, Opera, Adobe Flash, ARM, AMD og NVIDIA. En fullstendig liste finnes på prosjektets nettsider. [6]

Støtte er allerede inkorporert i utviklerutgaver av flere store nettlesere, og flere er i vente.

2.3 Underteksting

Det finnes per i dag ingen definerte standarder for underteksting av video, men det er flere interessante pågående prosjekter.

Det finnes flere løsninger for underteksing av HTML5-video, men ingen standardiserte. Løsningen per i dag er å bruke egenutviklede JavaScript-rammeverk for å oppnå underteksting, i påvente av en fastsatt standard. Ved bruk av denne løsningen er det ikke mulig å oppnå teksting av video i fullskjermmodus. Med en fastsatt standard slipper man å bruke JavaScript, man muliggjør undertekster i fullskjerm, og det vil gjøre det langt enklere for utviklere å benytte seg av en satt standard fremfor å skape sin egen.

Det er ikke tatt høyde for bruk av undertekster i Flash-avspillere.

2.3.1 Timed Text Markup Language

W3 har gitt ut en anbefalt standard for å knytte tekst til video, bundet i tid. Standarden spesifiser mulighet for valgfri plassering av tekstinnhold, med full

[5] eksempelvis MP4, hvor man i praksis står fritt til å velge mellom nærmest et hvilket som helst kodeks til både lyd og video

[6] `http://www.webmproject.org/about/supporters/`

frihet for formatering i form av form, farge, animering og plassering. Standardenens spesifikasjoner for formatering vil gjøre det samme for video som CSS gjorde for HTML, hvor man trekker formatering ut i et eget stilsett.

Sammen med videofilen vil man få et XML-formatert dokument som definerer underteksten, tilknytting til tid, plassering, dens form og animering. Oppsetttet av dokumentet har store likheter til et HTML-dokument, hvor man skiller formatering og innhold i et hode og kropp, respektivt. Som med HTML og CSS kan hver enkelt tekstsekvens knyttes til en definert formateringsgruppe, og i tillegg også en definert plassering. I likhet med CSS kan man også gjøre inline-formatering, hvor man spesifikkt formaterer hver enkelt tekstsekvens uten bruk av grupper.

Prosjektet er fullført, og anbefalingen som gis er endelig. Mer informasjon finnes på deres nettsider. [7]

2.3.2 Mozilla Drumbeat Universal Subtitles

Mozilla tar del i et prosjekt for skaping, deling og visning av undertekster i video. Det finnes for øyeblikket en tidlig prøveutgave av webverktøyet for å lage og vise undertekster, og baserer seg på HTML5 og JavaScript. [8] For visning ønsker de å skape en åpent spesifisert protokoll, som skal kunne brukes i nettlesere så vel som frittstående applikasjoner.

Muligheten for søking og visning i frittstående applikasjoner er under utvikling, uten en prøveutgave tilgjengelig. Denne funksjonaliteten er ikke beskrevet i detalj, så det er vanskelig å si hva en kan forvente i fremtiden.

Når en endelig standard for underteksting fremlegges av W3, vil nok selve annoteringsfunksjonen være den mest nyttige fra dette prosjektet. Rammeverket for visning av undertekster er skrevet i JavaScript, og nedsidene med JavaScript følger da med.

Mer informasjon rundt prosjektet finnes på deres nettsider [9] .

2.3.3 jQuery srt

I påvente av implementasjon av standarden for underteksting, er det blitt skrevet en JavaScript-komponent for jQuery for å oppnå visning av undertekster. [10] Sammen med videofilen trenger man en fil i SRT-format [11] , som blir lenket til videoen. Når siden lastes inn leses SRT-filen, og teksten knyttes opp til tidsområder i videoen. Teksten vises deretter innenfor en HTML DIV-tagg, som gjør at man selv kan formatere det som man vil.

Det er dog per nå ikke mulig å vise tekst når man viser videoen i fullskjerm, og utviklerens nettside gir minimalt med informasjon. Det ser heller ikke ut til å være mulig å kunne velge mellom flere forskjellige språk, dog det antageligvis vil kreve en enkel modifikasjon av JavaScript-et.

[7] `http://www.w3.org/TR/ttaf1-dfxp/`
[8] `http://universalsubtitles.org/`
[9] http://www.drumbeat.org/project/universal-subtitles
[10] Jan Gerber, jQuery srt: `http://v2v.cc/~j/jquery.srt/`
[11] `http://zuggy.wz.cz/`

2.4 Media Fragments URI

For å gjøre det enklere å jobbe med lyd og video på nett, har W3 igangsatt utviklingen av en standard for hvordan man bedre skal håndtere multimedia på nett. Dette vil bringe frem essensielle funksjoner på server-side, hos klient, og leddene imellom. Av funksjonalitet nevnes søking, uthenting av regioner både innenfor bilde og video, og tidsintervaller av video.

Denne standarden skal hovedsaklig omfatte alle tidsbaserte ressurser, men samtidig også stillbilder som ikke er bundet til tid.

Standarden er forsatt en skisse, og en endelig anbefaling skal etter planen fremlegges i januar, 2011.

Søking Søking i lyd- og videostrømmer gjøres i dag på klientsiden. Utifra datastrømmen og eventuelle hint fra server, estimerer klienten dataområdet som er nødvendig for å søke seg til ønskede tidspunkt. Deretter forespør klienten serveren om dette dataområdet. Om den ikke klarer å regne seg frem, må hele mediet lastes ned.

Det Media Fragments URI vil tilføre søking, er at klienten forespør serveren om hvilket tidsområde som er ønskelig, uten å håndtere hvilket dataområde det vil omfatte. [12] Serveren vil deretter håndtere jobben det er å finne ut hvilket dataområde som korresponderer med den ønskede tiden, og returnere den nødvendige dataen.

Regioner Det skal være mulig å hente ut egendefinerte regioner av stilbilder så vel som video. Det innebærer at man velger ut et pikselområde man ønsker data for, man sender med den definerte regionen til server, og server returnerer kun data for den spesifikke regionen. På denne måten får man kun tilsendt den dataen som er nødvendig for å vise den ønskede regionen.

Denne funksjonen gjelder både løpende video så vel som stillbilder.

Sanntidsstrømmer I en sanntidsstrøm skal det være mulig å stoppe strømmen, for så å kunne fortsette strømmen der man slapp, slik man ville gjort med en tradisjonelt lagret medie.

Tekniske endringer Vi skal ikke gå spesielt dypt inn i de tekniske aspektene av denne standarden, ettersom det ikke er relevant.

Endringene som skal måtte gjøres på server- og klientside så vel som generell infrastruktur, sies å skulle holdes til et minimum, samtidig som man vil oppnå best mulig kompatibilitet. Det skal ikke være nødvendig å måtte gjøre tilpasninger for enkeltformater på klientsiden.

Mer om Media Fragments Spesifikasjonene for denne standarden er omfattende, og tilbyr en rekke funksjonalitet hvorav ikke alle er blitt nevnt her. Les de spesifiserte brukerscenarioene som standarden er bygget rundt på for å få en mer omfattende forståelse. [13]

[12] For å kunne dra nytte av mellomlagret data i proxyservere, forespørres server først med tidspunkt som parameter, server returnerer hvilket dataområdet det korresponderer med, og klienten forespør om det aktuelle dataområdet på tradisjonelt vis.

[13] `http://www.w3.org/2008/WebVideo/Fragments/WD-media-fragments-reqs/`

Det siste utkastet av standarden er også å finne på nett. [14]

3 Biblioteker

Herunder finnes rammeverk og generelle verktøy for behandling av åpen video. TTC er den eksisterende transkodingsløsningen som brukes per i dag, og VITAL er et nytt prosjekt i regi av Columbia University for inkorporering av video i typiske stiler og rapporter.

3.1 TTC

Tista Transcode(TTC) er løsningen som brukes i dag for transkoding av multimedia. Man gir systemet jobber som skal utføres ved å oppgi ønsket utgangformat og inngangsfilene til tjeneren. Disse jobbene distribueres til transkodingsklienter som løpende spør om nye jobber. Når jobben er utført begynner igjen klienten å spørre om nye jobber. Ettersom man teoretisk sett kan ha et ubegrenset antall klienter, er systemet svært skalerbar.

Dette verktøyet er utviklet av Høgskolen i Østfold, den gang for å transkode eldre opptak gjort på Stortinget. Programmeringsspråket som blir brukt er Python, og kildekoden finnes tilgjengelig på prosjektsiden hos Google Code [15] .

I ettertid er systemet blitt videreutviklet og integrert med VideoBin [16] . Dette tillater en bruker å laste opp en film i et tilnærmet hvilket som helst format, og få returnert filmen transkodet i et åpent format. Brukeren kan da selv velge mellom å laste ned den transkodede filmen, eller å spille den av direkte fra nettsiden ved bruk av video-taggen som definert i HTML5-standarden. Siden ligger tilgjengelig på nett, men filmene blir øyeblikket ikke transkodet [17] .

Utgangsformatene er per i dag satt til OGG(Theora+Vorbis) og MP4(h.264+AAC). For transkoding av OGG benyttes FFmpeg2theora, [18] og for MP4 benyttes HandBreak. [19]

For å inkorporere støtte for enkoding til WebM kan man benytte seg av siste utgave av FFmpeg [20] , som har innebygd støtte for dette.

3.2 VITAL

VITAL er et verktøy for å gjøre annoteringer til lyd og video, og visning og samling av disse. Man har også mulighet til å laste opp egenprodusert materiale, og annotere på dette.

Bakgrunnen for VITAL er å skape et verktøy for annotering av video, og for å skape multimediahistorier ved bruk av tekst og video. Verktøyet er tiltenkt for bruk mellom instruktør og student, for å kunne utføre og evaluere oppgaver. Det er et fullstendig nettbasert verktøy som kan nås og brukes av nettlesere med Adobe Flash installert.

[14] `http://www.w3.org/TR/media-frags/`
[15] `http://code.google.com/p/tista-transcode/`
[16] `http://videobin.org/`
[17] `http://ogg.hiof.no/`
[18] `http://v2v.cc/~j/ffmpeg2theora/`
[19] `http://handbrake.fr/`
[20] FFmpeg, versjon 0.6

Ved åpning av verktøyet får man en liste over oppgaver, med en beskrivelse og multimedia tilhørende hver enkelt oppgave. En oppgave kan være å gjøre en skriftlig analyse av en samling videoklipp. Under avspilling har brukeren mulighet til å velge start og slutt på en sekvens og stoppe klippet. Brukeren kan da gjøre en annotasjon som blir knyttet opp til den gitte sekvensen. Ved sammensetning av analysen får man en side bestående av tre deler: oppgaveteksten, et tekstområde hvor brukeren kan skrive, og listen over videoklipp man har gjort annoteringer i. Ved sammensetning av analysen kan man putte klipp direkte inn i teksten, som peker direkte til sekvensen og annotasjonen som brukeren har gjort. Ved å bruke dette verktøyet har man mulighet til å gjøre direkte referanser til spesifikke deler av én eller flere vidoeklipp. Dermed slipper man å beskrive klippene med tekst, og man kan enklere forstå sammenhengen.

Prosjektet er i regi av Columbia University, og er for øyeblikket i prøvestadiet hos en rekke utdanningsinstitusjoner rundtom i USA.

Les mer informasjon på nettsidene deres [21] .

4 Publiseringsløsninger

En publiseringsløsning i dette sammenheng menes et system som kan ta seg av selve publiseringen, og samtidig også ha kontroll på videoen og dataen som er tilgjengelig. Et slikt system vil måtte kunne håndtere hele livslinjen til en video, fra opplasting, transkoding og lagring, til publisering og avspilling.

Herunder vil det gjennomgåes forskjellige publiseringsløsninger, og tilleggsmoduler for å håndtere video i eksisterende løsninger.

Erfaringer er også hentet inn fra en tidligere rapport utredet av Andreas Bergstrøm, ved Høgskolen i Østfold [22] .

4.1 Kaltura

Kaltura er et system for publisering av video, men er ikke et CMS-system. Det finnes en gratis- og en utvidet betalingsløsning, hvor betalingsløsningen tilbyr lagring og streaming av video fra Kaltura's egne servere. Ved bruk av gratisløsningen må man selv håndtere lagring, transkoding og streaming på egne maskiner. Kaltura i seg selv er ikke et CMS-system, men tilbyr støtte for andre CMS-systemer gjennom bruk av tilleggsmoduler [23] , blant annet Drupal og Joomla. Programmeringsspråket som er brukt på serversiden er PHP.

Videoavspilleren støtter HTML5, med tilbakefall til Flash. Denne løsningen finnes også under navnet mwEmbed [24] . Man har mulighet til å tilpasse utseendet på avspilleren, både Flash- og HTML5-utgaven. Noen eksempler finnes på nett [25] .

[21] `http://ccnmtl.columbia.edu/our_services/vital/introduction_to_vital.html`

[22] `https://ow.feide.no/mms` (krever pålogging)

[23] `http://exchange.kaltura.com/`

[24] `http://www.html5video.org/`

[25] `http://corp.kaltura.com/technology/video_player`

Undertekster i videoer er ikke støttet direkte, men gjennom en tilleggsmodul med navn PLYmedia. Dette avhenger dog at man benytter seg av betalingsløsningen.

Systemet avhenger av å kjøre på en Linux-maskin med en rekke nødvendige eksterne biblioteker, på en webserver med støtte for PHP. Kaltura er testet og fungerer på Ubuntu 10.04 og CentOS 5.5.

Nettsiden for betalings- og gratisløsningen er kaltura.com [26] og kaltura.org [27] , respektivt.

Gratisutgaven av Kaltura er distibuert under lisensen Creative Commons [28] .

4.2 MediaMosa

MediaMosa er et Drupal-basert CMS-system, med mulighet for transkoding og avspilling av video. Systemet benytter seg av FFmpeg som transkodingsapplikasjon, som dermed gir støtte for de mest typisk brukte formater. Herunder ligger h.264(MP4+AAC), OGG(Theora+Vorbis) og WebM(VP8+Theora). Det skal angivelig være mulig å benytte seg av andre eksterne transkodingsapplikasjoner, men her er dokumentasjonen mangelfull. Programmeringsspråket som er brukt på serversiden er PHP.

Det er for øyeblikket ingen støtte for HTML5 video-taggen, men støtte skal komme i en fremtidig utgivelse. Det er heller ingen støtte for underteksting, og det foreligger heller ingen konkrete planer om det i fremtiden.

Gjennom administrasjonsgrensesnittet har man mulighet til å sette parametere for transkoding i FFmpeg. Parameterne man har mulighet til å sette er derimot begrenset, og man kan ikke sette parametere manuelt om de ikke finnes gjennom webgrensesnittet.

Systemet er utviklet i Nederland, og majoriteten av dokumentasjon som foreligger er gjort på nederlandsk. Mye er dog i ettertid blitt oversatt til engelsk, men dokumentasjonen virker enda begrenset og mangelfull.

MediaMosa er distibuert under lisensen GPLv2 [29] .

5 Videoavspillere

I tillegg til fullstendige CMS-er skal vi også se på frittstående videoavspillere, som ikke trenger å være del av et CMS.

Det finnes utallige forskjellige HTML5-videoavspillere, derfor nevnes kun nåen få som har unike egenskaper i forhold til mengden. Spesielt ønsket funksjonalitet er, underteksting, og tilbakefall til andre formater når det ikke er støtte for HTML5-video. Spesielt underteksting er et problem som enda er tilgode å bli løst på en god måte. Det vil forhåpentligvis bli bedre etterhvert som standarden for underteksting blir implementert av de store nettleserne.

[26] http://kaltura.com/
[27] http://kaltura.org/
[28] http://creativecommons.org/
[29] http://www.gnu.org/licenses/old-licenses/gpl-2.0.txt

5.1 Kaltura

Tilbyr en åpen videoavspiller, som faller tilbake til Flash eller alternativt Java om ikke HTML5 kan avspilles. Har også støtte for underteksting i samtlige utgaver, men denne funksjonaliteten er fortsatt under utvikling og er derfor ikke feilfri.

Går også under navnet mwEmbed, [30] et prosjekt som har som mål å utvide støtten for video i programvaren, MediaWiki. Dette innebærer da blant annet Wikipedia. Det samme rammeverket kan hentes frittstående fra Kaltura's egne sider. [31]

Avspilleren for både HTML5, Flash og Java har identisk brukergrensesnitt, til tross for noen synlige forskjeller.

Rammeverket av avspilleren er gjort tilgjengelig under Creative Commons Share Alike 3.0, [32] som i praksis tilsier at man kan gjøre hva en vil med det, men det en endrer og selv lager må også gjøres tilgjengelig til offentligheten under samme lisens.

5.2 Open Standard Media Player(OSM)

Åpen videoavspiller. Spiller av HTML5 video, med tilbakefall til Flash. Har også mulighet for integrasjon av Youtube og Vimeo direkte i avspilleren, og grensesnittet i avspilleren er identisk uansett hvilket format man spiller av. Det er også lagt til mulighet for at brukeren kan rangere hver enkelt video, som kan brukes til å si noe om popularitet.

Avspilleren er bygd opp rundt jQuery og jQuery UI, som enkelt muliggjør tilpassing av utseendet på spilleren.

Det er ikke støtte for underteksting, men det skal være planlagt støtte for det i fremtiden.

5.3 JW Player

En av de mest brukte åpne videoavpillerene på nett. Er i utgangspunktet bygget på Flash, men det utvikles en identisk HTML5-avpiller som er tilgjengelig som Beta.

Har støtte for HTML5-video, og faller sømløst tilbake til Flash når HTML5-video ikke kan spilles av. Felter for videoavspilling defineres ved bruk av video-taggen, og ved å legge til en liten snutt JavaScript vil avspilleren av seg selv falle tilbake til en Flash-avspiller som benytter seg av den h.264-formaterte filmen.

Endring av utseendet på avspilleren gjøres ved bruk av CSS, JavaScript og et bilde som brukes som mal for avspilleren.

Ettersom denne avspilleren er annrkjent og mye brukt, vil det være spennende å se hvordan den utvikler seg i tiden som kommer. Det er ikke støtte for underteksting direkte, men det er tilgjengelig ved bruk av tilleggsmoduler. Det er dog ingen støtte for underteksting av HTML5-video. Det finnes også moduler for integrasjon med flere CMS-er.

[30] `http://www.mediawiki.org/wiki/MwEmbed`
[31] `http://www.kaltura.org/project/HTML5_Video_Media_JavaScript_Library`
[32] `http://creativecommons.org/licenses/by-sa/3.0/`

En rekke demonstasjoner finnes på nettsidene deres. [33]

5.4 Video for Everybody

Video for Everybody er en lettvekter av en videoavspiller. Den spiller av HTML5-video og har mulighet til å falle tilbake til Flash. Ulikt fra de fleste andre avspillere fungerer denne avspilleren fullstendig uten bruk av JavaSctipt. Det kan være en svært gunstig for scenarioer hvor man ikke kan eller ikke ønsker å benytte seg av JavaScript, eksempelvis RSS-lesere. I verste fall om alt feiler, vil man få opp et bilde og mulighet for å laste ned videoen manuelt i de forskjellige formatene som blir lagt ut.

Mer informasjon finnes på nettsidene deres. [34]

6 Synkronisert video og grafikk

I noen scenarioer kan det være ønskelig å synkronisere en vidoeavspilling med passende tekst/bilder. Eksempelvis i et opptak eller direktestrøm av en forelesning, hvor foreleseren legger ut tekstlig og/eller billedlig materiale som er knyttet til temaet som diskuteres.

En lignende løsning bygget på Flash eksisterer, utviklet for en rekke italienske universiteter. [35] Ettersom den er bygd på Flash og er lukket, er det ikke mulig å benytte seg av denne løsningen, men det gir allikevel en god idé om hvordan en tilsvarende løsning kan se ut.

Med HTML5 og video-taggen vil det være enklere å oppnå en slik løsning, uten å måtte benytte seg av tilleggsmoduler som Adobe Flash eller Microsoft Silverlight. Man kan enkelt hente ut tidsinformasjon fra videoen, og ved hjelp av JavaScript kan man da knytte til og vise det passende materialet.

Utfordringen med en slik løsning vil være å definere et format for kommunikasjon fra server til klient, som forteller når det forskjellige materialet skal være synlig for brukeren. For et opptak hvor denne informasjonen allerede finnes vil løsningen være enkel. Før avspillingen av videoen begynner henter klienten inn den nødvendige informasjonen om hva som skal vises og når. I en direktestrøm vil ikke en slik løsning fungere, da det ikke er bestemt når materialet skal vises. Løsningen vil da være å kontinuerlig forespørre serveren om det er nytt materiale som skal vises eller skjules. En slik løsning vil måtte være godt gjennomtenkt for å unngå unødvendig belastning av serveren, men er absolutt gjennomførbart.

[33] `http://www.longtailvideo.com/support/jw-player/jw-player-for-html5`
[34] `http://camendesign.com/code/video_for_everybody`
[35] `http://demo.cineca.it/mod/ltol/view.php?id=2`