

nello.

Documentation

26. March 2009
Flo Eberle

Table of contents

1. Introduction.....	3
1.1 Purpose of this documentation.....	3
1.2 What is «nello»?.....	3
2. Technical Facts.....	4
2.1 Server side.....	4
2.2 Client side.....	4
3. Data Structure (Web Application).....	5
3.1 General.....	5
3.2 Main-Folders.....	5
3.3 Folder «Includes».....	6
3.4 Folder «Live» and Subfolders.....	7
3.5 Special directories.....	8
4. Database.....	8
4.1 General.....	8
4.2 Users, Groups and Permissions.....	8
4.3 Transactions.....	9
4.4 Entity Relationship Model.....	9
5. Sample Page Hit Sequence.....	9
6. Core-Functions.....	11
6.1 Create Account.....	11
6.2 TV-Guide.....	11
6.3 Shop.....	12
6.4 Watch TV.....	12
7. JavaScript / JQuery	13
7.1 The JQuery Library.....	13
7.2 Used JQuery Plug ins.....	14
7.3 Other JS Stuff.....	15
7.4 Self written JS functions.....	15
7.5 JavaScript (and CSS) Compressing	15
8. Backend.....	15
8.1 Security Concept.....	15
8.2 Authorisation.....	15
8.3 Backend Pages.....	16
9. Miscellaneous.....	17
9.1 Revision Control (Git).....	17
9.2 Publishing (nello_sync.sh).....	17
9.3 How to «clone database».....	18
9.4 How to create «auto-date trigger».....	18
10. Dictionary.....	19
11. Version / Revision.....	19
12. Attachments.....	19

1. Introduction

1.1 Purpose of this documentation

- ☑ General Project Overview
- ☑ Core Procedures Explained
- ☑ Important Hints for future developing
- ☑ Data Structure and Database Design illustrated
- ☑ Resources needed (e.g. Servers, Interfaces and so on)

1.2 What is «nello»?

Nello is a further development of «ADSL-TV». ADSL-TV offered WMV TV streaming. Because of the limited compressing features and the operating-system dependent software we tried get rid of the old technology. The new idea was a stream using a H264 Codec. On the client side the only requirement is an installed flashplayer at least at Version 9. The internet bandwidth stays the same as in ADSL-TV, 1.2Mbit/s. Because of running subscriptions and increased CPU-Usage we had to implement it downwardly compatible with the WMV Stream.

2. Technical Facts

2.1 Server side

2.1.1 Streaming-Encoders

Streaming Encoders are documented in our Wiki. See http://wiki.netstream.ch/wiki/index.php/Nello.tv_-_yaeps for additional information.

2.1.2 Web Servers

- 1 LoadBalancer
- 3 Webservers running Apache 2 with PHP 5.2.6-2
 - u0203
 - u0140
 - u0141
- <http://wiki.netstream.ch/wiki/index.php/Nello/Web-Cluster>

2.1.3 Database Server

Actually the goal was also a Database Cluster. Because of several issues it was unfortunately not implemented in the productive system. We have decided to run a Single-Server without redundancy.

- Server: u0238
- Database Engine: Postgres
- Additional Information (Mainly Cluster): <http://wiki.netstream.ch/wiki/index.php/Nello/DB-Cluster>

2.2 Client side

2.2.1 Browser Requirements

- At least: Internet Explorer 6.0, Firefox 2.0
- JavaScript enabled
- Screen Resolution at least 1024x768

2.2.2 Flash

- We assume the Flash Plugin at least at Version 9.x.

- ☑ http://wiki.netstream.ch/wiki/index.php/Nello.tv_-_FlashClient

3. Data Structure (Web Application)

3.1 General

- ☑ Path on Filesystem: /home/server/www/nello/
- ☑ Must be «Whilelabel»-able

3.2 Main-Folders

3.2.1 Root-Folder

Contains index file (which redirects to /live folder) and a small «status.php» Interface-Script for Munin.

3.2.2 Admin

Contains the whole Backend. This Folder cannot be accessed from external IP Adresses what guarantees additional security.

3.2.3 Change

Static Splashpage with information for old ADSL Users.

3.2.4 Cron

Files which are called by a cronjob can here be found.

3.2.5 Flashgateway

Interface between the Flashapplication and the Webapplication. At the Moment it contains only a simple Load balancer for the Streaming Servers used on Stream initialising.

3.2.6 Images

TV-Guide images and other general images

3.2.7 Includes

This folder contains the PHP function and classes.

See Chapter 3.3 Folder «Includes»

3.2.8 Live

Its the Main folder of the Project, explained in an own chapter.

See Chapter 3.4 Folder «Live» and Subfolders

3.2.9 Payment

Internal Payment Scripts Interface

3.2.10 Smarty

Folder of the Template-Engine System Smarty at Version 2.6.19

3.2.11 Templates

All HTML-Templates used by Smarty are stored here in «white-labelled» folder structure.

3.2.12 Themes

Contains in «white-labelled» folder structure:

- Flashapplications
- Stylesheets (and Minify Script)
- JavaScripts (and Minify Script)
- Images

3.3 Folder «Includes»

3.3.1 «init.php» and «close.php»

These two scripts are called on every Page load like a Header and Footer.

3.3.2 Classes

All Classes are located here.

- Captcha
- Database
- Mail
- Open-Flash-Chart (Backend)
- XML to Array

3.3.3 Config

Contains the Database-connection Parameters and every Used Link as a Constant trailed by «URL_».

3.3.4 Functions

Contains a 3000-Line file filled with functions and the Custom Session Handler Script that Stores the PHP-Session in the Database.

3.4 Folder «Live» and Subfolders

3.4.1 «Account»

Contains all Userprofile-depending pages like Register, Login, Settings and so on.

3.4.2 «Guide»

The TV-Guide Related Pages are in here.

3.4.3 «Now»

Now on TV - Part

3.4.4 «Scripts»

AJAX-Interface Scripts like Favourites or Form-Validation

3.4.5 «Service»

FAQ, Contact Form, AGB

3.4.6 «Shop»

Shop-Subapplication

3.4.7 «Watch»

Watch Window Frame for Flashplayer or Windows-Media

3.5 Special directories

- /tmp** Minify Cache directory, needs write Access, See 7.5 JavaScript (and CSS) CompressingJavaScript (and CSS) Compressing
- smarty/templates_c** Smarty Converted Templates, needs write access!

- ☑ **smarty/cache** Smarty Cache files (not active yet), needs write access!

4. Database

4.1 General

First, we tried to create a Relational Database. Because of we had no experience with Postgres, we did not hard-wire the relations in the database, so there is actually **no logic and integrity** in the DBMS! (Except of the tables with trailing «yaeps¹_»)

Most of the tables contain a field called `date_add` and `date_modif`, which are filled by the trigger `update_dates()`.

4.2 Users, Groups and Permissions

In this chapter only Project-related roles are explained.

4.2.1 Frontend-Group (aka «restricted»)

User: `nello_web`

This Group has only read access on the most tables.

4.2.2 Backend-Group (aka «backend»)

User: `nello_backend`

This user has more permissions than `nello_web`, because they're needed in backend.

4.2.3 Administrative/Other Users

`epg_import`: Used to fill the EPG-tables (channels, main, person), see ERM

`phpsessions`: Used to clean up the table `php_sessions`

`postgres`: Superuser to administrate the DB

4.3 Transactions

Transactions are used in two cases:

- ☑ The whole yaeps-Part
- ☑ The Database-based PHP-Session-Handler (Table `php_sessions`)

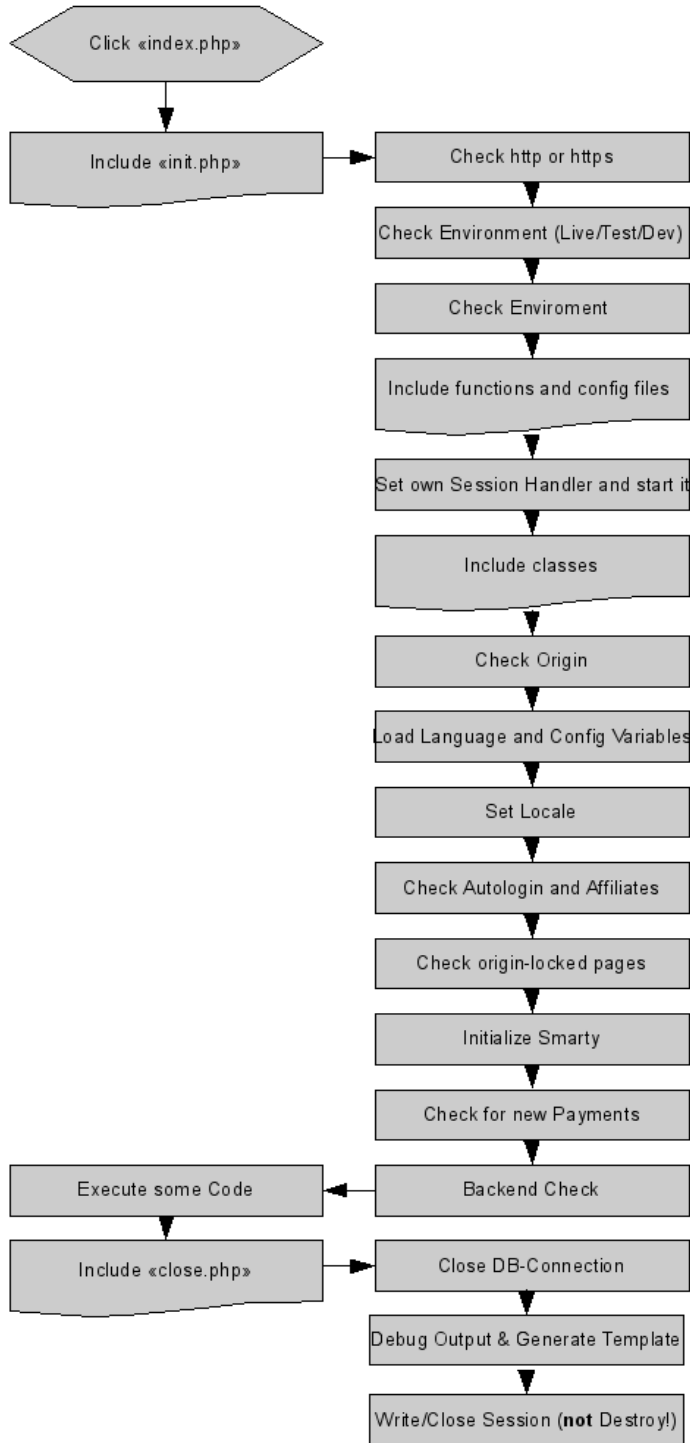
1 Yet Another Streaming Process Starter

4.4 Entity Relationship Model

See 12.Attachments

5. Sample Page Hit Sequence

To get a short impression how the includes are cascaded and what code is always executed see the following example of a Page-View.



5.1 Additions

5.1.1 Http / Https Decision

The Shop area has forced https because of security reasons. Other areas can not be viewed with https.

5.1.2 Environment Check

We do every time an environment check to load different configurations with the same piece of code.

5.1.3 Own Session Handler

We are forced to save the session data in the database because the load balancer probably switches the webserver after 5 minutes of inactivity. The session time-out is actually at 30 minutes.

We used a script from Jon Parise² at Version 2.1.

5.1.4 Origin Check

Because of legal restrictions we have to check the origin of the visitor and check if his country is in our whitelist. We use the ripe³ database as source and make a copy of the result to unload their database and prevent getting banned.

We use the following ripe parameters:

```
|whois -r -T inetnum -G [IP-Adress]
```

5.1.5 Language and Config variables

Language and Config variables are loaded from the database at script start and are saved in the Session. Config variables have the Prefix «c_» and language variables «|_».

5.1.6 Affiliates

Because the marketing wants to know how many new users an advertisement campaign attracts, every ad-link has attached a GET-Parameter which is evaluated here.

² Jon Parise <jon@php.net> http://www.csh.rit.edu/~jon/projects/pgsql_session_handler/

³ <http://www.ripe.net/db/whois.html>

6. Core-Functions

6.1 TV-Guide

6.1.1 General Introduction

The TV Guide is based on raw data from TV-Star⁴. A Perl-Script fills periodic the tables «channels», «main» and «person». Out of this data the whole Guide is generated. Every Broadcast has an Unique Broadcast ID (aka BID) as identifier.

6.1.2 Overview View

Main Problem on this Page is that the Guide is foldable (Morning , Afternoon, Evening). Because of this we have to split the daily program of a channel into these three parts. This is important because its the only possibility for Smarty to generate an output like this. For every broadcast there's a Link to the Detail View.

6.1.3 Detail View

The Detail View of a Broadcast does open in a «Thickbox⁵». There is a possibility to create a reminder which sends an E-Mail to your adress at begin or you can recommend the broadcast to someone (which also generates an email). It's important that search engines can index these sites because they generate many keywords.

6.1.4 Now

An overview with a progress bar of the current broadcast and a preview of the following broadcast. There's also the possibility to show the Detail View of a Broadcast.

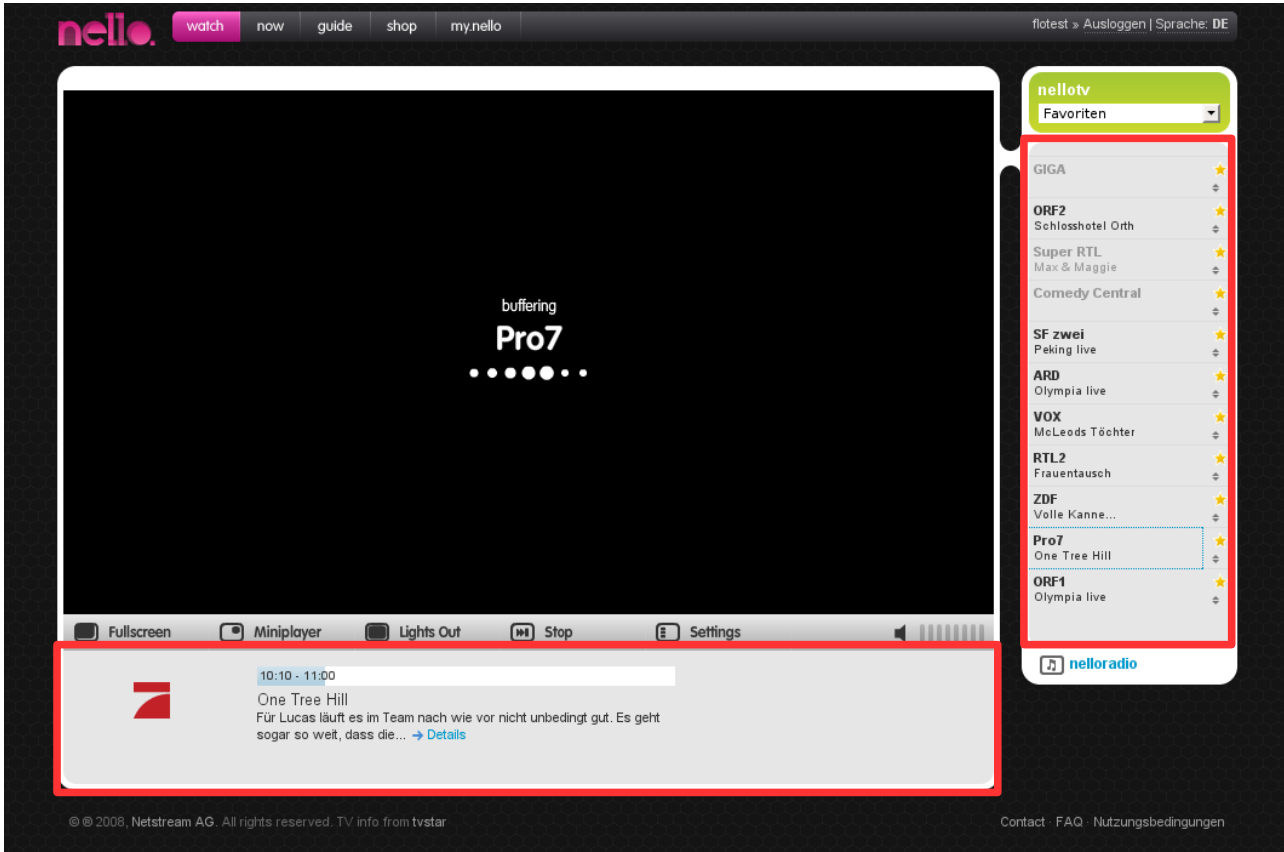
6.1.5 Favourite Channels

Every Channel can be added to favourites by clicking on the Star next to the Channel Name. Now the channel appears under «Favourites» and every user can create his personal categorie. Additional an individual channel sequence can be saved using JS (See 7.2.4 TableDND).

4 <http://www.tvstar.ch>

5 <http://jquery.com/demo/thickbox/>

6.2 Watch TV



Picture 1: Red-Framed Content is loaded by AJAX

6.2.1 First Pageload

- User has chosen WMV-Player in Settings? → Redirect to WMV-Player
- User has at least Flash Version 9? → Else Update Flash Player
- Embed the Flashplayer-Application with corresponding Settings (e.g. Protocol)
- Load the Channel list (Picture on Page 13, Red Frame on the right) as a Loop, JS Function getChannels()
- Update the Users Flashplayer-Version in our Database

6.2.2 Channel Switch

- «OnClick» any channel on the right
- Call JavaScript function showChannel() which transmit the following

Values to the Flashplayer-Application

- Channel-ID (e.g. 123)
- Profile-ID (e.g. 1)
- Current Unix-Timestamp (e.g. 1219136527)
- Lifetime in seconds (e.g. 123123)
- Hash (Blowfish Crypted Hash with secret Salt, prevents Parameter manipulation)
- Streamname (e.g. SF1)
- Application (e.g. Nello)
- User-ID (e.g. 1234)
- Call JavaScript Function `getEpgInfo()` which refreshs the EPG Information (See Picture on Page 13) as a Loop
- Set the Page Anchor to «Channel-ID»

If a Channel has Multiple Languages (e.g. SF2), all the Parameters occur 2 Times in the same order, the 2nd time the parameters belong to the child channel.

6.2.3 Categorie Switch

- JavaScript Function `getChannels()` gets called as a Loop and Channellist does refresh with the new Content

6.2.4 Nello Radio

- New Pop-up opens
- Switching the Channel works like in the Main-Window (See 6.2.2 Channel Switch)

7. JavaScript / JQuery

7.1 The JQuery Library

We have decided for the JQuery JS-Library because:

- It's quite fast and small
- It's popular
- It has a good Documentation and many DOM-Selectors
- It's easy to use and has a short notation

- ☑ It has no predefined animations and design elements

7.2 Used JQuery Plug ins

7.2.1 Thickbox

Thickbox is used at many parts of the website. It creates nice popup Windows and does not disturb search engines.

See <http://jquery.com/demo/thickbox/>

7.2.2 Ajaxcontent

Used under «My nello» to display the subcategories without reloading the whole page.

See <http://www.andreacfm.com/index.cfm/jquery-plug-ins/ajax-content>

7.2.3 Delegate, Form, Validate and MaskedInput

We used these four plugins for nice looking forms and some-validation at severnal forms.

7.2.4 TableDND

Used for dynamic ordering of the favourite channels.

See <http://www.isocra.com/2008/02/table-drag-and-drop-jquery-plugin/>

7.2.5 TableSorter and datepicker

These two plugins are used on nearly every page in the backend. It allows to sort tables on the fly with javascript and a little datepicker.

See <http://tablesorter.com> and

<http://www.kelvinluck.com/assets/jquery/datePicker/v2/demo/>

7.2.6 Checkboxes

It's only used at the backend and allows some more Checkboxes manipulation.

See <http://www.texotela.co.uk/code/jquery/checkboxes/>

7.3 Other JS Stuff

7.3.1 SWFobject

Used for embedding the Flash snippets and the Flashplayer. This Script is very popular and it makes embedding Flash more compatible and comfortable.

<http://code.google.com/p/swfobject/>

7.3.2 Coda-Slider

This is only used at the «Tour» page to generate a nice looking JS-slideshow.

See <http://www.ndoherty.com/coda-slider>

7.4 Self written JS functions

Can be found in these two files:

- flashplayer.js
- functions.js

7.5 JavaScript (and CSS) Compressing

We used the library called «minify» to compress the javascript and css files. They are only active on the productive servers because its not funny to debug a compressed script.

Minify writes a Cache File at `sys_get_temp_dir()` which equals to `/tmp` at our configuration.

See <http://code.google.com/p/minify/>

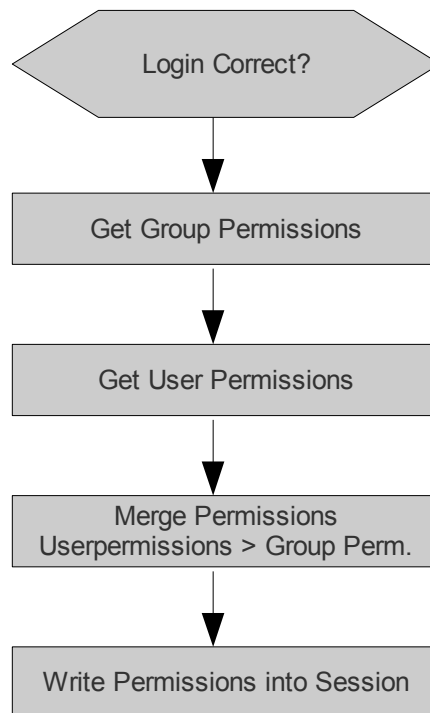
8. Backend

8.1 Security Concept

- Directory «admin» is only accessible from internal network
- User Passwords have at lease 8 characters, the login name should be `firstname.lastname`

8.2 Authorisation

The following schema should give you a short view how the authorisation in the backend works.



8.3 Backend Pages

8.3.1 Edit Textpattern

Used to modify the Language Pattern used all over the page. Changes don't happen immediately, they are delayed until a new PHP-session gets started. (You can debug it with GET-Parameter «killbill»)

8.3.2 Show Log Entries

All Log-Entries are displayed here. There are several filter possibilities.

8.3.3 Show Users

All Users are displayed here. There are several filter possibilities. You can also edit userinformations and settings here.

8.3.4 Show Sessions

Use this page to see, who and what is being watched on nello right now.

8.3.5 Show Orders

Display the orders on nello and use several filter possibilities.

8.3.6 Vouchers

Add Vouchers or display already generated vouchers and see who used them.

8.3.7 Payments

Display and Filter Payments

8.3.8 Show Admin-Log

Every action which modifies data in the Backend is logged and can be displayed here.

8.3.9 Show Statistics

You can display several graphical statistics which are realized with «Open Flash Chart».

See <http://teethgrinder.co.uk/open-flash-chart/>

8.3.10 Show Channelstatus

Display the status of the Channels and if required do restart them. This Action needs some minutes because «yaeps» checks periodical the «kill-flag».

9. Miscellaneous

9.1 Revision Control (Git)

9.1.1 Live-Branch

This branch should **always** equal to the version running on the Live-Server. You can synchronise this branch without any doubt.

9.1.2 Master-Branch

This branch mostly contains development files which are not ready for publishing. Please do ask the developer who has pushed last into this branch before you merge it into the Live-Branch.

9.2 Publishing (universal_sync.sh)

- Commit and push you changes into the master branch
- Merge the Master branch with the Live Branch
- Run «universal_sync.sh live» and confirm security question
- Check if all is working fine on the Live system

Push the Live Branch

As Code:

```
# git-fetch
# git-merge origin/master
# git-commit -a -m "Description"
# git-push origin master
# git-fetch
# git-checkout live
# git-merge master
# echo "j" | universal_sync.sh live
# git-push
```

9.3 How to «clone database»

Create database «foo»

Run following command in a shell

```
pg_dump -h 62.65.130.179 nello -U postgres | psql -h 62.65.130.179 -
d foo -U postgres
```

Get a coffee and afterwards check if the import was successful

9.4 How to create «auto-date trigger»

Execute the following code as a SQL command:

```
CREATE OR REPLACE FUNCTION update_dates()
  RETURNS trigger AS
$BODY$
BEGIN
IF (TG_OP = 'INSERT') THEN
NEW.date_add = current_timestamp;
RETURN NEW;
ELSIF (TG_OP = 'UPDATE') THEN
NEW.date_modif = current_timestamp;
RETURN NEW;
END IF;
END;
$BODY$
LANGUAGE 'plpgsql' VOLATILE
COST 100;
```

The Trigger should now be executable by the DB.

Now its time to alter your table (e.g. ex my_table) and add the required fields:

```
ALTER TABLE my_table ADD COLUMN date_add timestamp;
ALTER TABLE my_table ADD COLUMN date_modif timestamp;
```

After that you can bind the trigger to the table:

```
CREATE TRIGGER my_trigger_any_name BEFORE INSERT OR UPDATE ON my_table FOR
EACH ROW EXECUTE PROCEDURE update_dates();
```

10. Dictionary

Expression	Explanation
EPG	Electronic Program Guide
ERM	Entity Relationship Model
Git	Sort of Revision Control System
H264	H.264 is a standard for video compression. It is also known as MPEG-4 Part 10, or MPEG-4 AVC (for Advanced Video Coding).
JS	JavaScript
White-Label	We understand under Whitelabeling that we can sell the product to another partner company and just can replace the templates without copying the program code.
WMV	Windows Media Video
yaeps	Yet Another Streaming Process Starter

11. Version / Revision

Date	Revision
27/08/08	Finished without «Shop»-Part
19/08/08	First Draft
23/01/09	Updated Sync Script (universal_sync.sh)

12. Attachments

- Entity Relationship Model (A2 Size)
- Sunrise Desktop TV User concept
- Site structure Draft (does not match real situation)