

Sem vložte zadání Vaší práce.



ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE  
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ  
KATEDRA SOFTWAREVÉHO INŽENÝRSTVÍ



Bakalářská práce

# Flexibilní logování pro embedded Linuxové systémy

*David Vavříčka*

Vedoucí práce: Ing. Matěj Laitl

7. dubna 2016



---

## Poděkování

Doplňte, máte-li komu a za co děkovat. V opačném případě úplně odstráňte tento příkaz.



---

## Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 7. dubna 2016

.....

České vysoké učení technické v Praze  
Fakulta informačních technologií

© 2016 David Vavříčka. Všechna práva vyhrazena.

*Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí, je nezbytný souhlas autora.*

### **Odkaz na tuto práci**

Vavříčka, David. *Flexibilní logování pro embedded Linuxové systémy*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.



---

## Abstrakt

Doplňte

**Klíčová slova** logování, vestavěné systémy, logovací démoni, Linux, Rsyslog

---

## Abstract

Sem doplňte ekvivalent abstraktu Vaší práce v angličtině.

**Keywords** logging, embedded systems, logging daemons, Linux, Rsyslog



---

# Obsah

<b>Úvod</b>	<b>1</b>
<b>1 Technické požadavky</b>	<b>3</b>
1.1 Základní technické požadavky . . . . .	3
1.2 Rozšířené technické požadavky . . . . .	4
<b>2 Analýza a návrh</b>	<b>5</b>
2.1 Současné řešení . . . . .	6
2.2 Vývoj pro STB . . . . .	7
2.3 Nové řešení . . . . .	8
2.4 Srovnání logovacích démonů . . . . .	9
2.5 Vzdálená konfigurace . . . . .	10
2.6 Postprocessign zpráv . . . . .	11
<b>3 Realizace</b>	<b>13</b>
3.1 Rsyslog . . . . .	13
3.2 Shell API . . . . .	13
3.3 dmd . . . . .	13
<b>4 Testování</b>	<b>15</b>
<b>Závěr</b>	<b>17</b>
<b>Literatura</b>	<b>19</b>
<b>A Seznam použitých zkratk</b>	<b>21</b>
<b>B Obsah přiloženého CD</b>	<b>23</b>



---

## Seznam obrázků

2.1	STB . . . . .	5
2.2	STB-new . . . . .	8



---

# Seznam tabulek

2.1	Převodní tabulka . . . . .	11
-----	----------------------------	----





---

# Úvod



# Technické požadavky

Cílem je upravit logovací řešení pro set-top box EKT DID7006mTF [1] tak, aby splňovalo technické požadavky popsané v této kapitole. Řešení musí fungovat a být otestováno na zmíněném modelu set-top boxu a pokud možno by mělo být přenositelné i na jiné typy set-top boxů. Požadavky jsou rozděleny na základní a rozšířené. Rozšířené požadavky není nutno implementovat.

## 1.1 Základní technické požadavky

### 1.1.1 Snížení objemu logů

Je žádoucí umožnit snížit objem zasílaných logů z důvodu přílišného zatížení sítě a serverových disků. A to tak, že zadavatel bude schopen nadefinovat pro každou komponentu úroveň severity zpráv, od které mají být posílány na server. Výchozí nastavení dodá zadavatel.

### 1.1.2 Vzdálená konfigurace

Technické řešení musí být schopno za běhu pomocí SHELL-ového API měnit minimální severitu zpráv pro jednotlivé komponenty a dále toto API musí umožnit nastavit výchozí severitu, která se použije pro komponenty ji nemají explicitně nastavenou. Takto změněné nastavení musí být perzistentní i po restartu STB. Výchozí nastavení se obnoví až po factory resetu. API navrhne dle své libovůle sám řešitel.

### 1.1.3 Rate-limiting odesílaných zpráv

Nové logovací řešení musí být schopné provádět rate-limiting odesílaných zpráv tak, aby nepřekročilo maximální vyhrazenou šířku pásma. Bude umožněno nastavit jak dlouhodobé tak krátkodobé limity. Naivní rate-limiting je i v existujícím řešení, řešitel navrhne výchozí nastavení nového řešení tak, aby přibližně odpovídalo současnému chování.

### 1.1.4 Formát logů

Je nutno zachovat formát logů jako ho má původní řešení, aby se jednalo o drop-in replacement bez nutnosti jakkoli měnit konfiguraci serveru, který sbírá logy od set-top boxů.

### 1.1.5 Razítkování zpráv

Každé zprávě se musí přidat textový prefix id=N, kde N monotonicky roste s každou zprávou. To slouží pro detekci ztracených zpráv. Id přeteče po 32 nebo 64 bitech, to záleží na rozhodnutí řešitele. Po rebootu STB id znovu začíná od 1.

### 1.1.6 Post-processing zpráv

Zadavatel má pouze částečnou kontrolu nad zprávami generovanými aplikacemi na set-top boxu, například nedokáže ve všech případech eliminovat dlouhé prefixy u zpráv. Je proto nutno takové prefixy rozpoznat a vhodně odfiltrovat před odesláním. Ze stejného důvodu mají některé zprávy nevhodně vyplněnou severitu a položku app-name. Jejich správné hodnoty jsou uloženy v textu zprávy, jejíž formát je pro jednotlivé skupiny zpráv konstantní. Řešení bude schopné tyto údaje z těla zprávy extrahovat a nahradit jimi původní metadata. Tato pravidla musí být možné definovat a měnit bez nutnosti nového sestavení softwaru. Řešitel vytvoří pro ukázkou 3 pravidla, která budou sloužit zadavateli jako šablony pro možná budoucí filtrovací pravidla. Tato pravidla jsou popsána v kapitole analýza.

## 1.2 Rozšířené technické požadavky

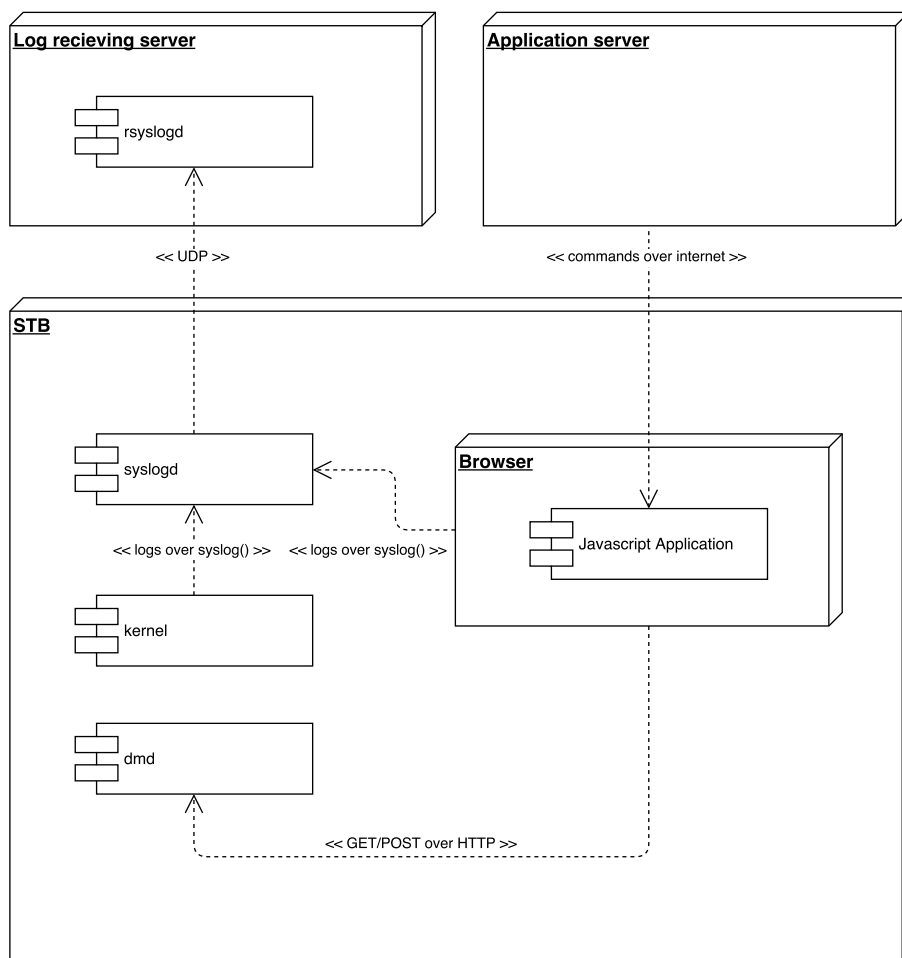
### 1.2.1 Komprese zpráv

Bylo by vhodné zvážit pro a proti komprese zpráv. Vyplatí se ušetřená přenesená data oproti režiji spojené s kompresí a dekompresí zpráv?

### 1.2.2 Integrace do existující servisní C++ komponenty

Zadavatel na STB provozuje malého démona dmd napsaného v C++, který mimo jiné obsahuje minimalistický HTTP webserver. Dále v browseru běží Javascript aplikace (nangu.TV portál), která pomocí messagingu komunikuje s centrálním serverem. Tato Javascript aplikace ovšem nemůže přímo používat Shell API. Požadavkem je rozšířit C++ komponentu dmd tak, aby umožnila Javascript aplikaci řídit konfiguraci logování (viz bod Vzdálená konfigurace).

## Analýza a návrh



Obrázek 2.1: Diagram nasazení původního řešení STB

### 2.1 Současné řešení

V této kapitole jsou uvedeny a popsány jednotlivé pro tuto práci důležité komponenty současného řešení.

#### 2.1.1 Runtime prostředí

Na STB běží OS GNU/Linux s Busybox sadou aplikací, která nahrazuje standardní GNU shellové utility. Busybox aplikace jsou díky své malé velikosti vhodnější pro embedded zařízení. Na druhou stranu neobsahují všechny funkcionality a možnosti nastavení.

##### Browser

Browser je jednoduchý internetový prohlížeč dodaný výrobcem STB. V něm běží v Javascriptu zadavatelem napsaná aplikace poskytující uživatelské rozhraní. Mimo to také naslouchá zprávám z aplikačního serveru, na jejichž základě může provádět akce, a to včetně generování POST či GET requestu pro démon dmd. Tato javascriptová aplikace spolu s dalším podpogramem nesoucím název player generují velké množství logů. Tyto logy nepoužívají standardizovanou syslog množinu severit, ale svou vlastní s ní nekompatibilní. Logovací démon ovšem očekává zprávy právě podle syslog standardu. Implementaci browseru ani playeru nemůže zadavatel měnit a proto je třeba nekompatibilitu vyřešit pomocí logovacího démonu.

##### Servisní komponenta dmd

Dmd je v C++ napsaná servisní komponenta, která zprostředkovává komunikaci mezi Browserem a shellovým prostředím pomocí minimalistického HTTP serveru. Umožňuje JavaScript aplikaci na STB provádět operace, které jsou dostupné jen z shellového API.

##### BusyBox Syslogd

Jedná se o minimalistický logovací démon, který je podrobněji popsán v následující kapitole „Srovnání logovacích démonů“. Zadavatel mu pro své potřeby navíc doimplementoval číslování jednotlivých zpráv pro rozpoznání výpadku a za druhé naivní rate-limiting zpráv. Ten je naivní z důvodu, že používá statický buffer na počet zpráv, který nebere ohled na jejich velikost a hlavně bufferované zprávy se neodešlou samy, ale jen s nově příchozími zprávami. Za použití tohoto démonu není možné splnit některé stěžejní technické požadavky, jako například „Snížení objemu logů“ nebo „Post processing zpráv“. Proto bude nutné ho nahradit nějakým vyspělejším démonem. Tím se ale zabývá až další kapitola „Nové řešení“.

### 2.1.2 Aplikační server

Aplikační server se stará o business logiku, například poskytuje uživateli práva přehrát daný kanál nebo film, komunikuje s javascriptovou aplikací, která se vůči němu autorizuje a podobně. AS není možné v této práci jakkoli měnit. Je nutné zachovat kompatibilitu.

### 2.1.3 Servery pro sběr logů

Zadavatel provozuje cluster serverů sbírajících logy od statisíců STB. Běží na nich rsyslog nakonfigurovaný pro co nejlepší výkonost.

## 2.2 Vývoj pro STB

### 2.2.1 Buildovací systém Gu

Gu je buildovací systém pro linuxová embedded zařízení, který si klade za cíl zjednodušit a urychlit buildovací proces. Gu využívá balíčkovací systém pacman (převzatý z Arch Linuxu) a nástroj scratchbox2 sloužící k zjednodušení cross-kompilace. Tyto 2 nástroje jsou skryty v konzolovém příkazu „gu“, kterým se celé Gu ovládá.

### 2.2.2 PKGBUILD

Jedná se o shellový script obsahující informace potřebné pro build jednotlivých aplikací systémem Gu. Níže je pro představu uveden vzorový script:

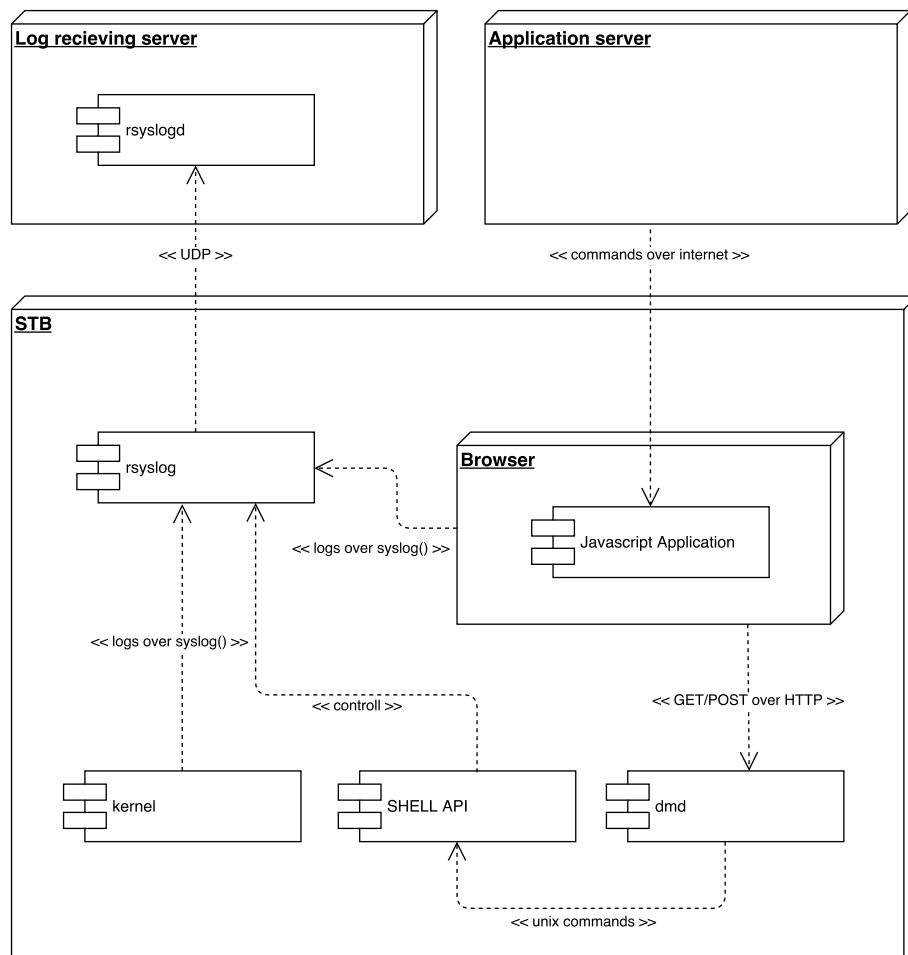
```
pkgname=libee
pkgver=0.4.1
pkgrel=1
arch=('armv7h' 'armv7sp')
depends=('libestr')

build() {
    cd ${pkgname}-${pkgver}
    PKG_CONFIG_PATH=/mnt/hdd_1/lib/pkgconfig
    ./configure --prefix=/mnt/hdd_1
    make V=1
}

package() {
    cd ${pkgname}-${pkgver}
    make install DESTDIR=${pkgdir}
}
```

Povinná pole jsou „pkgname“, označující název balíčku, „pkgver“ pro určení jeho verze, „pkgrel“ sloužící k označení verze PKGBUILD scriptu a „arch“ definující cílové architektury. Script dále obsahuje funkce zajišťující build a instalaci balíčku. Každý PKGBUILD musí obsahovat minimálně funkci package(), která nainstaluje soubory do cílové složky. Užitečnou funkcí je funkce build() sloužící k přípravě a kompilaci zdrojových kódů před samotnou jejich instalací.

## 2.3 Nové řešení



Obrázek 2.2: Diagram nasazení nového řešení STB

Prvně je nutno zvážit, zda problém řešit na straně serveru nebo set-top boxu. Vhodnou konfigurací logovacího démona na straně serveru, který by nepotřebné zprávy závčas rozpoznal, zahodil a dále nezpracovával bychom splnili požadavek na snížení zátěže serverových disků. Přetížení sítě se takto vyřešit ale nedá a proto toto řešení zavrhuji. Je tedy nutno problém řešit na straně set-top boxu kde původní řešení je postaveno na busy-box syslogd. Nabízí se možnost upravit fungování tím způsobem, aby se logy s nízkou severitou už na set-top boxu zahazovaly a pouze v případě potřeby bylo umožněné na dálku změnit konfiguraci démona tak, aby se povolilo logování pro logy s nastavenou danou komponentou a severitou. To vše přes SHELL-ové API. Součástí zadání je ale i implementovat škrcení zpráv, aby nedocházelo k zahlcení linky. Takovou možnost prostý syslogd neposkytuje a je proto nutno zvážit napsání



vlastního démona či nasazení jiného, vyspělejšího logovacího démona.

## 2.4 Srovnání logovacích démonů

Démon v UNIXovém světě je označení pro takový proces, který oproti běžným procesům neinteraguje přímo s uživatelem, ale běží na pozadí operačního systému a funguje samostatně. Účelem logovacího démona je sběr logů od ostatních procesů, které následně v závislosti na jeho konfiguraci dokáže filtrovat a ukládat na disk či odesílat na požadovaný vzdálený server.

V této kapitole zmíním a popíši vybrané logovací demony a v závěru kapitoly je porovnáám.

### BusyBox Syslogd

Tato logovací utilita se skládá ze dvou démonu, jmenovitě z Klogd, který zachytává logy z kernelu a předává je ke zpracování Syslogdemonu. Syslogd pak zachytává i všechny zbylé logy a dále s nimi nakládá. Má však velice omezenou funkcionalitu. Dokáže pouze logy lokálně ukládat, přeposílat je dále po síti, zahazovat duplikáty, rotovat logy v závislosti na velikosti a filtrovat zprávy podle omezených kritérií.

### Syslog-ng

Flexibilní logovací démon zaměřený na centralizované a zabezpečené logování. Má široké možnosti nastavení a poskytuje obrovské množství funkcionalit. Takže jeho vhodným nakonfigurováním se dají snadno splnit všechny vytyčené technické požadavky až na požadavek pro možnost vzdálené změny konfigurace. Je nutno ale zmínit, že pokročilé funkce jako například šifrování zpráv, bufferování nebo message-rate kontrola jsou dostupné pouze v komerční closed-source verzi.

### Rsyslog

Výčet funkcionalit Rsyslogu je ještě obsáhlejší než u Syslog-ng. Technické požadavky se s jeho použitím tedy také dají splnit všechny, kromě vzdálené změny konfigurace. Oproti Syslog-ng je Rsyslog kompletně zdarma a open-source. Navíc není jen logovacím démonem, ale i analyzérem logů. Dokáže logy podle obsahu zprávy měnit, třídit a jinak s nimi nakládat. Že je Rsyslog vyspělý a kvalitní program dokazuje fakt, že je defaultním logovacím démonem na spoustě linuxových distribucích, jmenovitě například v Ubuntu. Jeho slabiny shledávám v nedostatečné dokumentaci a ve specifických případech v neefektivním analyzování logů mající za následek (obzvláště na embedded zařízení s pomalým ARM procesorem) rychlostní deficit. Jeho vývoj obstarává

z velké většiny pouze jeden člověk, jeho původní tvůrce Rainer Gerhards. A v jednom člověku není snadné dovést tak rozsáhlý projekt k dokonalosti.

### Porovnání výše zmíněných logovacích utilit

Pouhým nasazením jakéhokoli známého logovacího démonu není možné splnit všechny vytyčené technické požadavky. V případě ponechání původního Busy-Box syslogd démonu by pro splnění technických požadavků bylo nutno doimplementovat tolik funkcionalit, že by to výrazně přesahovalo rozsah bakalářské práce. Výhodněji se jeví nasadit pokročilý logovací démon jako je Syslog-ng či Rsyslog. Oba totiž poskytují námi požadované funkcionality. Syslog-ng však většinu z nich poskytuje pouze v placené closed-source verzi a proto jsem se rozhodl pro Rsyslog.

## 2.5 Vzdálená konfigurace

Rsyslog při svém zapnutí čte konfigurační soubor `rsyslog.conf`, který za jeho běhu není možné měnit. Je pro to nutné napsat SHELL-ové API, které umožní na dálku přenastavit tento konfigurační soubor a restartovat rsyslog.

### Shellové API - 1. způsob

API ve formě scriptu, který na vstupu přijímá jednotlivé názvy komponent a jemu příslušící maximální povolenou severitu, na jejichž základě překonfiguruje rsyslog.

```
set_log_verbosity.sh [component] [severity] [component2] [severity] ...
```

### Shellové API - 2. způsob

API, které čte z vlastního konfiguračního souboru seznam komponent a jim příslušící maximální povolené severity.

```
/etc/logging.conf
```

```
component1 = DEBUG
componentXY = INFO
...
DEFAULT   = INFO
```

### Shellové API - porovnání

Jako vhodnější se jeví 1. způsob, jelikož s jeho použitím odpadá starost o další konfigurační soubor a navíc je jednodušší a rychlejší na obsluhu.

## 2.6 Postprocessign zpráv

### 2.6.1 Smazání podřetězce z těla syslog zprávy

Zprávy s nastavenou severitou INFO a komponentou sld\_br je třeba změnit podle následujícího vzoru.

#### Originální zpráva:

```
2016-02-18T14:05:24+01:00 cc-b8-f1-00-6f-07 sld_br: id=559
:[stbhal.cpp:debug:520]: INFO: [94mDEBUG: InformationService:
Reading 'nangu.video.forcedScart': false[0m
```

#### Výsledná zpráva:

```
2016-02-18T14:05:24+01:00 cc-b8-f1-00-6f-07 sld_br: id=559
[94mDEBUG: InformationService: Reading 'nangu.video.forcedScart':
false[0m
```

### 2.6.2 Zahazování zpráv podle typu komponenty

Je třeba dokázat rozpoznat a zahodit zprávy podle zadané komponenty a obsahující určitý podřetězec.

### 2.6.3 Převod severit

Aplikace na STB používají pro logování jinou množinu severit, než s jakými pracuje Rsyslog. Zadavatel proto požaduje změnit severity zpráv podle následující tabulky.

Tabulka 2.1: Převodní tabulka

Portal	Syslog
ERROR	ERR
WARN	WARN
INFO	NOTICE
DEBUG	INFO
TRACE	DEBUG



---

## Realizace

### 3.1 Rsyslog

#### 3.1.1 build

Rsyslog bylo nutno zkompilevat ze zdrojových kódů včetně mnoha závislostí.

#### 3.1.2 konfigurace

TODO jednotlivy konfigurace pro pravidla, dodat ukázky configů.. krome popsaní rsyslog.conf i logrotate etc all scripty nebo možná do jiné podsektce toto

#### 3.1.3 modul mmstrdel

#### 3.1.4 modul mmsevalbab

### 3.2 Shell API

TODO code

### 3.3 dmd



# Testování





---

## **Závěr**



---

## Literatura

- [1] *EKT DID7006*. Dostupné z: <http://exploredoc.com/doc/3174828/model-did7006-high-definition-ott-stb>



## Seznam použitých zkratek

**API** Application Programming Interface

**ASH** Almquist shell

**dmd** Download manager daemon

**STB** Set-top box



## Obsah přiloženého CD

	readme.txt.....	stručný popis obsahu CD
	exe .....	adresář se spustitelnou formou implementace
	src	
	impl.....	zdrojové kódy implementace
	thesis .....	zdrojová forma práce ve formátu L <sup>A</sup> T <sub>E</sub> X
	text .....	text práce
	thesis.pdf .....	text práce ve formátu PDF
	BP_Vavricka_David_2016.pdf .....	text práce ve formátu PDF