Mendelova univerzita v Brně
Provozně ekonomická fakulta

Monitoring diskových polí na platformě Centreon

Bakalářská práce

Vedoucí práce:
Ing. Martin Pokorný, Ph.D.

Roman Opluštil

Brno, 2016

Tímto bych chtěl poděkovat zejména vedoucímu této bakalářské práce panu Ing. Martinovi Pokornému, Ph.D. za kvalitní vedení bakalářské práce, poskytnuté rady, ochotu a čas strávený při konzultacích. Dále bych chtěl poděkovat firmě CDC Data, s. r. o. za poskytnuté zadání bakalářské práce a prostředky pro její řešení, panu Ing. Tomáši Holomkovi za ochotu vyjít vstříc při zpracovávání řešení a v neposlední řadě svým přátelům a rodině za podporu při vzdělávání.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Monitoring diskových polí na platformě Centreon**
vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*
Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.
Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

Brno, 19. 5. 2016 .....

Abstract

Opluštil, R. Disk arrays monitoring based on Centreon platform. Bachelor thesis. Brno, 2016

This bachelor thesis is concerned with plugin development, which allows to monitor states of physical components of specific disk array set and its integration into monitoring system Centreon. The access to these devices is provided by SNMP. The thesis contains analysis of specific MIB tables, proposal and implementation of plugin in programming language Perl and description of installation and configuration of monitoring system Centreon.

Keywords

Centreon, monitoring, disk array monitoring, network monitoring, SNMP, Perl, plugin

Abstrakt

Opluštil, R. Monitoring diskových polí na platformě Centreon. Bakalářská práce. Brno, 2016

Tato bakalářská práce se zabývá vytvořením pluginu, umožňující monitorování stavů fyzických komponent dané množiny diskových polí a jeho integraci do monitorovacího systému Centreon. Přístup k zařízení je řešen použitím SNMP. V práci je obsažen rozbor jednotlivých MIB tabulek, návrh a implementace pluginu řešená v programovacím jazyku Perl a dále popis instalace a konfigurace monitorovacího systému Centreon.

Klíčová slova

Centreon, monitoring, monitoring diskových polí, síťový monitoring, SNMP, Perl, plugin

Obsah

Seznam obrázků

1 Úvod

Monitoring počítačových sítí a prvků do těchto sítí připojených je velice mocným nástrojem pro správu a kontrolu. Použití monitorovacích systémů umožňuje shromažďovat informace o stavu sítě a zařízení na jednom místě, aniž bychom museli fyzicky kontrolovat jednotlivé prvky sítě zvlášť. To s sebou přináší zejména zvýšení efektivnosti řešení problémů na síti a predikci možného vývoje krizových situací jako jsou například výpadky sítě nebo přetížení zařízení připojených do sítě. Monitorovací systémy tedy slouží ke sledování sítě nebo zařízení v reálném čase. Jedná se o pravidelné periodické kontroly, které mají za úkol informovat správce o aktuálním stavu sítě nebo zařízení. Získané hodnoty z kontrol zařízení následně ukládají do vlastní databáze, čímž umožňují zkoumání vývoje sledovaných hodnot na daném zařízení ve zvoleném časovém období.

Zadání bakalářské práce bylo vytvořeno firmou CDC Data, s. r. o. (dále CDC), konkrétně jejím pracovníkem Ing. Tomášem Holomkem. Společnost CDC se profiluje jako systémový integrátor pro IT infrastrukturu s důrazem na poskytování kvalitních servisních služeb. Mezi hlavní část servisních služeb patří i centralizovaný monitoring IT infrastruktury jednotlivých klientských podniků, čímž jim pomáhá s kontrolou výkonu a funkčnosti celého systému, předcházením nepříjemných překvapení nebo výpadků systému (CDC Data, s. r. o., 2015a). Nutno zmínit, že CDC v současné době poskytla naší fakultě tři zadání bakalářských prací, které se zabývají monitoringem na platformě Centreon. Tyto práce mimo mne dále zpracovávají Tomáš Honek, který se zabývá monitoringem tiskáren na platformě Centreon a Marek Hevier, který se zabývá monitoringem serverů na platformě Centreon. Tyto tři práce mají společný základ, jímž je zmiňovaný monitorovací systém, nadále se však každý z nás zaměřuje na svoji specifickou oblast monitorované techniky. Společný základ bakalářských prací jsme s kolegy řešili v týmu, formou společných setkání. Jednalo se zejména o instalaci a seznamování se s monitorovacím systémem Centreon. Implementační a obsahovou část práce jsme posléze řešili již každý sám.

CDC v současné době využívá pro monitoring IT infrastruktury svých klientů monitorovací systém Centreon (CDC Data, s. r. o., 2015b). Momentálně je firma jediným partnerem systému Centreon v České republice (Centreon, 2015c), což firmě přináší jistou konkurenční výhodu. Z těchto důvodů bude plugin pro monitorování diskových polí vyvíjen pro systém Centreon. Návrhem řešení pro jiné systémy či podrobným zkoumáním a popisem jiných monitorovacích systémů se v práci nadále nebudu zabývat.

2 Cíl

Na základě požadavků firmy CDC Data, s. r. o. je cílem práce analýza, návrh a implementace pluginu, který detekuje typ diskového pole, posléze provede analýzu stavu zařízení, vrátí hodnoty ve formě zpracovatelné pro monitorovací systém Centreon a v případě již existujících řešení zpracovat jejich review a publikační rešerži. Po vytvoření analýzy je vypracován návrh řešení, jeho implementace a nasazení v testovací síti a provedení verifikačního testu, po kterém následuje nasazení pluginu ve spolupráci s CDC na produkčním zařízení. V závěru práce je celý projekt ekonomicky zhodnocen.

3 Analýza uživatelských požadavků

Požadovaný výstup zadavatelem bakalářské práce je plugin integrovatelný do monitorovacího systému Centreon, který bude kontrolovat a analyzovat stavy určených typů diskových polí. Konkrétně se tedy jedná o vývoj pluginu, který bude určen pro monitorování diskových polí Fujitsu Eternus DX60, DX80, DX100, DX60 S2, DX100 S2, DX100 S3, DX200 S3 a DX500 S3. Zadavatel požaduje, aby byl plugin na zmíněných typech zařízení použitelný bez zásahu do zdrojového kódu, úpravy parametrů nebo jakýchkoliv jiných změn. Další požadavek je definován souborem informací, které musí plugin z monitorovaného zařízení získat.

Jedná se o kontrolu:

- typu monitorovaného zařízení
- sériového čísla
- stavu fyzických komponent
- chybových hlášení

Uvedené informace plugin musí efektivně zpracovat a vytvořit výstup informující o stavu zařízení ve formě zpracovatelné monitorovacím systémem Centreon.

Zjištění typu monitorovaného pole je stěžejní pro další získávání dat z daného zařízení. Jelikož se MIB tabulky jednotlivých zařízení od sebe liší, lze tuto informaci využít pro výběr správné struktury MIB tabulky a následné získávání sledovaných dat ze zařízení. Ve výstupu pluginu uvedení monitorovaného typu zařízení zvyšuje uživatelovu přehlednost a snažší orientaci, zejména při souběžném monitorování více zařízení.

Sériové číslo je pro uživatele důležité z pohledu určení konkrétního monitorovaného fyzického zařízení. Ve spojení s dobře vedenou invertarizací usnadňuje lokalizaci monitorovaného zařízení.

Kontrolování stavu jednotlivých komponent zařízení je nejdůležitější část požadované funkcionality pluginu. Sledováním a analýzou stavů jednotlivých komponent lze uživatele rychle informovat o kritickém stavu konkrétní komponenty nebo její poruše. Na sledovaných zařízeních budou monitorovány následující komponenty.

Na zařízeních používající FJDARY-E60.mib a FJDARY-E101.mib se jedná o kontrolu:

- controller module (CM) – modul obsahující CPU a hlavní paměť
- channel adapter – realizuje konektivitu k hostům
- CM memory – paměť controller modulu
- CM flash-rom – paměť flash-rom controller modulu
- system capacitor unit (SCU) – elektrický dvojvrstvý kondenzátor nahrazující konvenční baterie

- controller enclosure (CE) – konstrukce udržující několik komponent jako controller moduly, zdroje napájení nebo front-end a back-end routery
- device enclosure (DE) – konstrukce udržující komponenty diskového pole
- expander – komponenta kontrolující interakci mezi controller modulem a disky
- power supply unit (PSU) – zdroj napájení
- inlet thermal sensor – vnitřní tepelný senzor
- thermal sensor – tepelný senzor
- disk – disk pro ukládání dat
- LAN port – pro připojení k síti

Na zařízeních používající FJDARY-E150.mib se jedná o kontrolu:

- controller module (CM) – modul obsahující CPU a hlavní paměť
- channel adapter – realizuje konektivitu k hostům
- CM memory – paměť controller modulu
- system capacitor unit (SCU) – elektrický dvojvrstvý kondenzátor nahrazující konvenční baterie
- controller enclosure (CE) – konstrukce udržující několik komponent jako controller moduly, zdroje napájení nebo front-end a back-end routery
- power supply unit (PSU) – zdroj napájení
- disk – disk pro ukládání dat
- LAN port – pro připojení k síti
- CM central processing unit – procesor, který je součástí CM
- boot-up and utility device (BUD) – velmi rychlé stabilní SSD disky, na nichž je uložený operační systém a konfigurace a také slouží pro dočasné ukládání dat
- CM fan – větráček ochlazující CM
- battery control unit (BCU) – kontrolní jednotka ovládající baterie
- battery unit (BTU) – baterie, sloužící pro uchování energie pro případ výpadku proudu
- CE inlet thermal sensor – vnitřní tepelný senzor CE
- CE thermal sensor – tepelný senzor CE
- CE power supply unit (CPSU) – zdroj napájení CE

- drive enclosure (DE) – konstrukce udržující komponenty jako jsou disky, zdroje napájení a I/O moduly
- I/O module (IOM) – komponenta kontrolující interakci mezi controller modulem a disky, umožňuje propojení disků QSFP kabely
- DE inlet thermal sensor – vnitřní tepelný senzor DE
- DE thermal sensor – tepelný senzor DE
- fan expander module (FEM) – obsahuje expandery, které slouží jako interní interfejsy mezi I/O moduly a disky
- PCIe flash module (PFM) – rozšíření paměti pro systémové cache

Výstupem provádění těchto kontrol jsou chybová hlášení, která plugin předává monitorovacímu systému. Monitorovací systém na základě získaných dat informuje uživatele o stavu zařízení a případných chybách. Chybová hlášení jsou rozdělena do čtyř rozdílných stavů, konkrétně se jedná o stavy OK, WARNING, CRITICAL a UNKNOWN.

Vstupem pro zpracovávání informací o stavu komponent jsou data získávaná z MIB (Management information base) tabulek monitorovaných zařízení pomocí SNMP (Simple network managment protocol). Na základě takto získaných dat provede plugin analýzu stavů komponent a vytvoří výstupní informace.

U každé jednotlivé komponenty jsou jejich možné stavy rozděleny do kategorií podle závažnosti. Do stavu OK je u všech komponent zařazen stav, kdy je komponenta v pořádku. Stavy spadající do kategorie WARNING jsou takové, které nevyjadřují bezprostřední ohrožení, ale signalizují, že komponenta pracuje jinak, než bychom od ní očekávali. Kategorie CRITICAL zahrnuje ty stavy, které přímo ohrožují správnou funkci sledované komponenty. Stav UNKNOWN se využívá pro případy, kdy nelze o komponentach z nějakého důvodu získat informace nebo se nepodaří se zařízením navázat spojení.

V rámci chybových hlášení budou primárně vraceny informace o komponentách, jejichž stavy překračují hranice kategorií WARNING nebo CRITICAL. Budou tak uživatele přímo informovat o nastalé komplikaci a dále poskytnou informace o komponentě, která tuto situaci způsobila. Uživatel tak okamžitě dokáže zareagovat na nastalý problém. Pokud se nepodaří navázat spojení se zařízením, uživatel bude o tom informovám stavem UNKNOWN. Pokud budou všechny komponenty v pořádku, vrací se stav OK.

4 Přehled existujících řešení a přehled literatury

4.1 Vyhledávací kritéria

4.1.1 Aktuálnost zdrojů

Závěrečné práce považuji za aktuální od roku 2010, tedy 6 let zpět, z důvodu zachycení vývoje monitorovacích systémů. Starší práce by již mohly obsahovat techniky, které již nejsou v dnešní době používané.

Aktuálnost knih nebo internetových stránek věnujících se programovacímu jazyku Perl nijak neomezuji, přestože i tento jazyk se neustále vyvíjí. Základní principy práce s tímto jazykem se dají stále používat bez problémů.

Internetové zdroje a články zabývající se samotným monitorovacím systémem považuji za aktuální od roku 2012, tak aby zachycovaly informace o aktuální verzi monitorovacího systému, či jeho blízké předchozí verze, z jejíž principů aktuální verze vychází.

4.1.2 Klíčová slova

Vyhledávání bylo prováděno na základě následujících klíčových slov: centreon, monitoring, perl, plugin, snmp, disková pole, nagios, network monitoring, síťový monitoring.

4.1.3 Prostředky hledání

Vyhledávání bylo prováděno na těchto vyhledávacích portálech: www.google.com, www.thesis.cz, www.amazon.com, www.books.google.com, www.seznam.cz.

4.2 Závěrečné práce zabývající se monitoringem

Následující kapitola se zabývá prozkoumáním závěrečných prací na stejné nebo podobné téma. Cílem je zjistit, zda téma bakalářské práce nebylo již dříve někým zpracováváno, popřípadě získat informace a inspirovat se v řešených pracích na téma podobné.

Implementace monitorovacího systému v síti poskytovatele internetových služeb (Vosecký J.), diplomová práce

Vosecký (2012) se ve své práci zaměřuje na možnosti monitorovacích systémů. Dále se věnuje popisu a implementaci monitorovacích systémů Nagios a Centreon, jejichž použití a výsledky nasazení v reálně síti popisuje v druhé části práce.

Tato práce je pro mne velmi přínosná. Autor se ve čtvrté a páté kapitole věnuje podrobnému popisu instalace systému Nagios a následného využití nadstavby Centreon. Práce ukazuje původní možnost použití systému Centreon, kdy se jednalo pouze o nadstavbu webového rozhraní systému Nagios.

Integrace monitorovacího systému počítačové sítě na bázi open source a komerčních produktů (Benda L.), bakalářská práce

Benda (2011) se ve své práci zabývá popisem a srovnáním vybraných monitorovacích systémů na bázi open source, ale i komerčních produktů. Dále také prezentuje nasazení některých systémů na podnikové síti a popisuje potřebné kroky pro úspěšné implementování systémů.

Práce slouží k rozšíření přehledu o monitorovacích systémech. Mimo open source systémů jsou zde popsány i některé komerční systémy. Poměrně dobře jsou zde definované obecně sledované technické parametry.

Monitoring lokálnej siete prostredníctvom Nagiosu (Szabo G.), bakalářská práce

Szabo (2012) se v první části práce zabývá obecným popisem monitoringu a principem získávání dat o zařízeních v síti pomocí SNMP protokolu. V další části práce se věnuje návrhu pluginu pro monitorovací systém Nagios, kde se zabývá monitoringem tiskárny.

Přínosem práce je praktická ukázka komunikace pluginu s daným zařízením, která ukazuje způsob získávání potřebných údajů z MIB tabulek zařízení za pomocí SNMP protokolu.

Monitorování aktivních prvků počítačové sítě (Habrman Z.), diplomová práce

Habrman (2013) ve své práci popisuje základní principy a technické prostředky monitoringu. Dále se zaměřuje na srovnání nejpoužívanějších monitorovacích systémů. Po zhodnocení těchto systémů vybírá nejvhodnější systém, jehož implementací a popisem jednotlivých kroků implementace na síti univerzity Tomáše Bati ve Zlíně se v práci nadále zabývá.

Z práce lze získat obecný přehled o dalších monitorovacích systémech. Dále je zde dobře popsaná implementace monitorovacího systému Cacti a jeho výstupy. Tyto poznatky poslouží k subjektivnímu porovnání systému Centreon s jiným běžně užívaným monitorovacím systémem.

Správa různorodých síťových zařízení pomocí SNMP protokolu (Malík, J.), diplomová práce

Malík (2011) se ve své práci zabývá monitoringem různorodých síťových zařízení pouze pomocí SNMP protokolu, bez využití jakéhokoliv z dostupných monitorovacích softwarů. V práci je popsán vývoj vlastního monitorovacího systému založeného na SNMP protokolu, který využívá abstraktní vrstvy pro přístup k různorodým zařízením. V úvodu práce se také zabývá popisem principu a vzniku SNMP protokolu.

V práci je velmi dobře popsán SNMP protokol a princip komunikace pomocí tohoto protokolu. Dále je zde také dobře zpracován popis MIB tabulek.

Udržitelnost a rozvoj monitorovacích systémů v komerční společnosti, (Kocourková L.) diplomová práce

Kocourková (2013) se ve své práci nejprve zabývá obecným popisem několika monitorovacích systémů. V další části své práce posuzuje jednotlivé monitorovací systémy z hlediska několika definovaných kritérií. V závěru práce jednotlivé systémy srovnává a dále se věnuje popisu procesu nasazení vybraného systému ve vybrané firmě.

Práce je zajímavá zejména závěrečnou částí, kde autorka vytváří projekt pro nasazení monitorovacího systému v reálné firmě. Tato část práce přináší pohled na implemetaci monitorovacího systému i z ekonomického hlediska.

V rámci bakalářských a diplomových prací, uveřejněných v České republice, nebyla nalezena práce zabývající se monitoringem diskových polí na platformě Centreon. Obecně lze říci, že v současné době, existuje velmi málo závěrečných prací zabývajících se alespoň z části monitorovacím systémem Centreon. Lze však najít práce založené na monitorovací platformě Nagios, ze které Centreon vychází. Proto jsou i tyto práce jsou zajímavých přínosem cenných informací.

4.3 Přehled literatury

Cílem následující kapitoly je získat studijní zdroje potřebné pro nastudování dané problematiky a vyřešení zadaného úkolu.

An introduction to SNMP (Ellingwood, 2014)

Autor v článku popisuje základy SNMP protokolu, rozdíl mezi SNMP agenty a manažery, struktury MIB tabulek, příkazy a verze SNMP.

Článek poskytuje úvod do problematiky užívání SNMP protokolu a vysvětluje nezbytné základní pojmy.

How to set up your monitoring platform - configuration (IPINSIVY, 2013)

Autor ve svém článku popisuje proces nutný pro spuštění základního monitorovacího skriptu v systému Centreon. Popisuje vytvoření a definici monitorovaného hosta, dále příkazu a služby, přes kterou budeme hosta monitorovat a jejich vzájemné přiřazení.

Článek názorně ukazuje základní operace v monitorovacím systému Centreon. Jedná se o dobře zpracovaný tutoriál, podle kterého se lze naučit definování nových hostů.

How to set up your monitoring platform - part 1 (IPINSIVY, 2013)

Autor v článku popisuje základní stavební prvky monitorovacího systému Centreon a vysvětluje, k čemu slouží.

Článek slouží jako užitečné shrnutí informací o součástech systému.

Myslíme v jazyku PERL (Dařena, 2012)

Kniha slouží jako kompletní průvodce programovacím jazykem Perl od úplných základů až po různé speciality tohoto jazyka.

Tato publikace je použita jako studijní zdroj pro naučení se potřebných částí jazyka Perl, ve kterém bude napsaný plugin pro monitorovací systém.

Welcome to Centreon's documentation! (Centreon, 2015e)

Dokumentace k monitorovacímu systému Centreon. Stránky poskytují informace o funkcionalitě, ovládání a instalaci monitorovacího systému jako celku. Dokumentace je dále rozdělena na samotnou dokumentaci částí Centreon Engine, Centreon Broker, Centreon Plugins a Centreon.

Velmi kvalitně zpracovaná dokumentace umožňuje rychlé řešení problémů se systémem. Navíc je zde dobře popsán princip funkcionality monitorovacího systému a postup instalace systému krok po kroku.

5 Popis technologického aparátu

Transmission control protokol/Internet protokol

Rodina protokolů Transmission control protokol/Internet protokol (TCP/IP) obsahuje sadu protokolů zajišťujících přenost dat po síti. Narozdíl od OSI (Open Systems Interconnection) modelu rozlišujeme v TCP/IP čtyři vrstvy. Jedná se o vrstvu síťového rozhraní, síťovou vrstvu, transportní vrstu a vrstvu aplikační.

Nejnižší vrstvou architektury TCP/IP je vrstva síťového rozhraní, která zajišťuje přístup k fyzickému přenosovému médiu a slouží k fyzickému propojení a přenosu dat mezi zařízeními. Jako příklad lze uvést ethernet, token ring nebo FDDI.

Síťová vrstva zajišťuje adresaci zařízení v síti a stará se o správné směrování datagramů. Důležitým protokolem operujícím na síťové vrstvě je Internet protokol (IP), který je jedním ze stavebních kamenů síťové komunikace umožňující přenos informací a dat rozložených do datagramů od odesílatele k příjemci prostřednictvím počítačové sítě na základě IP adresace. IP sám o sobě nezajišťuje přenos ani kontrolu správnosti doručení dat a jejich konzistence, a proto je využíván ve spojení s protokoly transportní vrstvy. Jednotlivé datagramy vytvářené na úrovni síťové vrstvy obsahují řídící a uživatelská data. Řídící data poskytují informace směrovacím zařízením na síti, které je využívají ke správnému směrovaní datagramů k příjemci. Uživatelská data obsahují samotná přenášená data. Podrobnějším popisem IP se mimo jiné zabývá Kurose a Ross (2014).

Transportní vrstva vytváří proces přenosu dat mezi odesílatelem a příjemcem. Rozlišujeme dva přístupy k přenosu dat, a sice spolehivý přenos dat, který zajišťuje TCP (Transmission control protocol) a nespolehlivý přenos dat, který zajišťuje UDP (User datagram protocol).

TCP se používá pro zajištění správného přenosu dat a jejich konzistence mezi odesílatelem a příjemcem. Ke směrování datagramů, obsahujících odesílaná data TCP využívá Internet protokol (IP). Při přenosu dat TCP navazuje pevné spojení mezi odesílatelem a příjemcem, díky kterému se mohou obě strany podílet na kontrole dat. K navázání spojení dochází prostřednictvím tzv. trojcestného handshakingu, díky kterému se obě strany dohodnou na číslu sekvence a potvrzovacím čísle. Na základě těchto údajů dochází v průběhu celého spojení ke kontrole správného pořadí odesílaných a přijímaných datagramů a případným opravám chyb vzniklých při přenosu. Podrobnějším popisem TCP se mimo jiné zabývá Fall a Stevens (2011).

UDP je alternativou TCP. Využívá se zejména v oblastech, kde je kladen důraz na rychlý přenos velkého objemu dat, jako je například VoIP, streamování videa nebo online hry. Další využití UDP nachází v aplikacích a systémech pracujících na principu otázka-odpověď. Tohoto principu využívají často i klíčové systémy, jako jsou DNS, DHCP nebo SNMP. Narozdíl od TCP nenavazuje mezi odesílatelem a příjemcem spojení, čímž se však zvyšuje šance ztráty nebo narušení pořadí vysílaných paketů. Díky odstranění pevného spojení mezi příjemcem a odesílatelem se přenos datagramů značně zjednodušuje a snižuje se nadbytečný provoz na síti. Podrobnějším popisem UDP se mimo jiné zabývá Hartpence (2011).

Aplikační vrstva umožňuje přístup aplikacím ke komunikačnímu systému. Aplikační vrstva používá k rozlišení datagramů příslušných aplikací tzv. porty, na základě kterých dokáže rozlišovat, pro kterou aplikaci jsou příchozí nebo odchozí datagramy poskytovány. Příkladem jsou HTTP, DHCP nebo DNS.

Simple network managment protocol

Simple network managment protocol (SNMP) je nástroj pro sledování a řízení zařízení v síti. Umožňuje získávat informace o stavech zařízení, aniž bychom je museli fyzicky kontrolovat. Využívá se nejen k monitoringu konkrétních zařízení, ale i ke sledování stavu sítě. Při používání SNMP rozlišujeme zařízení na manažery a agenty. Manažery jsou označována zařízení, typicky servery, na kterých běží nějaký software určený pro monitoring. Často se můžeme setkat s názvem Network managment station (NMS), které mají za úkol reagovat na informace poskytované agenty. Agenty označujeme software běžící na monitorovaném zařízení. Jejich účelem je poskytovat informace o zařízení, kontrolu a popřípadě zasílat varování při neočekávané události.

Dnes existují tři verze SNMP. Jedná se o verzi 1 (SNMPv1), verzi 2 (SNMPv2) a verzi 3 (SNMPv3). SNMPv1 je původní verze protokolu definovaná v RFC 1157. Zabezpečení je v SNMPv1 řešeno tzv. komunitami, které definují možné operace prováděné na monitorovaných zařízeních. SNMPv2 obsahuje zejména opravené chyby z předchozí verze a stejně jako SNMPv1 používá komunity. Hlavním přínosem SNMPv3 je zabezpečení přístupu a přenosu dat. To je řešeno autentizací, autorizací, kontrolou přístupu a kódováním dat. Narozdíl od předchozích verzí ustupuje od komunit a používá definované uživatele k přístupu.

SNMP definuje tři přístupy k datům zařízení. Jednotlivé typy přístupů rozdělují přípustné operace prováděné uživatelem na daném zařízení. Přístup read-only povoluje uživateli pouze číst a znemožňuje provádět jakékoliv změny. Read-only přístup je často používan pro veřejně přístupné informace, bez nutnosti použití kontrolních zabezpečení. Dalším definovaným přístupem je read-write, který umožňuje provádět prostřednictvím SNMP změny v nastavení zařízení. Při používání tohoto přístupu je důležité využívat zabezpečení přístupu, abychom zamezili neoprávněným změnám a operacím. Posledním přístupem je trap, který NMS umožňuje zachytávat zprávy o chybách, které generuje agent.

Podrobnějším popisem SNMP se zabývá Mauro a Schmidt (2005).

Management information base

Management infromation base (MIB) reprezentuje sadu objektů, které jsou předmětem správy zařízení na počítačové síti. Reprezentuje databázi hierarchicky uspořádaných dat nesoucích informace o daném zařízení. MIB je využíván SNMP pro správu a monitorování zařízení na síti.

Podrobnějším popisem MIB se zabývá Blokdijk (2015).

Programovací jazyk Perl

Programovací jazyk Perl byl vyvíjen a do značné míry je i dnes používán jako skriptovací jazyk. V průběhu vývoje byl jazyk obohacován o práci s objekty, modulární programování nebo programování ve více vláknech. V dnešní době se používá zejména k tvorbě webových aplikací a zpracování velkého množství textu. Perl je specifický jazyk, požadovanou funkcionalitu lze napsat v poměrně krátkém kódu a bez velkých znalostí. Stinnou stránkou je skutečnost, že je poměrně těžké a složité se programovací jazyk naučit používat. Nenajdeme zde žádné deklarace proměnných, definice vlastních datových typů, typovou konverzi ani pevně danou sytaxi. Základním datovým typem programovacího jazyka Perl jsou tzv. skaláry. Ty umožňují uchovávat jedinou hodnotu, přičemž se může jednat o znak, číslo nebo řetězec. Skaláry jsou základními kameny složitějších datových typů, jako je pole skalárů nebo hash skalárů (Dařena, 2012).

K jazyku Perl existuje v současnosti velké množství modulů, které jsou vytvořený vývojáři nebo samotnými uživateli. Tyto dostupné moduly se snaží shromažďovat a dále poskytovat CPAN (Comprehensive Perl Archive Network), který v současnosti disponuje několika desítkami tisíc dostupných modulů.

Úvodem do programovacího jazyka a výukou se zabývá Dařena (2005), Schwartz a Foy a Phoenix (2011).

Monitorovací systém Centreon

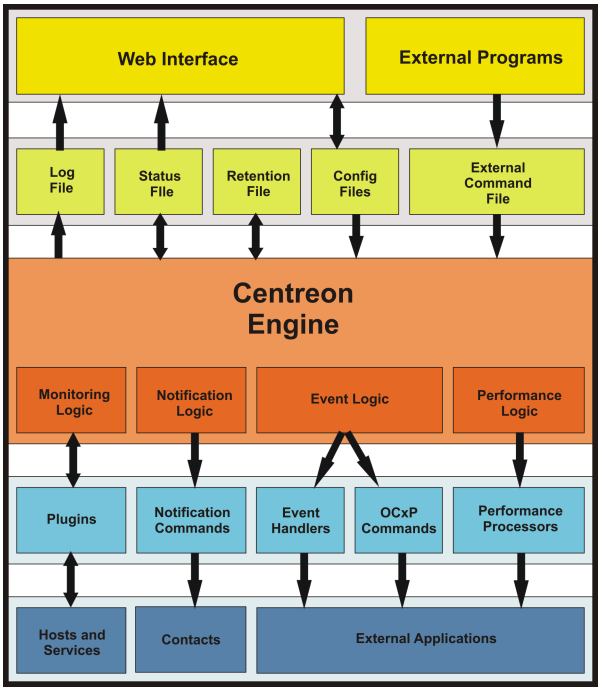
Centreon je monitorovací systém založený na bázi open source řešení a dnes je jedním z nejvíce používaných monitorovacích nástrojů. Centreon byl od svého vzniku v roce 2003 po dlouhou dobu vyvíjen pouze jako webové rozhraní použitelné na monitorovacím systému Nagios. V roce 2011 vzniká forkem Nagiosu verze 3. 2. 3 vlastní jádro monitorovacího systému Centreon (Centreon, 2015a).

Základní část systému, kterou tvoří Centreon Web Interface, Centreon Engine a Centreon Broker, je dostupná zdarma a rozšiřující balíčky si uživatel musí dokoupit. Zmíněné části systému jsou pro uživatele dostupné v produktovém balíčku Centron Enterprise Server (CES) Standard, který umožňuje snadnou a rychlou instalaci monitorovacího systému. CES je dále nabízen v produktových balících Essentials, Advanced a Complete které k základu systému nabízí rozšiřující služby a funkce. Jedná se o profesionální podporu, správu a údržbu aplikace třetí stranou, asistence při instalaci, softwarová rozšíření, šablony pro tvorbu pluginů, Centreon Map, Centreon Business Intelligence a Centreon Business Activity Monitoring. Ceny jednotlivých balíků začínají na 4 400 €, 7 400 € respektive 17 780 €. Alternativou je možnost instalace jednotlivých částí systému zvlášť (Centreon, 2015d). Toho lze využít, pokud chceme Centreon používat jako nadstavbu na monitorovacím systému Nagios nebo Icinga. Nejnovější stabilní verzí systému je Centreon 2. 7. 4 (Centreon, 2016).

Centreon má velmi dobře propracované webové rozhraní. Přehledně zobrazuje momentální výpis sledovaných zařízení a jejich aktuální stav. Prostřednictvím dat z monitorovaných zařízení, která jsou průběžně ukládána do databáze, dokáže

vytvářet časové řady a grafy umožňující sledovat vývoj za určité časové období. Prostřednictvím webového rozhraní je snadné vytvořit v systému novou monitorovanou entitu nebo monitorovací službu. Uživatel má tak na jednom místě přehledně uspořádané informace o všech monitorovaných zařízeních. Centreon také podporuje reportovací službu, která dokáže uživatele informovat o anomálii ve sledovaných datech prostřednictvím e-mailu nebo SMS zprávy.

Systém Centreon využívá k získávání dat z monitorovaných zařízení funkcionality pluginů, které získaná data předávají do systému ve formě výstupů zpracovatelných systémem Centreon. To umožňuje tvorbu vlastních pluginů, které mohou být určeny k monitorování široké škály síťových zařízení. Na obrázku č. 1 je zobrazena funkční logika monitorovacího systému Centreon.



Obrázek 1: Funkční logika monitorovacího systému Centreon (Centreon, 2015b)

Plugin v celém komplexu funguje jako zprostředkovatel přístupu a komunikace se zařízením. Prostřednictvím pluginu, který získává data z vybraného zařízení například pomocí SNMP, se požadované informace dostávají do jádra monitorovacího systému. Plugin musí poskytovat přesně definovaná hlášení o stavu, která jsou dále v enginu zpracována a interpretována. Engine systému Centreon prostřednictvím stavových souboru předává informace do webového rozhraní, kde jsou zobrazeny uživateli v ucelené grafické podobě. Mimo to může engine po příjmutí a vyhodnocení dat z pluginu odeslat uživateli notifikační zprávu a také ukládat získaná data pro pozdější tvorbu časových řad a grafů nebo poskytnutí dat externí aplikaci.

Monitorovací systém Centreon více popisuje Centreon (2015f) ve své dokumentaci k produktu.

Virtual private network

Virtual private network (VPN) umožňuje soukromé spojení mezi dvěma body nacházejícími se kdekoliv v síti Internet. Jde o zabezpečenou komunikaci nebo výměnu dat mezi dvěma subjekty a zamezuje možnost čtení obsahu datagramů třetí osobě. Zařízení spolu komunikují, jakoby byly připojeny ve společné privátní síti. Data jsou během přenosu zapouzdřena pomocí protokolů, které jsou určeny pro tunelové spojení VPN a zašifrována. Po doručení jsou data ověřována, zda během přenosu nedošlo k modifikaci. VPN využívá k zabezpečení autorizaci, autentizaci, kontrolu integrity dat a anti-reply.

Autentizaci chápeme jako proces, při kterém dochází k prokázání a ověření identity uživatele. Typickým prvkem autentizace je uživatelské jméno a heslo. Méně obvyklými prostředky je prokazování identity pomocí privátního klíče nebo biometrických vlastností.

Autorizace je přidělování práv uživatelům. Typicky se jedná o definování možných operací a přístupů k datům.

Díky kontrole integrity dat lze odhalit, zda během přenosu dat mezi dvěma subjekty nedošlo k neočekávanému narušení nebo ztrátě části přenášených dat nebo k neoprávněné modifikaci dat třetí osobou, o což se stará anti-reply. V případě zjištění, že k takové modifikaci nebo ztrátě dat došlo, příjemce požádá o nové zaslání narušených dat.

Popisem a možnostmi použití VPN se zabývá Hendel (2015).

6 Analýza přístupu k datům (MIB tabulky)

Následující kapitola se zabývá analýzou přístupu k datům zařízení a zejména analýzou a rozborem struktury MIB tabulek a informací v nich uložených. Je zde zejména popsanáno uspořádání a logická návaznost MIB záznamů, které jsou zdrojem požadovaných informací o zařízení.

6.1 Přístup k datům

K datům uložených na monitorovaných zařízeních bude plugin přistupovat pomocí SNMP. Disková pole používající MIB tabulky FJDARY-E60.mib a FJDARY-E101.mib podporují SNMPv1, v případě polí používajících FJDARY-E150.mib se navíc jedná o podporu SNMPv2c a SNMPv3. Data budou prostřednictvím SNMP čtena z jednotlivých MIB tabulek a dále zpracovávána. Všechny požadované informace, které budeme získávat ze zařízení, jsou zabezpečeny přístupem read-only. Nehrozí tak manipulace s daty, ke které může dojít omylem nebo záměrem třetí osoby.

Plugin bude pro přístup k datům využívat volně dostupný modul Net::SNMP, který je možný stáhnout v knihovně modulů programovacího jazyka Perl – CPAN. Modul obsahuje funkce definující použití SNMP v jazyku Perl. Modul podroběji popisuje Town (2010).

6.2 MIB tabulky

MIB tabulky jsou stěžejním zdrojem informací pro SNMP, díky kterému můžeme provádět monitoring na požadovaném zařízení. Výrobce diskových polí Fujitsu neposkytuje MIB tabulky veřejně přístupné na své oficiální stránce, ale jednotlivé MIB tabulky se dají získat exportem přímo z požadovaných zařízení. Z osmi monitorovaných diskových polí získáme tři MIB tabulky, protože více typů těchto polí používá stejnou strukturu a obsah tabulek.

Konkrétní rozdělení zařízení podle společných MIB tabulek je následující:

- FJDARY-E60.mib: Fujitsu Eternus DX60, DX80 a DX90
- FJDARY-E101.mib: Fujitsu Eternus DX60 S2 a DX100 S2
- FJDARY-E150.mib: Fujitsu Eternus DX100 S3, DX200 S3 a DX500 S3

Samotný soubor s MIB tabulkami obsahuje v textové formě definované objekty podle ASN.1(Abstract Syntax Notation One) uspořádané v hierarchické struktuře. Objekty v tabulkách jsou rozděleny do několika logických částí. Jedná se o počáteční definice, status jednotky, specifické objekty pro zachytávání hlášení, parametry subsystému, fyzické komponenty, logická konfigurace, informace ze senzorů, informace o firmwaru, informace o výkonu a definici hlášení. Pro další účely bakalářské práce

jsou zajímavé části parametry subsystému a fyzické komponenty. Obsahují informace o komponentách, které má plugin kontrolovat. Všechny dále zveřejněné MIB záznamy a OID hodnoty jednotlivých komponent jsou převzaty od Fujitsu (2009, 2011, 2013).

Pro definování přístupu k určitému vybranému objektu je zapotřebí specifikovat jeho cestu ve stromové struktuře MIB záznamů. K určení cesty se používá OID (Object Identifier) který je reprezentován posloupností číslic oddělených nulou reprezentující uzly ve struktuře MIB. Nejdříve musíme definovat OID výrobce diskových polí Fujitsu a dostat se v jeho produktech na zařízení určená pro ukládání dat. V následující části MIB tabulky je zobrazen záznam popisující hodnoty a označení jednotlivých uzlů.

```
-- FUJITSU Specific Object Identifier

fujitsu                         OBJECT IDENTIFIER ::= { enterprises 211 }
product                         OBJECT IDENTIFIER ::= { fujitsu 1 }
storage                         OBJECT IDENTIFIER ::= { product 21 }

-- User Specific Object Identifier(1)

nsp                             OBJECT IDENTIFIER ::= { storage 1 }
fjdarye60                       OBJECT IDENTIFIER ::= { nsp 60 }
```

První šestičíslí je typické pro všechny společnosti, které chtějí registrovat svoje MIB záznamy. Jedna se o OID 1.3.6.1.4.1, která nás ve stromové struktuře zavede do sekce enterprises. Jak je na obrázku vidět, uzel fujitsu je pod číslem 211, produkty této firmy jsou pod číslem 1 a zařízení určená pro ukládání dat jsou pod číslem 21. Dalším společným uzlem pro všechna disková pole je uzel nsp definovaný číslem 1. Následující číslo již reprezentuje strukturu MIB, která se bude u rozdílných typů diskových polí lišit. Při zpracování pluginu se na tuto důležitou informaci musíme zaměřit, abychom se odkazovali na správnou MIB strukturu monitorovaného pole. Zde tedy můžeme získat hodnoty 60, 101 nebo 150, v závislosti na příslušnost diskového pole k MIB tabulce. OID k specifickému diskovému poli, v případě ukázkové části tabulky, kde se jedná o pole používající MIB tabulku FJDARY-E60.mib, je 1.3.6.1.4.1.211.1.21.1.60. Další části řetězce OID hodnot se budou odvíjet od požadované informace, kterou chceme ze zařízení získat.

Pro splnění požadované funkcionality pluginu je třeba nálézt vhodné záznamy v MIB tabulkách, ze kterých budeme moci získávat požadovaná data. Začneme pátráním po záznamu, který obsahuje informace o typu pole, ke kterému přistupujeme a jeho sériovém čísle. Tento záznam nalezneme v části parametry subsystému pod označením *fjdarySspMachineId*.

```
fjdarySspMachineId OBJECT-TYPE
    SYNTAX OCTET STRING
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "This value indicates the identification number of this system.
         ttsssssssssssmmmmmmmmmmmmccsssssssssss
```

```
        | |           |           | |
        | |           |           | +--serial number(12byte)
        | |           |           +--check code(2byte)
        | |           +--model name(12byte)
        | +--series name(12byte)
        +--type code(2byte)"
    ::= { fjdarySsp 1 }
```

Dotazem na tento MIB záznam získáme řetězec o délce 40 bajtů, ze kterých můžeme vyčíst potřebné údaje. Z prvních 26 bajtů získáme kód typu zařízení (2 B), název série (12 B) a název modelu (12 B). Na následujících dvou bajtech se nachází kontrolní kód a na zbývajících dvanácti nalezneme sériové číslo.

Získávání informací o stavu komponent už bude o poznání složitější a komplexnější. Každý typ komponenty, který budeme na diskovém poli sledovat, je uložen ve specifickém MIB záznamu. Jelikož je možné, že se komponenty monitorovaných typů komponent v těchto polích vyskytují více než jednou, obsahuje každý takový záznam o typu komponenty tabulku udržující informace o všech komponentách daného typu.

```
fjdaryThmlCount OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The number of thermal sensor's~informations (regardless of their
         current state) present on this system."
    ::= { fjdaryThml 1 }

fjdaryThmlTable OBJECT-TYPE
    SYNTAX SEQUENCE OF FjdaryThmlEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        ""
    ::= { fjdaryThml 2 }
```

Princip těchto záznamů je u všech typů komponent stejný, příklad postupu získávání informací si popíšeme na tepelných čidlech. V záznamu *fjdaryThmlCount* se dozvídáme o počtu tepelných čidel na daném diskovém poli. To je důležitá informace, díky níž můžeme procházet tabulku záznamů o jednotlivých tepelných čidlech, která je definována v záznamu *fjdaryThmlTable*. Zde je důležitá informace, že tabulka je sekvencí záznamů *FjdaryThmlEntry*.

```
fjdaryThmlEntry OBJECT-TYPE
    SYNTAX FjdaryThmlEntry
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        ""
    INDEX { fjdaryThmlIndex }
    ::= { fjdaryThmlTable 1 }

FjdaryThmlEntry ::=
```

```
    SEQUENCE{
        fjdaryThmlIndex  INTEGER,
        fjdaryThmlItemId INTEGER,
        fjdaryThmlStatus FjdaryComponentStatus
    }

fjdaryThmlIndex OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The value of this object uniquely identifies this fjdaryThml entry."
    ::= { fjdaryThmlEntry 1 }

fjdaryThmlItemId OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The value of this object indicates the identification number of this
         thermal sensor."
    ::= { fjdaryThmlEntry 2 }

fjdaryThmlStatus OBJECT-TYPE
    SYNTAX FjdaryComponentStatus
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The value of this object indicates the state of this thermal sensor."
    ::= { fjdaryThmlEntry 3 }
```

Záznam tepelného čidla *FjdaryThmlEntry* obsahuje tři základní informace, které nalezneme mimo jiných i ve všech ostatních komponentách. Jedná se o index, id a stav komponenty. Zatímco index a id jsou integerové hodnoty, stav komponenty je typu *FjdaryComponentStatus*. Jedná se o záznam, který obecně definuje možné stavy všech komponent diskového pole. Na základě této informace zjistíme, jaký stav komponenta vykazuje. V následujícím MIB záznamu jsou vypsány možné hodnoty stavů, které komponenty pole mohou vykazovat.

```
-- Textual Conventions
FjdaryComponentStatus ::= INTEGER{
    normal(1),       -- Normal state
    alarm(2),        -- Abnormal state
    warning(3),      -- Warning state
    invalid(4),      -- Not installed state
    maintenance(5), -- Maintenance state
    undefined(6)     -- Undefined state
}
```

Takto definované stavy mohou vykazovat všechny typy komponent diskového pole mimo disků a LAN portů, které mají definované stavy vlastní. Příklad možných stavů disků je na následujícím obrázku.

```
fjdaryDiskStatus OBJECT-TYPE
    SYNTAX INTEGER{
```

```
        available(1),
        broken(2),
        notavailable(3),
        notsupported(4),
        present(5),
        readying(6),
        recovering(7),
        partbroken(64),
        spare(65),
        formatting(66),
        unformatted(67),
        notexist(68),
        copying(69)
    }
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The value of this object indicates the state of this DISK.

         available    : Normal state
         broken       : Abnormal state
                        (The data can not be guaranteed)
         notavailable : Not available state
                        (Because the RAID group is not set)
         notsupported : Unsupported error state
         present      : Undefined volume set state
         readying     : Data readying state
         recovering   : Data recovering state
         partbroken   : Abnormal state
         spare        : Using spare disk state
         formatting   : Data formatting state
         unformatted  : Unformatted state
         notexist     : Not exist
         copying      : Data copying state"
    ::= { fjdaryDiskEntry 3 }
```

Informace o ostatních typech komponent jsou v MIB tabulkách zaznamenány na stejném principu. Jde tedy zejména o identifikaci potřebných OID vedoucí k požadovaným záznamům.

V následujících tabulkách jsou shrnuty potřebné OID pro získání požadovaných informací z zařízeních. Základem pro všechny uvedené hodnoty je 1.3.6.1.4.1.211.1.21.1.*60/101/150*.2. , kde hodnoty 60/101/150 závisejí na typu použité MIB tabulky.

Tabulka 1: Souhrn OID pro získávaná data z FJDARY-E60.mib a FJDARY-E101.mib

Typ komponenty	Počet	Stav	SN/ID
controller module (CM)	1.1.0	1.2.1.3	1.2.1.7
channel adapter	2.1.0	2.2.1.3	2.2.1.2
CM memory	3.1.0	3.2.1.3	3.2.1.6
CM flash-rom	4.1.0	4.2.1.3	4.2.1.2
system capacitor unit (SCU)	5.1.0	5.2.1.3	5.2.1.7
controller enclosure (CE)	6.1.0	6.2.1.3	6.2.1.2
device enclosure (DE)	7.1.0	7.2.1.3	7.2.1.4
expander	8.1.0	8.2.1.3	8.2.1.6
power supply unit (PSU)	9.1.0	9.2.1.3	9.2.1.5
inlet thermal sensor	10.1.0	10.2.1.3	10.2.1.2
thermal sensor	11.1.0	11.2.1.3	11.2.1.2
disk	12.1.0	12.2.1.3	12.2.1.2
LAN port	13.1.0	13.2.1.3	13.2.1.2 a 3

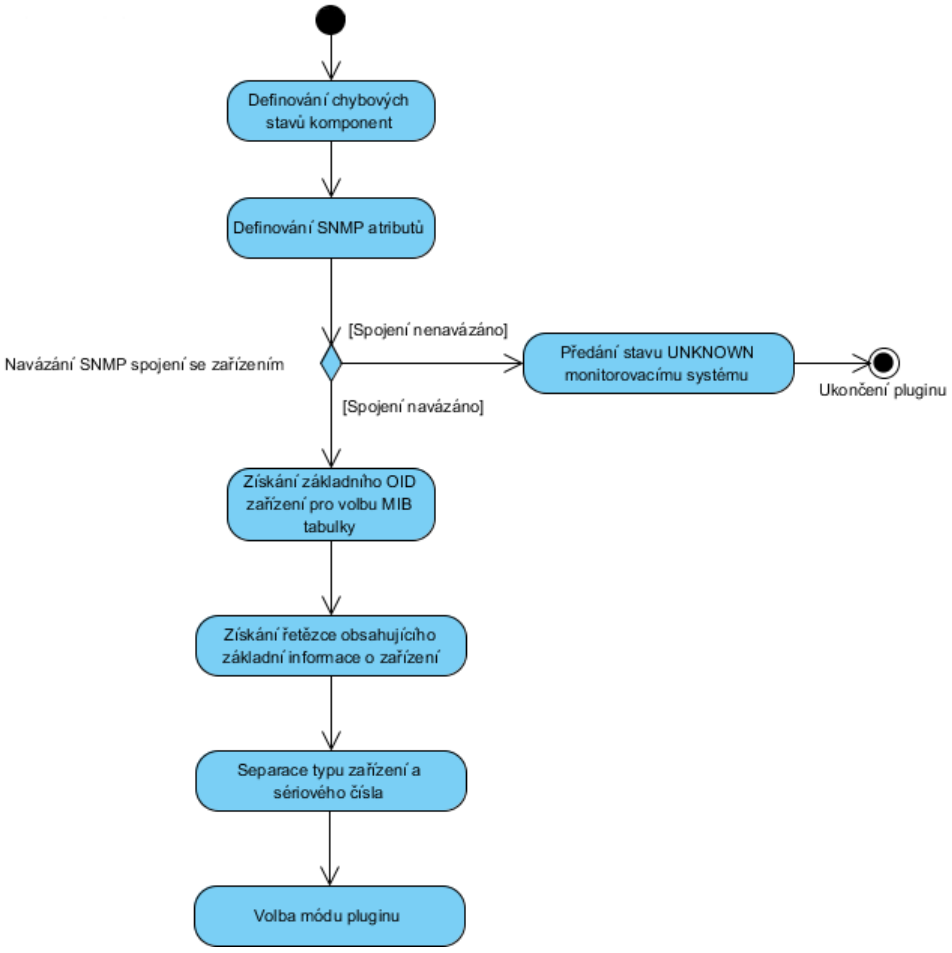
Tabulka 2: Souhrn OID pro získávaná data z FJDARY-E150.mib

Typ komponenty	Počet	Stav	SN/ID
controller module (CM)	1.1.0	1.2.1.3	1.2.1.8
CM central processing unit	2.1.0	2.2.1.3	2.2.1.2
channel adapter	3.1.0	3.2.1.3	3.2.1.2
CM memory	4.1.0	4.2.1.3	4.2.1.7
boot-up and utility device (BUD)	5.1.0	5.2.1.3	5.2.1.2
CM fan	6.1.0	6.2.1.3	6.2.1.2
battery control unit (BCU)	7.1.0	7.2.1.3	7.2.1.2
battery unit (BTU)	8.1.0	8.2.1.3	8.2.1.2
system capacitor unit (SCU)	9.1.0	9.2.1.3	9.2.1.2
controller enclosure (CE)	10.1.0	10.2.1.3	10.2.1.7
CE inlet thermal sensor	11.1.0	11.2.1.3	11.2.1.2
CE thermal sensor	12.1.0	12.2.1.3	12.2.1.2
CE power supply unit (CPSU)	13.1.0	13.2.1.3	13.2.1.6
drive enclosure (DE)	14.1.0	14.2.1.3	14.2.1.5
I/O module (IOM)	15.1.0	15.2.1.3	15.2.1.7
power supply unit (PSU)	16.1.0	16.2.1.3	16.2.1.6
DE inlet thermal sensor	17.1.0	17.2.1.3	17.2.1.2
DE thermal sensor	18.1.0	18.2.1.3	18.2.1.2
disk	19.1.0	19.2.1.3	19.2.1.4
LAN port	20.1.0	20.2.1.3	20.2.1.6
fan expander module (FEM)	21.1.0	21.2.1.3	21.2.1.2
PCIe flash module (PFM)	22.1.0	22.2.1.3	22.2.1.2

7 Návrh řešení

Následující kapitola se zabývá rozborem principu funkcionality pluginu. Před zahájením implementace pluginu je dobré mít ucelenou představu o principu jeho funkcionality, kterou si nastíníme definováním logiky programu. K utříbení a vyjádření algoritmického postupu využiji vývojový diagram, ve kterém budou uvedeny jednotlivé kroky, vyjadřující stěžejní části algoritmu pluginu. Na následujících obrázcích jsou uvedeny jednotlivé části vývojového diagramu doplněné popisem jejich významu.

Úvodní část vývojového diagramu na obrázku č. 2 se zabývá inicializační definicí potřebných dat, kontrolou navázání spojení s daným zařízením a identifikací základních informací.

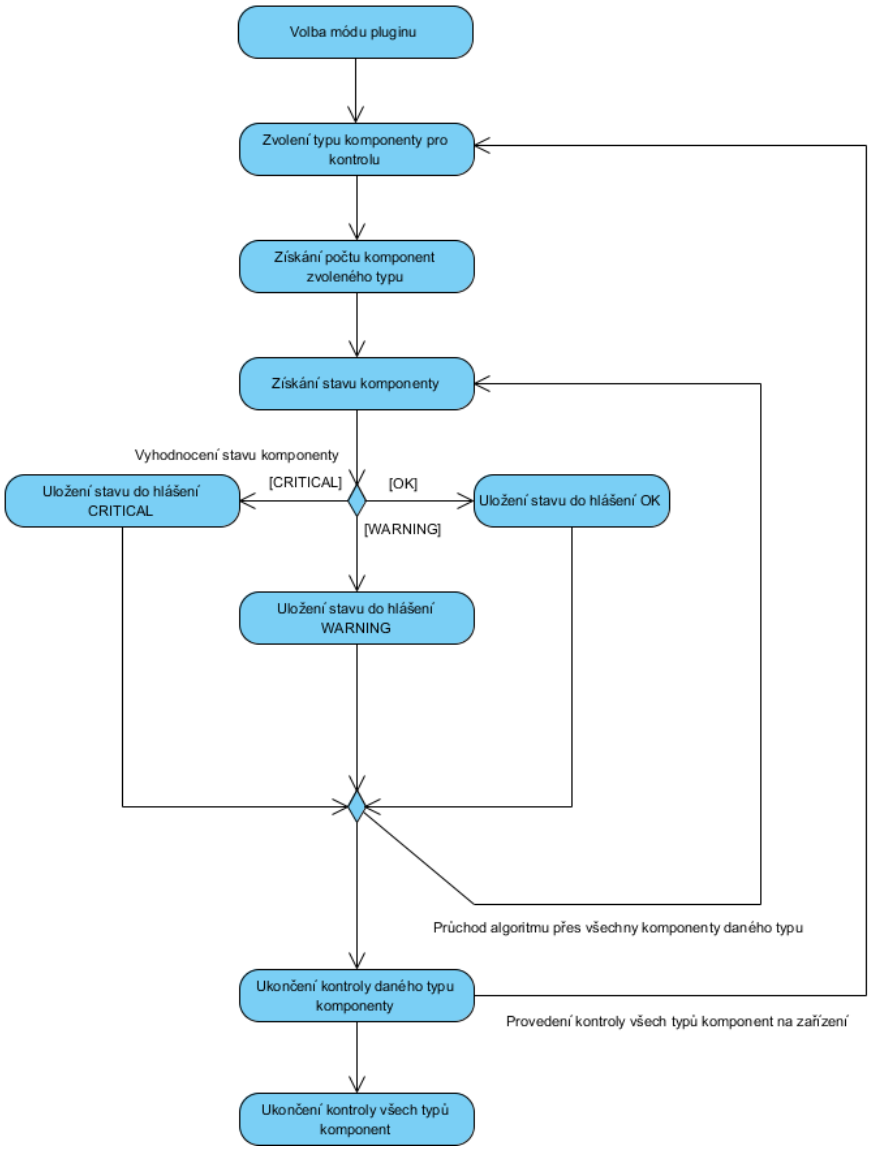


Obrázek 2: Vývojový diagram pluginu – část 1

- **Definování chybových stavů komponent**
  Příprava datových struktur, které budou později využívány k ukládání a transformaci chybových hlášení o stavech jednotlivých komponent.

- **Definování SNMP atributů**
  Definice SNMP atributů potřebných pro úspěšné vytvoření SNMP spojení. Jedná se o IP adresu monitorovaného zařízení, jeho SNMP verzi a SNMP komunitu.
- **Navázání SNMP spojení se zařízením**
  Pokus o vytvoření SNMP spojení na základě dříve definovaných hodnot. V případě, že je spojení úspěšně navázáno, plugin vykonává další akce. Pokud se připojení nepodaří navázat, následuje akce *Předání stavu UNKNOWN monitorovacímu systému* a následně dojde k ukončení pluginu.
- **Předání stavu UNKNOWN monitorovacímu systému**
  Akce, která se vykoná v případě neúspěšného připojení. Plugin předává jako svůj výstup pro monitorovací systém Centreon stav UNKNOWN s doplňující informací, která obsahuje příčinu neúspěšného připojení.
- **Získání základního OID zařízení pro volbu MIB tabulky**
  Pro určení typu monitorovaného zařízení musíme nejprve zjistit, jakou MIB tabulku zařízení používá. V této fázi je tedy nutné určit základní část OID zařízení, které budeme dále využívat pro přístup k potřebným informacím. Základní OID zařizení je hodnota, na základě které se dostaneme stromovou strukturou OID identifikátorů k určení MIB záznamů konkrétního diskového pole.
- **Získání řetězce obsahujícího základní informace o zařízení**
  Po zjištění základního OID již lze přistupovat na základě znalostí získaných z rozboru MIB tabulek zařízení k informacím, jenž jsou středem zájmu. V této fázi dojde k získání řetězce obsahujícího typový kód, název série, název modelu, kontrolní kód a sériové číslo.
- **Separace typu zařízení a sériového čísla**
  Získané informace z předešlého kroku je nutné rozdělit a vybrat jen informace potřebné pro zpracování. Jedná se o název série, název modelu a jeho sériové číslo.
- **Volba módu pluginu**
  Po získání základních informací o zařízení následuje samotný proces získávání požadovaných informací o stavech komponent. Ty uživatel definoval volbou módu při spuštění pluginu a nyní se na základě této volby rozhoduje, které typy komponent se v daném běhu programu budou kontrolovat.

Druhá část vývojového diagramu na obrázku č. 3, navazující na část první, vyjadřuje nejdůležitější část pluginu a to získávání informací o stavech jednotlivých komponent zařízení.



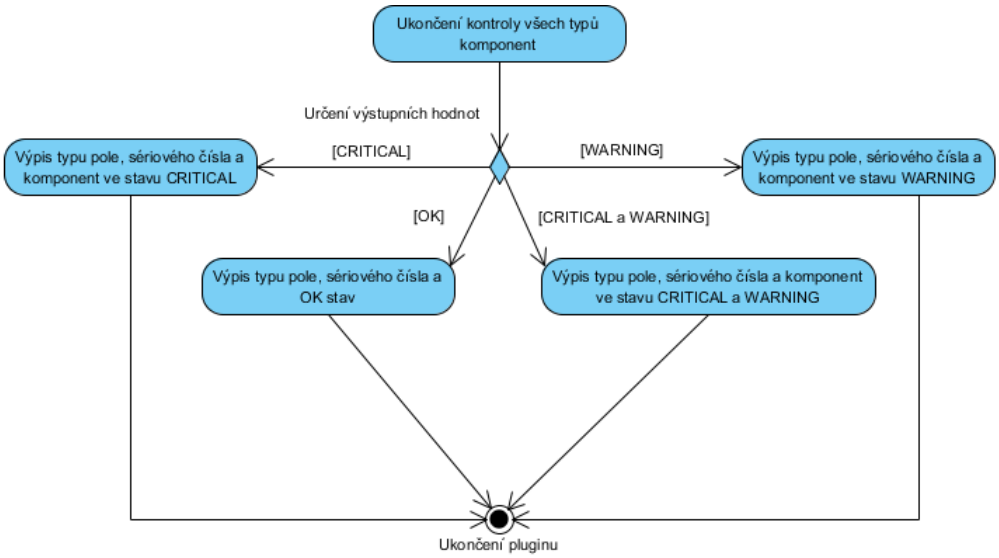
Obrázek 3: Vývojový diagram pluginu – část 2

- **Zvolení typu komponenty pro kontrolu**
  Na základě typu monitorovaného zařízení známe množinu komponent, které zařízení obsahuje a které je prostřednictvím SNMP protokolu možné sledovat. V této části dochází k výběru jednoho typu komponenty, kterou monitorované zařízení obsahuje (např. controller module) a zvolený typ komponenty bude následně zkotrolován.

- **Získání počtu komponent zvoleného typu**
  Po výběru jednoho typu komponenty následuje určení počtu komponent na zařízení zvoleného typu.
- **Získání stavu komponenty**
  Získání stavu, který aktuálné vykazuje kontrolovaná komponenta. OID reprezentující stavová hlášení mohou být u různých typů komponent na různém místě, je třeba tyto odlišnosti rozlišovat na základě struktury MIB tabulek.
- **Vyhodnocení stavu komponenty**
  Na základě získané hodnoty popisující stav komponenty je vytvořeno hlášení o komponentě a následně přiřazeno do datové struktury reprezentující stav OK, WARNING nebo CRITICAL. Rozmezí hodnot spadajících do příslušných datových struktur reprezentující chybová hlášení jsou subjektivně rozděleny podle závažnosti stavu.
- **Uložení stavu do hlášení OK**
  Pokud stav komponenty vykazuje hodnotu 1, reprezentující stav normal, pak je údaj o komponentě obsahující typ komponenty, ID nebo sériové číslo a její stav uložen do datové struktury udržující hlášení o komponentách ve stavu OK.
- **Uložení stavu do hlášení CRITICAL**
  Pokud stav komponenty vykazuje hodnotu 2, reprezentující stav abnormal, pak je údaj o komponentě obsahující typ komponenty, ID nebo sériové číslo a její stav uložen do datové struktury udržující hlášení o komponentách ve stavu CRITICAL.
- **Uložení stavu do hlášení WARNING**
  Pokud stav komponenty vykazuje hodnotu 3, 5 nebo 6 reprezentující stavy warning, maintenance a undefined, pak je údaj o komponentě obsahující typ komponenty, ID nebo sériové číslo a její stav uložen do datové struktury udržujících hlášení o komponentách ve stavu WARNING.
- **Průchod algoritmu přes všechny komponenty daného typu**
  Podle počtu komponent zvoleného typu definovaných na sledovaném zařízení se postup získávání stavu komponenty a jeho následného vyhodnocení iteruje přes všechny komponenty zvoleného typu. To znamená, že pokud jsou na zařízení dvě komponenty typu controller module, pak bude tato kontrola provedena pro obě komponenty.
- **Ukončení kontroly daného typu komponenty**
  Po provedené kontrole stavů všech komponent daného typu se ukončuje kontrola tohoto typu a v případě, že na zařízení jsou stále ještě nezkontrolované typy komponent, plugin pokračuje s kontrolou zbylých typů.

- **Ukončení kontroly všech typů komponent**
  Po provedení kontroly všech typů komponent přechází plugin k tvorbě výstupu do monitorovacího systému Centreon.

Poslední část vývojového diagramu na obrázku č. 4, navazující na část druhou, se zabývá tvorbou výstupu pluginu pro monitorovací systém Centrteon, na základě informací získaných po provedení kontroly komponent zařízení.



Obrázek 4: Vývojový diagram pluginu – část 3

- **Určení výstupních hodnot**
  Na základě obsahu datových struktur udržujících získaná hlášení o stavech zkontrolovaných komponent se rozhoduje o typu hlášení předávaného na výstup pluginu.
- **Výpis typu pole, sériového čísla a OK stav**
  Pokud jsou datové struktury udržující informace o komponentách ve stavu WARNING a CRITICAL prázdné, pak plugin předává na svůj výstup stav zařízení OK a doplňující informace o typu zařízení a jeho sériové číslo.
- **Výpis typu pole, sériového čísla a komponent ve stavu WARNING**
  Pokud datová struktura udržující informace o komponentách ve stavu WARNING obsahuje nějaký záznam a přitom datová struktura udržující informace o komponentách ve stavu CRITICAL je prázdná, pak plugin předává na svůj výstup stav zařízení WARNING a doplňující informace o typu zařízení, jeho sériové číslo a výčet komponent vykazujících stav *warning, maintenance nebo undefined*.

- **Výpis typu pole, sériového čísla a komponent ve stavu CRITICAL**
  Pokud datová struktura udržující informace o komponentách ve stavu CRITICAL obsahuje nějaký záznam a přitom datová struktura udržující informace o komponentách ve stavu WARNING je prázdná, pak plugin předává na svůj výstup stav zařízení CRITICAL a doplňující informace o typu zařízení, jeho sériové číslo a výčet komponent vykazujících stav *abnormal*.
- **Výpis typu pole, sériového čísla a komponent ve stavu CRITICAL a WARNING**
  Pokud datová struktura udržující informace o komponentách ve stavu CRITICAL obsahuje nějaký záznam a přitom datová struktura udržující informace o komponentách ve stavu WARNING udržuje také nějaký záznam, pak plugin předává na svůj výstup stav zařízení CRITICAL a doplňující informace o typu zařízení, jeho sériové číslo a výčet komponent vykazujících stav *abnormal, warning, maintenance nebo undefined*.
- **Ukončení pluginu**
  Předáním hodnot na výstup plugin končí.

8 Implementace

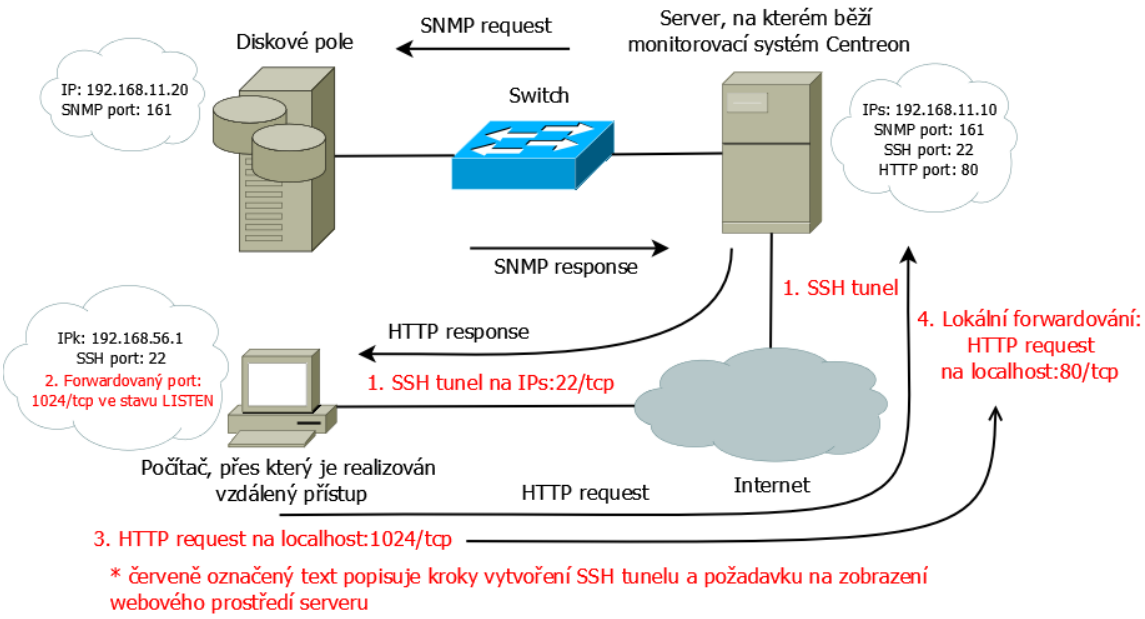
8.1 Přístup k zařízení a monitorovacímu systému Centreon

V praxi je běžné nasazení monitorovacího systému na serveru připojeném přímo v podnikové síti, tak aby byl umožněn co nejjednodušší a nejrychlejší přístup nástrojů monitorovacího systému k zařízením určených pro monitoring. Pro účely vývoje a pozdějšího testování pluginu je zapotřebí stálé připojení na zařízení, která jsou předmětem monitoringu. Firma CDC Data, s. r. o. pro tyto účely zajistila možnost připojení na daná zařízení prostřednictvím technologie VPN. Jedná se o vzdálené připojení na server, na kterém běží monitorovací systém Centreon pomocí SSH tunelu. Spolu se serverem jsou v síti připojena testovací zařízení, což umožňuje testovat vyvíjený plugin na všech zařízeních, pro která je plugin určen. V praxi pak samotný plugin bude integrován do podnikového monitorovacího systému v podnikové síti.

SSH tunel umožňuje vytvoření soukromého spojení mezi dvěma body v síti Internet. Zabezpečuje šifrovaný kanál, po kterém mohou být přenášena nekódovaná data a díky možnosti obejití firewallu sítě umožňuje zobrazování interních webových stránek (Chamith, 2014). Toho využijeme pro zobrazení webového rozhraní monitorovacího systému Centreon, který je dostupný na localhostu serveru, na který vytváříme SSH tunel.

8.1.1 Síťová topologie přístupu

Na obrázku č. 5 je znázorněna síťová topologie představující zapojení, které bylo realizováno pro účely navázání spojení se zařízením a následného testování pluginu.



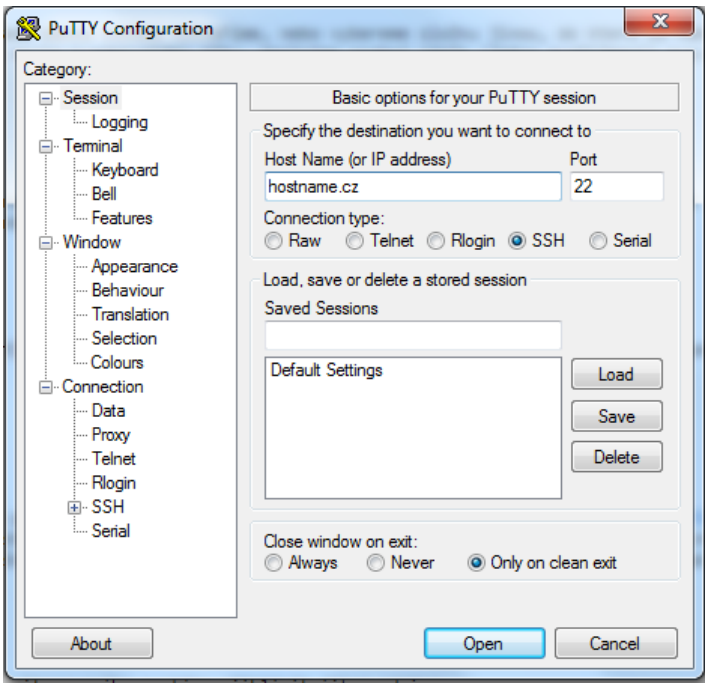
Obrázek 5: Síťová topologie přístupu k diskovému poli

Prostřednictvím jakéhokoliv počítače připojeného do sítě Internet realizujeme pomocí SSH tunelu, jehož vytvoření bude popsáno později, vzdálené připojení na server. Díky tomuto spojení máme možnost pomocí příkazové řádky manipulovat se serverem a také možnost zobrazení webového prostředí localhostu serveru ve webovém prohlížeči počítače. Toho později využijeme pro zobrazení webového prostředí monitorovacího systému Centreon, který je na tomto serveru již nainstalovaný. Pro získávání dat ze zařízení použijeme funkcionality pluginu, který bude uložen na serveru a použitím SNMP získá požadovaná data. Díky možnosti ovládání serveru pomocí vzdálené příkazové řádky můžeme tento plugin ve fázi implementačního testování spouštět ručně, později bude automaticky spouštěn monitorovacím systémem.

8.1.2 Vytvoření přístupu

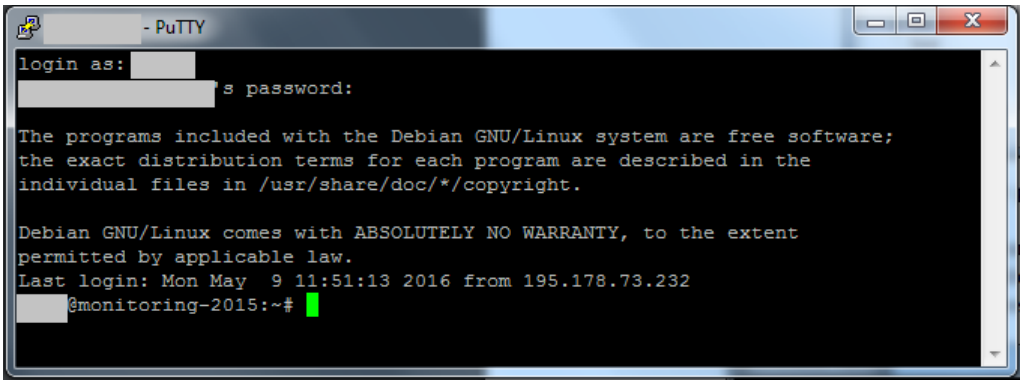
Pro vytvoření přístupu na server, který je zapojený ve společné lokální síti s monitovanými zařízeními a ze kterého se bude vytvořený plugin spouštět, použijeme software, který umí navázat vzdálené připojení pomocí SSH. V našem případě jsme použili program PuTTY. Pro možnost zobrazení webového prostředí monitorovacího systému Centreon, které běží na localhostu serveru, je dále zapotřebí vytvořit SSH tunel, díky kterému budeme schopnit zobrazit webové prostředí Centreonu na vzdáleně připojeném počítači. Abychom nemuseli vytvářet plugin v konzolovém prostředí připojení na server, máme možnost vytvořený soubor na počítači přenést na server pomocí softwaru umožňující přenos souborů, například použitím SSH file transfer protokolu (SFTP). K tomuto účelu použijeme program WinSCP.

Na obrázku č. 6 je základní prostředí programu PuTTY. Nastavení pro vytvoření klasického SSH spojení je velmi jednoduché a intuitivní. Do políčka hostname zadáváme doménové jméno nebo IP adresu zařízení, se kterým chceme vytvořit SSH spojení, a port, který má zařízení nastaveno pro naslouchání příchozích pokusů o SSH připojení. Server standardně tomuto připojení naslouchá na portu *22*. Po vyplnění těch údajů zbývá zvolit typ spojení, které se chceme pokusit navázat jako *SSH* a tlačítkem *Open* se program pokusí navázat SSH spojení se serverem.



Obrázek 6: Definice SSH spojení v programu PuTTY

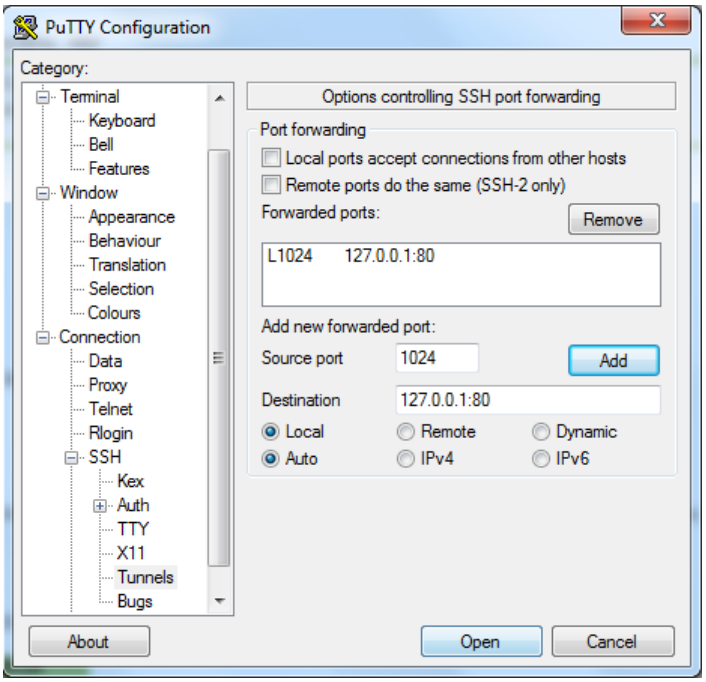
Pokud je server dostupný a umožní navázat SSH spojení, začne po nás vyžadovat autentizační údaje. Ty je zapotřebí správně vyplnit a po jejich ověření se vytvoří požadované SSH spojení na server. Otevřené SSH spojení je na obrázku č. 7.



Obrázek 7: Otevřené SSH spojení se serverem

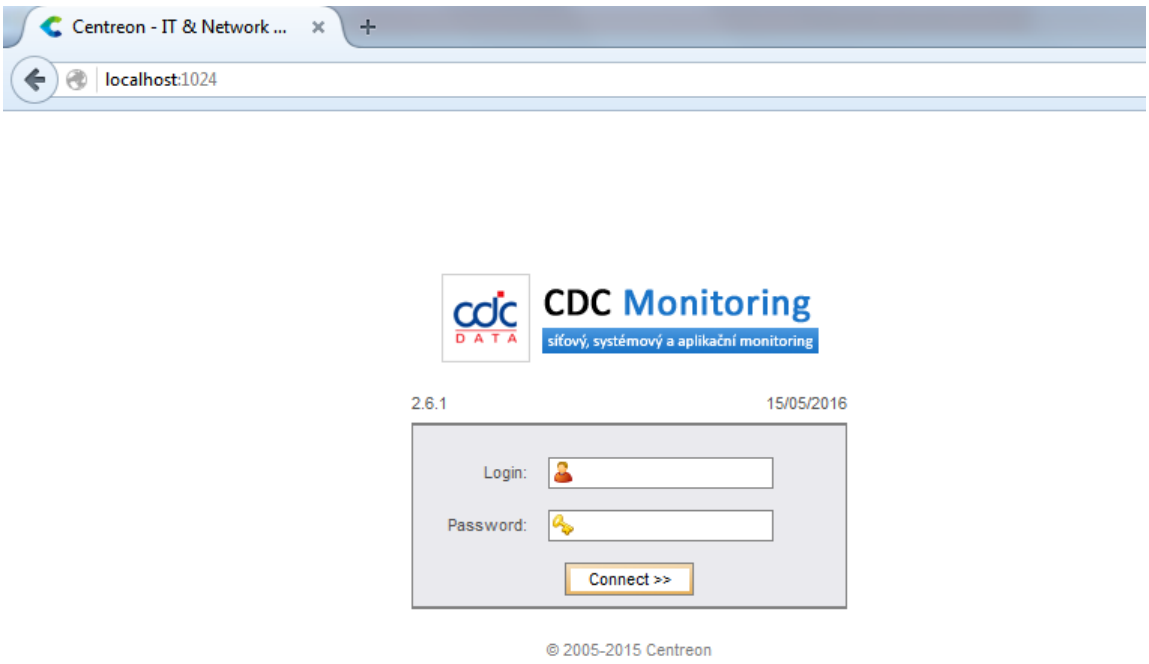
Tímto máme vytvořené SSH spojení na server. Nyní můžeme pomocí příkazového řádku v otevřeném SSH spojení vzdáleně provádět operace na serveru. Abychom však mohli vzdáleně v prohlížeči zobrazit webové prostředí Centreonu, je nejdříve zapotřebí vytvořit SSH tunel. Současné spojení tedy ukončíme a znovu spustíme program PuTTY. K nastavení SSH tunelu se dostaneme z úvodní obrazovky programu. Nejprve však vyplníme potřebné informace pro vytvoření klasického SSH spojení, které jsme si popsali před chvílí, ale prozatím se nepokoušíme kliknutím na

tlačítko *Open* spojení navázat. V levém sloupci úvodní obrazovky nalezneme v kategorii *Connection* podkategorii *SSH* a v ní volbu *Tunnels*. Zde budeme vyplňovat políčka *Source port* a *Destination*. Do políčka *Source port* musíme vložit číslo portu, přes který bude přesměrováván veškerý síťový přenos mezi počítačem a serverem na který SSH tunel vytváříme. Vložíme tedy libovolné číslo od 1 do 65535, ale volíme pouze takové číslo portu, které není již využíváno jinou síťovou službou. Tento port tedy reprezentuje výchozí port počítače, z kterého budeme vytvářet SSH tunel na port serveru. Zvolíme například číslo portu *1024*. Do políčka *Destination* vložíme adresu *127.0.0.1:80* reprezentující localhost na serveru, na který vytváříme SSH tunel. IP adresa je rozšířena určením portu, který chceme prostřednictvím SSH tunelu využívat. Jelikož potřebujeme zobrazit webové prostředí Centreonu, volíme port *80* určený pro HTTP. Dále ponecháme zakliklé políčko *Local* a klikneme na *Add*. Nyní máme nastaveny všechny potřebné údaje a po kliknutí na tlačítko *Open* se vytvoří SSH spojení s tunelem. Možné nastavení SSH tunelu je na obrázku č. 8.



Obrázek 8: Definice SSH tunelu v programu PuTTY

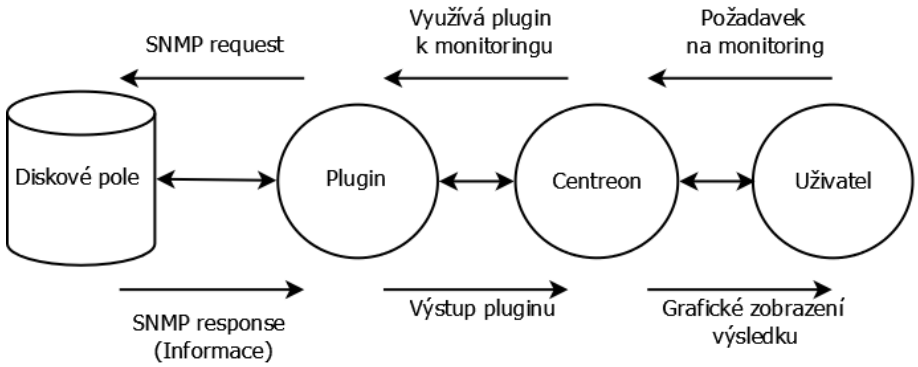
Nyní již máme vytvořené SSH spojení i s tunelem, a můžeme se pokusit otevřít webové prostředí serveru zadáním adresy *localhost:1024* do webového prohlížeče. Tímto dojde k využití SSH tunelu, který náš HTTP požadavek odešle přes port počítače *1024* na localhost serveru s využitím portu *80* pro HTTP. Dojde tak ke zobrazení webového prostředí Centreonu, běžící na localhostu serveru. Úvodní obrazovka webového prostředí monitorovacího systému Centreon je na obrázku č. 9.



Obrázek 9: Webové prostředí monitorovacího systému Centreon na serveru

8.1.3 Logika přenosu informací z diskového pole k uživateli

V následující podkapitole je popsána cesta, kterou musí data absolvovat, než se dostanou k cílovému uživateli. Zároveň znázorňuje logické celky a vazby mezi nimi, které jsou do procesu monitoringu zařízení zapojeny. Tyto skutečnosti jsou znázorněny na obrázku č. 10.



Obrázek 10: Logická struktura prvků monitoringu

Celý proces získávání informací z diskového pole začíná v okamžiku, kdy uživatel vznese požadavek na provedení této operace v monitorovacím systému Centreon. Vytvoření tohoto požadavku bude popsáno později.

Monitorovací systém Centreon tento požadavek interpretuje a k jeho splnění využívá funkcionality pluginu, která spočívá v získávání požadovaných informací o diskovém poli.

Plugin využitím funkcí SNMP request a SNMP response provádí čtení požadovaných informací přímo ze zařízení a následně získané informace zpracovává.

Po dokončení čtení všech požadovaných informací a jejich zpracování plugin odesílá monitorovacímu systému zpracované informace v upravené podobě, kterou je monitorovací systém schopný interpretovat.

Tyto přejaté informace monitorovací systém převádí do podoby grafického zobrazení a prezentuje je uživateli.

8.2 Nasazení a instalace monitorovacího systému Centreon

8.2.1 Nasazení

Nasazení monitorovacího systému Centreon bude realizováno na platformě Unix, konkrétně na distribuci Debian Wheezy. Systém Unix má dlouhodobě značný podíl na serverových nasazeních. Důkazem může být statistika nasazení systému Linux na webových serverech, kde jeho podíl na trhu činí 68 % (W3Techs, 2016). Dalším důvodem pro výběr operačního systému Unix je jeho balíčkový systém pro instalaci programů. Instalace Centreonu je dostupná i prostřednictvím balíčkové instalace, při jejímž průběhu se uživatel může hlouběji dozvědět, z jakých částí se monitorovací systém skládá a které nástroje a funkce potřebuje.

Hlavní komponenty monitorovacího systému Centreon nejsou nainstalovány v nejnověji publikovaných verzích. Na základě zkušeností Ing. Tomáše Holomka s tímto systémem bylo doporučeno využít otestované stabilní verze hlavních komponent systému, které jsou navzájem kompatibilní. Jedná se o Centreon Engine verze 1.4.6, Centreon Broker verze 2.6.3, Centreon Clib verze 1.4.0 a Centreon 2.5.2.

8.2.2 Instalace

Samotná instalace monitorovacího systému nebyla v bakalářské práci využita, jelikož testování pluginu probíhá na vzdáleném serveru firmy CDC Data, s. r. o., na kterém je monitorovací systém již nasazený. Procesem instalace jsem však prošel v rámci předmětu Aplikace vývojových technik, který slouží jako příprava na bakalářkou práci za účelem seznámení se s monitorovacím systémem a osvojení si potřebných funkcí tohoto programu.

Následující popis instalace monitorovací systému Centreon obsahuje všechny důležité kroky, které jsou potřebné pro úspěšnou instalaci systému. Průvodce se zabývá méně standardní formou instalace pomocí zdrojových souborů programu. Monitorovací systém je nasazován na Unixový systém Debian Wheezy.

Přehled jednotlivých kroků instalace:

- Instalace základních prerekvizit
- Doplnění on-line repozitářů
- Centreon Engine – vytvoření uživatele a skupiny
- Centreon Engine – instalace prerekvizit
- Centreon Engine – konfigurace a kompilace
- Centreon Broker – vytvoření uživatele a skupiny
- Centreon Broker – instalace prerekvizit
- Centreon Broker – konfigurace a kompilace
- Instalace Centreonu
- Nastavení monitorovacího systému

Instalace základních prerekvizit

První krok, který je zapotřebí provést, je doinstalovat všechny potřebné prerekvizity monitorovacího systému. Během celé instalace bude využívat balíčkové instalace jednotlivých komponent. V následujícím kódu jsou všechny prerekvizity uvedeny.

```
root@debian:~# apt-get install sudo tofrodos bsd-mailx lsb-release mysql-server
    libmysqlclient18 libdatetime-perl apache2 apache2-mpm-prefork php5 php5-
    mysql php-pear php5-intl php5-ldap php5-snmp php5-gd php5-sqlite rrdtool
    librrds-perl libconfig-inifiles-perl libcrypt-des-perl libdigest-hmac-perl
    libdigest-sha-perl libgd-gd2-perl snmp snmpd libnet-snmp-perl libsnmp-perl
```

Doplnění on-line repozitářů

Je možné, že některé balíky nepůjdou nainstalovat. Chybové hlášení oznamující, že požadovaný balík nemá kandidáta pro instalaci, může signalizovat chybějící definice cest k on-line repozitářům. Tyto repozitáže reprezentují servery, ze kterých je možné požadované balíčky stahovat. Seznam těchto cest je při defaultní instalaci operačního systému definován cestou */etc/apt/sources.list*. Tento soubor otevřeme pomocí libovolného textového editoru a upravíme tak, aby vypadal například tak, jak je uvedeno v následujícím kódu.

```
deb http://security.debian.org/ wheezy/updates main contrib
deb-src http://security.debian.org/ wheezy/updates main contrib

deb http://ftp.cz.debian.org/debian/ wheezy-updates main contrib
deb-src http://ftp.cz.debian.org/debian/ wheezy-updates main contrib

deb http://ftp.cz.debian.org/debian/ wheezy main contrib non-free
deb-src http://ftp.cz.debian.org/debian/ wheezy main contrib non-free

deb http://ftp.debian.org/debian wheezy-backports main
```

Po doplnění cest k potřebným repozitářům je nutné aktualizovat systémový přehled o externích repozitářích pomocí příkazu *apt-get update* a poté doinstalovat pomocí příkazu *apt-get install snmp-mibs-downloader* SNMP MIBy, které nejsou na Debianu implicitně instalovány, popřípadě další balíčky, jejichž prvotní instalace selhala. Tímto máme všechny potřebné prerekvizity pro Centreon nainstalované a můžeme přejít k samotné instalaci součástí monitorovacího systému.

Centreon Engine - vytvoření uživatele a skupiny

Základní součástí monitorovacího systému je Centreon Engine. Ten nejprve vyžaduje instalaci vlastních prerekvizit, začneme však vytvořením uživatele pro engine, jehož vytvoření se doporučuje z bezpečnostních důvodů. Vytvoření skupiny a nového uživatele je provedeno v následujícím kódu.

```
root@debian:~# groupadd centreon-engine
root@debian:~# useradd -g centreon-engine -m -r -d /var/lib/centreon-engine
    centreon-engine
```

Centreon Engine - instalace prerekvizit

Dále se v rámci doplnění prerekvizit jedná o kompilalační prostředí jazyka C++, jehož instalaci provedeme příkazem *apt-get install build-essential cmake* a dále musíme nainstalovat nutnou součást enginu Centreon Clib, jehož požadovanou verzi musíme stáhnout z oficiálních stránek Centreonu (www.download.centreon.com). Stažený soubor je komprimován a je zapotřebí jej rozbalit. K tomu použijeme příkaz *tar xzf centreon-clib.tar.gz*. Nyní si otevřeme rozbalenou složku, v níž nalezneme složku build, ve které se nachází základní skripty pro instalaci. Pomocí příkazu *cmake* můžeme provést konfiguraci instalačních skriptů, tak, jak je uvedeno v následujícím kódu.

```
root@debian:~/centreon-clib-1.4.0/build# cmake -DWITH_TESTING=0 -DWITH_PREFIX=/
    usr -DWITH_PREFIX_LIB=/usr/lib -DWITH_PREFIX_INC=/usr/include/centreon-clib
     -DWITH_SHARED_LIB=1 -DWITH_STATIC_LIB=0 -DWITH_PKGCONFIG_DIR=/usr/lib/
    pkgconfig .
```

Po provedení konfigirace následuje kompilace zdrojových skriptů a instalace Centreonu Clib. Příkazem *make* provedeme kompilaci a nainstalujeme příkazem *make install*.

Centreon Engine - konfigurace a kompilace

Tím jsme úspěšně splnili požadované prerekvizity a můžeme přejít k instalaci samotného enginu. Stáhneme si komprimovaný soubor Centreon Engine a soubor rozbalíme. Dostaneme se do podsložky *build* a příkazem *cmake* provedeme konfiguraci instalačních skripů, tak, jak je zobrazeno v následujícím kódu.

```
root@debian:~/centreon-engine-1.4.6/build# cmake -DWITH_PREFIX=/usr -
    DWITH_PREFIX_BIN=/usr/sbin -DWITH_PREFIX_CONF=/etc/centreon-engine -
    DWITH_PREFIX_LIB=/usr/lib/centreon-engine -DWITH_USER=centreon-engine -
    DWITH_GROUP=centreon-engine -DWITH_LOGROTATE_SCRIPT=1 -DWITH_VAR_DIR=/var/
```

```
    log/centreon-engine -DWITH_RW_DIR=/var/lib/centreon-engine/rw -
    DWITH_STARTUP_DIR=/etc/init.d -DWITH_PKGCONFIG_SCRIPT=1 -
    DWITH_PKGCONFIG_DIR=/usr/lib/pkgconfig -DWITH_TESTING=0
```

Po provedení konfigurace opět následuje kompilace zdrojových skriptů a instalace Centreonu Engine. Příkazem *make* provedeme kompilaci a nainstalujeme příkazem *make install*.

Centreon Broker - vytvoření uživatele a skupiny

Po úspěšné instalaci Centreon Engine následuje další základní část systému Centreon Broker. Začneme opět vytvořením skupiny a uživatele pro Centreon Broker a instalací prerekvizit. Vytvoření skupiny a uživatele je zobrazeno na následujícím kódu.

```
root@debian:~# groupadd centreon-broker
root@debian:~# useradd -g centreon-broker -m -r -d /var/lib/centreon-broker
    centreon-broker
```

Centreon Broker - instalace prerekvizit

Pokračujeme instalací prerekvizit, jejichž instalace je zobrazena v následujícím kódu.

```
root@debian:~# apt-get install build-essential cmake librrd-dev libqt4-dev
    libqt4-sql-mysql libgnutls28-dev
```

Při instalaci prerekvizity *libgnutls28-dev* může dojít k chybě, kdy požadovaný balíček je závislý na dalších balících, které nejsou naistalovány. Pro instalaci tohoto balíčku a všech jeho prerekvizit použijeme příkaz *apt-get -t wheezy-backports install libgnutls28-dev*.

Centreon Broker - konfigurace a kompilace

Dále postupujeme stejně jako u předešlých částí. Stáhneme komprimovaný soubor Centreon Broker, ten rozbalíme a dostaneme se do složky *build*, kde provedeme konfiguraci instalačních skriptů, tak, jak je znázorněno v následujícím kódu.

```
root@debian:~/centreon-broker-2.6.3/build# cmake -DWITH_DAEMONS='central-broker
    ;central-rrd' -DWITH_GROUP=centreon-broker -DWITH_PREFIX=/usr -
    DWITH_PREFIX_BIN=/usr/sbin -DWITH_PREFIX_CONF=/etc/centreon-broker -
    DWITH_PREFIX_INC=/usr/include/centreon-broker -DWITH_PREFIX_LIB=/usr/lib/
    nagios -DWITH_PREFIX_MODULES=/usr/share/centreon/lib/centreon-broker -
    DWITH_STARTUP_DIR=/etc/init.d -DWITH_STARTUP_SCRIPT=auto -DWITH_TESTING=0 -
    DWITH_USER=centreon-broker
```

Posléze provedeme příkazem *make* kompilaci a nainstalujeme příkazem *make install*.

Instalace Centreonu

Nyní máme nainstalované všechny základní komponenty monitorovacího systému a můžeme přejít k samotné instalaci Centreonu. Opět si stáhneme komprimovaný

soubor Centreon a rozbalíme ho. Následně otevřeme rozbalenou složku a v ní spustíme příkazem *./install.sh -i* instalační skript. Samotný průběh instalačního skriptu je velmi zdlouhavý, proto budou v následujícím kódu zveřejněny pouze akce, které vyžadují změny výchozích hodnot. U ostatních dotazů, mimo uvedených v následujícím kódu, ponecháme výchozí hodnoty.

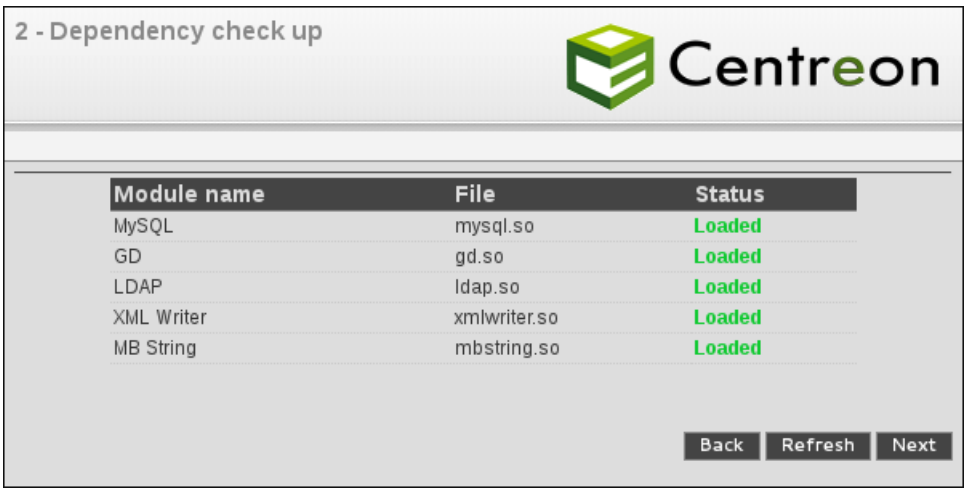
```
Do you accept GPL license ?
[y/n], default to [n]:
> y
-------------------------------------------------------------------------------
        Please choose what you want to install
-------------------------------------------------------------------------------

Do you want to install : Centreon Web Front
[y/n], default to [n]:
> y

Do you want to install : Centreon CentCore
[y/n], default to [n]:
> y

Do you want to install : Centreon Nagios Plugins
[y/n], default to [n]:
> y

Do you want to install : Centreon Snmp Traps process
[y/n], default to [n]:
> y

-------------------------------------------------------------------------------
        Start CentWeb Installation
-------------------------------------------------------------------------------

Where is your Centreon directory?
default to [/usr/local/centreon]
> /usr/share/centreon
Path /usr/share/centreon

Where is your Centreon log directory
default to [/usr/local/centreon/log]
> /var/log/centreon

Do you want me to create this directory ? [/var/log/centreon]
[y/n], default to [n]:
> y
Path /var/log/centreon

Where is your Centreon binaries directory
default to [/usr/local/centreon/bin]
> /usr/share/centreon/bin

Do you want me to create this directory ? [/usr/share/centreon/bin]
[y/n], default to [n]:
> y
Path /usr/share/centreon/bin                                    OK
```

```
Where is your Centreon data informations directory
default to [/usr/local/centreon/data]
> /usr/share/centreon/data

Do you want me to create this directory ? [/usr/share/centreon/data]
[y/n], default to [n]:
> y
Path /usr/share/centreon/data

Do you want me to create this directory ? [/var/lib/centreon]
[y/n], default to [n]:
> y

What is the Monitoring engine user ?
> centreon-engine

What is the Broker user ? (optional)
> centreon-broker

What is the Monitoring engine log directory ?
> /var/log/centreon-engine

------------------------------------------------------------------------
        Configure Sudo
------------------------------------------------------------------------

What is the Monitoring engine init.d script ?
> /etc/init.d/centengine

What is the Monitoring engine binary ?
> /usr/sbin/centengine

What is the Monitoring engine configuration directory ?
> /etc/centreon-engine

Where is the configuration directory for broker module ?
> /etc/centreon-broker

Where is the init script for broker module daemon ?
> /etc/init.d/cbd
Your sudo is not configured

Do you want me to configure your sudo ? (WARNING)
[y/n], default to [n]:
> y
Configuring Sudo

Do you want to add Centreon Apache sub configuration file ?
[y/n], default to [n]:
> y

Do you want to reload your Apache ?
[y/n], default to [n]:
> y

------------------------------------------------------------------------
```

```
        Start CentCore Installation
-------------------------------------------------------------------------------

Do you want me to install CentCore init script ?
[y/n], default to [n]:
> y

Do you want me to install CentCore run level ?
[y/n], default to [n]:
> y

-------------------------------------------------------------------------------
        Start CentPlugins Traps Installation
-------------------------------------------------------------------------------

Where is your CentreonTrapd binaries directory
default to [/usr/local/centreon/bin/]
> /usr/share/centreon/bin

Do you want me to install CentreonTrapd init script ?
[y/n], default to [n]:
> y

Do you want me to install CentreonTrapd run level ?
[y/n], default to [n]:
> y
```

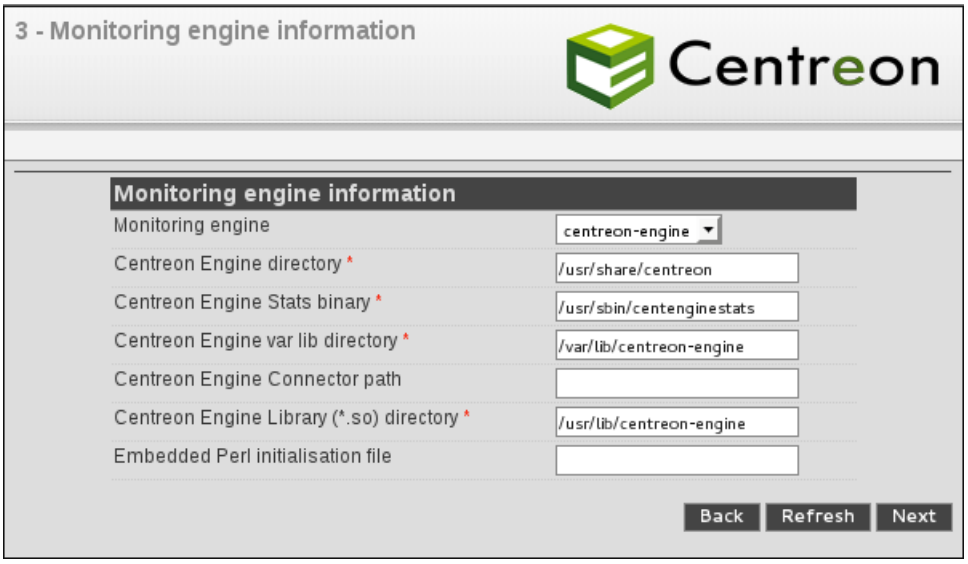
Nastavení monitorovacího systému

Po provedení instalace monitorovacího systému je zapotřebí dokončit konfiguraci ve webovém prostředí monitorovacího systému, do kterého se dostaneme prostřednictvím webového prohlížeče. Adresa je ve výchozím nastavení *localhost/centreon*. První dialogové okno nás uvítá ve webovém prostředí monitorovacího systému. Klikneme na tlačítko *Next*, čímž se dostáváme ke kontrole závislostí. Všechny by měly být ve stavu *Loaded*, jak je vidět na obrázku č. 11.



Obrázek 11: Kontrola závislostí ve webovém prostředí monitorovacího systému Centreon

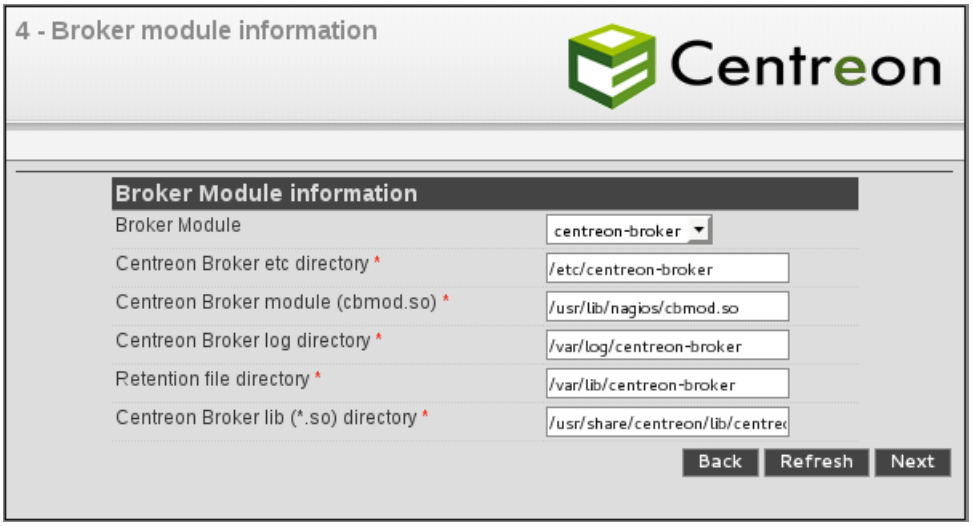
Pokud jsou všechny závislosti v pořádku, pokračujeme tlačítkem *Next* na další dialogové okno, ve kterém musíme vyplnit údaje o instalovaném monitorovacím enginu. Monitorovací engine tvoří jádro celého monitorovacího systému. Jedná se o aplikaci vykonávající všechny monitorovací činnosti. V prvním kroku vybereme typ monitorovacího enginu, v našem případě *Centreon Engine*. Po výběru enginu se nám zobrazí další políčka s předdefinovánými hodnotami. Ty je však třeba upravit podle obrázku č. 12 tak, abychom měli požadované cesty správně definované.



Obrázek 12: Doplnění informací o monitorovacím enginu monitorovacího systému Centreon

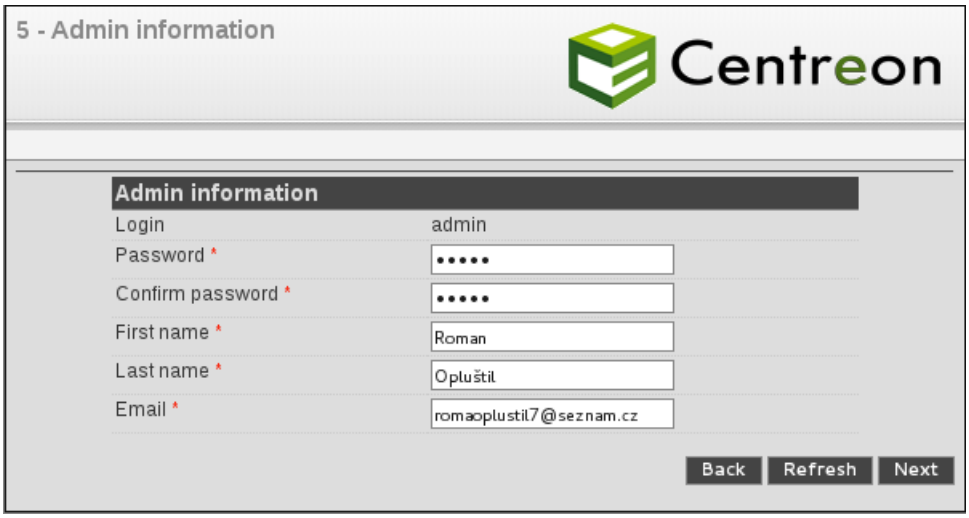
Po provedení úprav bychom se měli tlačítkem *Next* dostat k dalšímu dialogovému oknu, ve kterém budeme vyplňovat údaje o instalovaném monitorovacím modulu broker. Tento modul v monitorovacím systému zastává funkci datového multiplexeru. Stará se o přenos dat mezi monitorovacím enginem a databází. Zde v prvním kroku

vybereme typ broker modulu, v našem případě Centreon Broker. Po výběru brokeru se nám zobrazí další políčka s předdefinovánými hodnotami, které však musíme opět upravit podle obrázku č. 13.



Obrázek 13: Doplnění informací o broker modulu monitorovacího systému Centreon

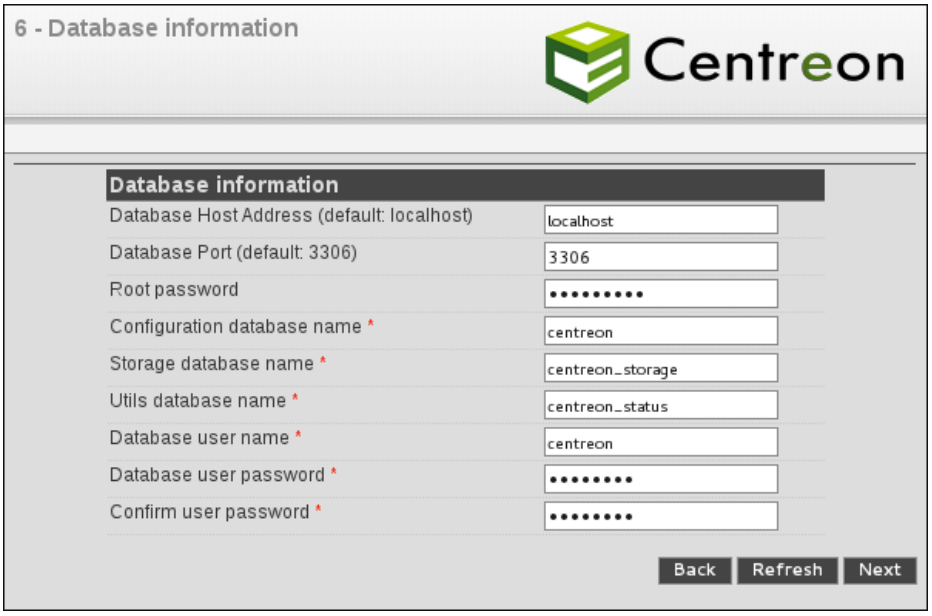
Může se stát, že neexistuje složka pro ukládání logů brokeru. V takovém případě danou složku vytvoříme nebo vybereme složku jinou, do které se nám budou tyto logy ukládat. Po provedení úprav pokračujeme tlačítkem *Next* k dalšímu dialogovému oknu. Zde vyplníme heslo k adminskému účtu, doplníme osobní údaje (jméno a příjmení) a e-mailovou adresu. Ilustrační příklad je na obrázku č. 14.



Obrázek 14: Vložení informací o adminovi monitorovacího systému Centreon

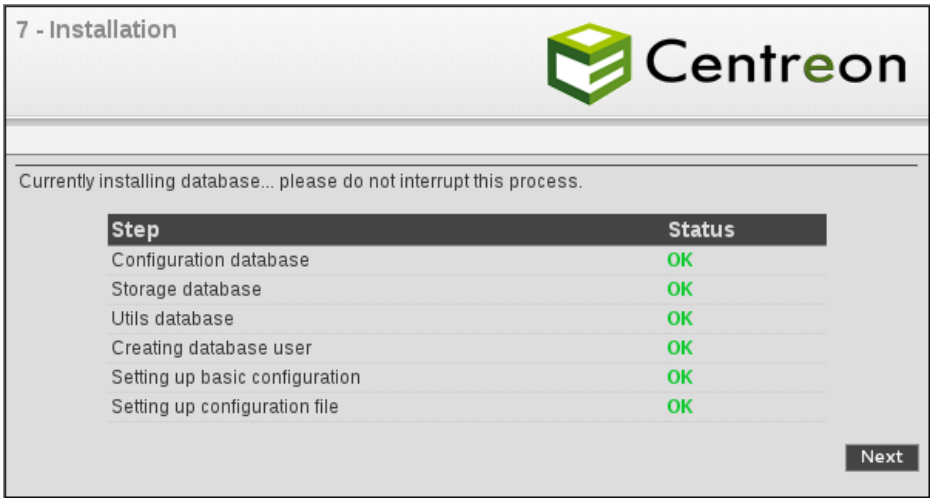
Po vložení všech požadovaných informací pokračujeme tlačítkem *Next* na další dialogové okno. Zde vyplňujeme informace o databázi. Je nutné vytvořit uživatele

databáze, tedy jeho login a heslo a doplnit heslo roota systému. Ostatní položky ponecháme na výchozích hodnotách tak, jak je vidět na obrázku č. 15.



Obrázek 15: Vložení informací o databázi monitorovacího systému Centreon

Po dokončení úprav pokračujeme na další dialogové okno tlačítkem *Next*, kde probíhá instalace databáze. Počkáme, než se dokončí všechny operace, které by měly být po skončení instalace ve stavu *OK* tak, jak je vidět na obrázku č. 16.



Obrázek 16: Instalace databáze monitorovacího systému Centreon

Pokračujeme tlačítkem *Next*, čímž se dostaneme na konec instalačního procesu. Je možné, že se zobrazí chybová hláška *Add innodb_file_per_table=1 in my.cnf file under the [mysqld] section and restart MySQL Server.*, v takovém případě se ve

svém systému přihlásíme pod účtem roota a modifikujeme soubor */etc/my.cnf* přidáním řádku *innodb_file_per_table=1*. Pokud byly všechny předchozí kroky úspěšně provedeny, otevře se nám přihlašovací okno do systému Centreon, jak je vidět na obrázku č. 17.

Obrázek 17: Prostředí pro přihlášení do monitorovacího systému Centreon

V takovém případě instalace všech součástí proběhla úspěšně a můžeme se do monitorovacího systému přihlásit účtem admina.

8.3 Spustitelné módy pluginu

Plugin je možné spouštět v několika různých módech, které rozhodují o tom, které typy komponent má plugin kontrolovat. Pro všechna disková pole, pro které je plugin určen, existuje společná množina použitelných módů. Dále pak existují přídavné módy, které lze použít pouze na zařízeních používající určitý typ MIB tabulky. Tyto rozdílné módy vyplývají zejména z rozdílných typů komponent, které jednotlivá zařízení mohou obsahovat. Následuje přehled módů použitelných v pluginu.

Společný základ:

- overall – kontrola všech komponent
- sensors – kontrola tepelných senzorů
- psu – kontrola zdrojů napájení
- disks – kontrola disků
- lanports – kontrola LAN portů
- scu – kontrola elektrických dvojvrstvých kondenzátorů nahrazující konvenční baterie
- channeladapters – kontrola kanálových adaptérů realizujících konektivitu k hostům

Módy pro zařízení používající MIB tabulky FJDARY-E60.mib a FJDARY-E101.mib:

- cmandcomponents – kontrola modulů obsahujících CPU a hlavní paměť, procesorů, které jsou součástí controller modulu, paměti controller modulu a větráčků ochlazujících controller module
- enclosures – kontrola konstrukcí udržujících několik komponent jako controller moduly, zdroje napájení nebo front-end a back-end routery a dále konstrukce udržujících komponenty diskového pole a dále konstrukce udržující komponenty jako jsou disky, zdroje napájení a I/O moduly
- expanders – kontrola komponent kontrolujících interakci mezi controller modulem a disky

Módy pro zařízení používající MIB tabulku FJDARY-E150.mib:

- cmandcomponents – kontrola modulů obsahujících CPU a hlavní paměť, paměti controller modulu a paměti flash-rom controller modulu
- enclosures – kontrola konstrukcí udržujících několik komponent jako controller moduly, napěťové zdroje napájení nebo front-end a back-end routery a dále konstrukce udržující komponenty jako jsou disky, zdroje napájení a I/O moduly
- batteries – kontrola kontrolních jednotek ovládajících baterie a baterií sloužících pro uchování energie pro případ výpadku proudu
- bud – kontrola velmi rychlých stabilních SSD disků, na nichž je uložený operační systém a konfigurace a také sloužících pro dočasné ukládání dat
- cpsu – kontrola zdrojů napájení controller enclosures
- iom – kontrola komponent kontrolujících interakci mezi controller modulem a disky
- fem – kontrola expanderů, které slouží jako interní interfejsy mezi I/O moduly a disky
- pfm – kontrola rozšířených pamětí pro systémové cache

8.4 Implementace pluginu

V následující části budou uvedeny pouze nejdůležitější části vytvořeného pluginu pro monitoring diskových polí. Celý plugin je dostupný v příloze. Jednotlivé části kódu, které mají svůj ekvivalent ve vývojovém diagramu popsaném v kapitole *Návrh řešení*, jsou uvozeny názvem příslušného bodu vývojového diagramu, jehož funkcionalita je v dané části kódu naimplementována.

Plugin začíná definicí datových struktur. V následujícím kódu je vytvořena datová struktura, do které se budou v průběhu kontrol stavů komponent zapisovat zjištěné výsledky, které se budou zařazovat do příslušné datové struktury, podle

stavu komponenty. Například bude-li disk vykazovat stav *broken*, zařadí se zjištěný výsledek o tomto disku do datové struktury CRITICAL.

Definování chybových stavů komponent

```
my %states = (                          # structure for storing informations
    about components
        OK => {},
        WARNING => {},
        CRITICAL => {},
        INFORMATIVE => {}
);
```

Dále jsou zde definovány struktury udržující konverzní vztah mezi číselnými hodnotami stavů komponent, které budeme číst z MIB záznamů zařízení, a jejich textovou interpretací. Zjistíme-li tedy z MIB záznamu o stavu komponenty hodnotu *1*, díky této struktuře ji následně dokážeme interpretovat jako stav *normal*. Zmínil bych však datovou strukturu udržující hodnotu specifické části OID vedoucí k záznamům o určitém typu komponenty a hodnotu specifické části OID vedoucí k záznamu, který udržuje informace o sériovém čísle nebo identifikačním čísle komponenty. Význam těchto hodnot si uvedeme na příkladu. Vezmeme-li záznam o typu komponenty *Controller module*, pak v tomto záznamu nalezneme dvě hodnoty. Hodnota *baseOID* reprezentuje část OID, definující postavení záznamu udržující informace o tomto typu komponenty ve struktuře MIB záznamů. Dosadíme-li před tuto hodnotu celé OID definující cestu k části MIB záznamů, ve které jsou uchovávané záznamy o fyzických komponentách daného zařízení, získáme celou cestu ve struktuře MIB záznamů vedoucí k typu komponenty *Controller module*. Hodnota *stateValueOID* reprezentuje část OID, definující záznam, ve kterém jsou informace o sériovém čísle nebo identifikačním čísle dané komponenty. Připojením této hodnoty k OID definující typ komponenty získáme celé OID záznamu udržující informace o sériovém nebo identifikačním čísle. Tyto hodnoty jsou právě takto definovány v datové struktuře, abychom mohli později vytvořit obecnou funkci pro kontrolu stavů komponent.

```
# OIDs of device components and their identification codes - based on FJDARY-
    E60.mib and FJDARY-E101.mib
my %componentsOID = (
        controllerModule    => {baseOID =>'.1.',
                                stateValueOID => '2.1.7.',},
        channelAdapter      => {baseOID => '.2.',
                                stateValueOID => '2.1.2.'},
        cmMemory            => {baseOID => '.3.',
                                stateValueOID => '2.1.6.'},
        cmFlashRom          => {baseOID => '.4.',
                                stateValueOID => '2.1.2.'},
        scu                 => {baseOID => '.5.',
                                stateValueOID => '2.1.7.'},
        controllerEnclosure => {baseOID => '.6.',
                                stateValueOID => '2.1.2.'},
        deviceEnclosure     => {baseOID => '.7.',
                                stateValueOID => '2.1.4.'},
```

```
        expander            => {baseOID => '.8.',
                                stateValueOID => '2.1.6.'},
        powerSupplyUnit     => {baseOID => '.9.',
                                stateValueOID => '2.1.5.'},
        inletThermalSensor  => {baseOID => '.10.',
                                stateValueOID => '2.1.2.'},
        thermalSensor       => {baseOID => '.11.',
                                stateValueOID => '2.1.2.'},
        disk                => '.12.',
        lanPort             => '.13.',

);
```

Po definici datových struktur se již dostáváme k samotné funkcionalitě pluginu. Nejprve se musí převzít hodnoty argumentů zadaných při volání skriptu, které jsou nezbytné pro další operace. Obsahem argumentů jsou hodnoty určující v jakém módu se má plugin spustit. Jedná se o IP adresu zařízení, SNMP komunitu a SNMP verzi. Převzetí argumentů je realizováno v následujícím kódu.

Definování SNMP atributů

```
if (!@ARGV) {
        printUsage();
        exit(3);
};

while (@ARGV) {
        switch(shift){
                case ['--mode','-M']
                        {
                                $inputArguments{mode} = lc(shift);
                        }
                case ['--hostname','-H']
                        {
                                $inputArguments{hostname} = shift;
                        }
                case ['--version','-V']
                        {
                                $inputArguments{version} = shift;
                        }
                case ['--community','-C']
                        {
                                $inputArguments{community} = shift;
                        }
        };
};
```

Argumenty se do pluginu předávají v poli *@ARGV*. V případě, že je toto pole prázdné, plugin vrací hodnotu *UNKNOWN* a vypisuje nápovědu pro použití skriptu. Pokud jsou nějaké argumenty zadány, postupně se berou z pole a na základě jejich hodnoty se rozhoduje o jejich přiřazení. Po úspěšném převzetí argumentů pluginu následuje návázaní SNMP spojení se zařízením. Implementace je v následujícím kódu.

Navázání SNMP spojení se zařízením, Předání stavu UNKNOWN monitorovacímu systému, Ukončení pluginu

```
# establish a~connection with device
my ($session, $error) = Net::SNMP->session(
   -hostname  =>  $inputArguments{hostname},
   -version   =>  $inputArguments{version},
   -community =>  $inputArguments{community},
);

# check if connection is succesfully established
if (!defined $session) {
   printf "ERROR: %s.\n", $error;
   printUsage();
   exit 3;
}
```

Spojení je vytvářeno pomocí funkce modulu Net::SNMP, které předáme hodnoty potřebné pro připojení, které jsme převzali z argumentů skriptu. V případě, že se spojení nepodaří navázat, je plugin ukončen výpisem chyby spojení, nápovědy použití skriptu a pomocí exit kódu vrací hodnotu interpretovanou jako stav *UNKNOWN*.

Po navázání SNMP spojení se zařízením můžeme začít se čtením informací. Jelikož jsou pole založena na třech různých MIB tabulkách, musíme nejprve zjistit, jaký typ tabulky zařízení používá. Způsob získání OID definující použitou MIB tabulku je v následujícím kódu.

Získání základního OID zařízení pro volbu MIB tabulky

```
# base nsp OID for MIB table type check
my $nspNumberOID = '1.3.6.1.4.1.211.1.21.1';

# will be defined further, based on MIB table version
my $currentDeviceOID;

#requesting base device OID
my $result = $session->get_next_request( -varbindlist =>  [ $nspNumberOID ],);
my $full_OID;
foreach my $OID (keys %{$result}) {
                $full_OID = $OID;
}


$full_OID =~ /^((\d*\.)){10}(\d*)(\.\d*)*/g;
#base part of device OID which will be used further
$currentDeviceOID = $nspNumberOID . '.' . $3;
#OID specifying type of device MIB table
my $specificDeviceOID = $3;
```

Do proměnné *nspNumberOID* si uložíme OID hodnotu uzlu, který je přímým předchůdcem hodnoty určující typ MIB tabulky. Funkcí *get_next_request* získáme hodnotu ukazující na první list stromové struktury MIB záznamů. Tuto hodnotu si uložíme a následně vyseparujeme číslo určující typ používané MIB tabulky.

Nyní již víme, kterou MIB tabulku zařízení používá a můžeme přejít ke čtení požadovaných informací. Nejprve si tedy zjistíme základní informace o diskovém poli. V následujícím kódu je uvedeno získání těchto informací.

Získání řetězce obsahujícího základní informace o zařízení, Separace typu zařízení a sériového čísla

```
#parsing infromations about specific disk array
my $machineInfoOID = $currentDeviceOID . '.1.1.0';

$result = $session->get_request(-varbindlist => [ $machineInfoOID ],);

my %arrayInfo;

$arrayInfo{type} = substr(${$result}{$machineInfoOID},0,2);
$arrayInfo{serie} = substr(${$result}{$machineInfoOID},2,12);
$arrayInfo{serie} =~ s/#*//g;
$arrayInfo{model} = substr(${$result}{$machineInfoOID},14,12);
$arrayInfo{model} =~ s/#*//g;
$arrayInfo{serialNumber} = substr(${$result}{$machineInfoOID},28,12);
$arrayInfo{serialNumber} =~ s/#*//g;
```

Do proměnné *machineInfoIOD* vložíme hodnotu OID odkazující na záznam udržující informace o daném diskovém poli. Tuto hodnotu následně použijeme ve funkci SNMP *get_request* pro získání řetězce obsahujícího požadované informace. V řetězci se mimo typu pole, modelu pole a jeho sériového čísla nacházejí i pro nás nepodstatné informace. Proto jsou v následujících krocích z řetězce vybrány pouze potřebné záznamy. Následně si do proměnné *overallStatus* uložíme doposud získané informace, které budou předávány jako součást výstupu všech módů pluginu.

Po získání základních hodnot následuje získání celkového stavu vykazovaného zařízením. Jeho získání je v následujícím kódu.

```
#getting overall status of device
$result = $session->get_next_request(-varbindlist => [ $currentDeviceOID . '.6'
    ],);
my $overallDeviceState = uc($unitStatus{${$result}{$currentDeviceOID . '.6.0'
    }});

my $overallStatus = $arrayInfo{serie} . ' - ' . $arrayInfo{model} . ' with
    serial number: ' . $arrayInfo{serialNumber};
```

Po získání požadované informace si její převedenou číselnou hodnotu do textové podoby uložíme do proměnné *overallDeviceState*. Tato hodnota bude využita při pozdějším výstupu pro mód pluginu *overall*.

Nyní již následuje kontrola požadovaných komponent. Algoritmus je realizovám následujícím způsobem. Nejprve se na základě MIB tabulky rozhodneme pro správnou větev úkonů pomocí přepínače. Provádí-li plugin kontrolu zařízení pracující na MIB tabulce FJDARY-E60.mib nebo FJDARY-E101.mib, pak běh programu pokračuje větví *case [60,101]*. Následně se opět pomocí přepínače upřesní, které komponenty chceme v konkrétním běhu programu kontrolovat. Program se zde rozhoduje na základě módu zvoleného uživatelem. Po výběru módu se již volají funkce,

které provádějí samotnou kontrolu daných typů komponent, které jsou předmětem kontroly v příslušném módu pluginu. Tyto funkce budou popsány posléze. V následujícím kódu je nastíněná část rozhodovací logiky pro výběr kontroly požadovaných komponent a volání funkce.

Volba módu pluginu, Zvolení typu komponenty pro kontrolu, Ukončení kontroly všech typů komponent

```
switch($specificDeviceOID){
 case [60,101]
  {
   switch($inputArguments{mode}){
    case 'overall'
         {
          checkComponent($currentDeviceOID . $informationTypes{physical},
              $componentsOID{controllerModule});
          checkComponent($currentDeviceOID . $informationTypes{physical},
              $componentsOID{channelAdapter});
          checkComponent($currentDeviceOID . $informationTypes{physical},
              $componentsOID{cmMemory});
          checkComponent($currentDeviceOID . $informationTypes{physical},
              $componentsOID{cmFlashRom});
          checkComponent($currentDeviceOID . $informationTypes{physical},
              $componentsOID{scu});
          checkComponent($currentDeviceOID . $informationTypes{physical},
              $componentsOID{controllerEnclosure});
          checkComponent($currentDeviceOID . $informationTypes{physical},
              $componentsOID{deviceEnclosure});
          checkComponent($currentDeviceOID . $informationTypes{physical},
              $componentsOID{expander}); #NOT DEFINED SN/ID IN MIB
          checkComponent($currentDeviceOID . $informationTypes{physical},
              $componentsOID{powerSupplyUnit});
          checkComponent($currentDeviceOID . $informationTypes{physical},
              $componentsOID{inletThermalSensor});
          checkComponent($currentDeviceOID . $informationTypes{physical},
              $componentsOID{thermalSensor});
          checkDisks($currentDeviceOID . $informationTypes{physical} .
              $componentsOID{disk});
          $overallStatus = 'Overall status of ' . $overallStatus . ' is ' .
              $overallDeviceState;
         }
        case 'sensors'
         {
          checkComponent($currentDeviceOID . $informationTypes{physical},
              $componentsOID{inletThermalSensor});
          checkComponent($currentDeviceOID . $informationTypes{physical},
              $componentsOID{thermalSensor});
          getCount('sensors');
         }
        .
        .
        .
```

Celý proces kontroly fyzických komponent diskového pole je postaven na volání příslušných funkcí, které provedou kontrolu zvoleného typu komponenty. Funkce

checkComponent byla vytvářena s cílem umožnit její používání na jakýkoliv typ komponenty, mimo disků a lan portů, které mají odlišné stavové hodnoty a nelze pro ně tuto obecnou funkci implementovat. Výběr typu komponenty budeme volit pouze změnou parametru funkce, která je v následujícím kódu.

Získání počtu komponent zvoleného typu, Získání stavu komponenty, Vyhodnocení stavu komponenty, Uložení stavu do hlášení OK - WARNING - CRITICAL, Průchod algoritmu přes všechny komponenty daného typu, Ukončení kontroly daného typu komponenty

```
# main function - designed for use on any type of physical component with
    global state definitions
sub checkComponent(){
 #incoming defined base OID for component
 my $baseOID = $_[0] . $_[1]{baseOID};
 $result = $session->get_request(-varbindlist => [$baseOID . '1.0'], );
 # retrieving count of specific components defined on device
 my $amount = ${$result}{$baseOID . '1.0'};
 # set default reference point for creating reports about components from
     arrays based on FJDARY-E60.mib and FJDARY-E101.mib
 my $componentsReportReferencePoint = \%componentsReport;
 # if the function is called on monitoring components from arrays based on
     FJDARY-E150.mib, set alternative reports for use
 if ($specificDeviceOID == 150) {
  $componentsReportReferencePoint = \%E150componentsReport;
 };
 # requesting table of informations about specific components
 $result = $session->get_table(-baseoid =>  $baseOID . '2.1');
 my $notDefinedCount = 0;
 # checking the state of each specific component
 for (my $i=0;$i<$amount;$i++) {
  # based on component status creating report
  switch(${$result}{$baseOID . '2.1.3.' . $i}){
   case 1                {
         $states{OK}{%$componentsReportReferencePoint{$_[1]{baseOID}} . ${
             $result}{$baseOID . $_[1]{stateValueOID} . $i}} = $componentStatus{
             ${$result}{$baseOID . '2.1.3.' . $i}};
   }
   case 2                {
    $states{CRITICAL}{%$componentsReportReferencePoint{$_[1]{baseOID}} . ${
        $result}{$baseOID . $_[1]{stateValueOID} . $i}} = $componentStatus{${
        $result}{$baseOID . '2.1.3.' . $i}};
   }
   case [3,5,6]          {
         $states{WARNING}{%$componentsReportReferencePoint{$_[1]{baseOID}} . ${
             $result}{$baseOID . $_[1]{stateValueOID} . $i}} = $componentStatus{
             ${$result}{$baseOID . '2.1.3.' . $i}};
   }
   case 4                {
    if (${$result}{$baseOID . $_[1]{stateValueOID} . $i} =~ /\a+|\d+/g){
          $states{INFORMATIVE}{%$componentsReportReferencePoint{$_[1]{baseOID}}
              . ${$result}{$baseOID . $_[1]{stateValueOID} . $i}} =
             $componentStatus{${$result}{$baseOID . '2.1.3.' . $i}};
         } else {
          $notDefinedCount++;
```

```
            $states{INFORMATIVE}{%$componentsReportReferencePoint{$_[1]{baseOID}}
                .'not defined(' . $notDefinedCount . ')'} = $componentStatus{${
                $result}{$baseOID . '2.1.3.' . $i}};
            }
    }
   };
  };
};
```

Funkce začíná převzetím hodnot z parametrů. Do proměnné *baseOID*, která reprezentuje OID hodnotu odkazující na daný typ komponenty, který se bude kontrolovat, se převezme první parametr, který představuje základní OID monitorovaného zařízení s přidanou hodnotou definující OID hodnotu fyzických komponent v MIB tabulce a následně se spojí s parametrem druhým, který na základě vybraného typu komponenty přiřadí část OID reprezentující tento typ komponenty. Tím získáme celé OID vedoucí celou strukturou MIB záznamů právě ke zvolenému typu komponenty. Funkce pokračuje získáním hodnoty z MIB záznamu zařízení informujícího o možném počtu komponent daného typu. Dále se na základě hodnoty, určující jaký typ MIB tabulky diskové pole používá, zvolí příslušná datová struktura udržující předdefinované textové výstupy pro dané typy komponent. Po tomto kroku následuje získání tabulky obsahující všechny definované informace o všech komponentách daného typu. Tabulkou postupně procházíme a kontrolujeme stav každé z komponent daného typu. Na základě hodnoty vykazovaného stavu každé z komponent jsou vytvářena jednotlivá hlášení o každé kontrolované komponentě a tato hlášení jsou přiřazována do příslušné datové struktury podle vykazovaného stavu komponenty.

Dále uvedu funkce na zpracování stavů disků a lan portů, které jsou uvedené v následujícím kódu.

```
# function for checking component - Disks
sub checkDisks(){
 #incoming defined base OID for disks
 my $baseOID = shift;
 $result = $session->get_request(-varbindlist => [$baseOID . '1.0'], );
 # retrieving count of disks defined on device
 my $amount = ${$result}{$baseOID . '1.0'};
 # requesting table of informations about disks
 $result = $session->get_table(-baseoid => $baseOID . '2.1' );
 # checking the state of each disk
 for (my $i=0;$i<$amount;$i++) {
  # based on disk status creating report
  switch(${$result}{$baseOID . '2.1.3.' . $i}){
   case [1,5,6,7,65,66,69]
        {
         $states{OK}{'Disk, ID: ' . ${$result}{$baseOID . '2.1.2.' . $i}} =
             $diskState{${$result}{$baseOID . '2.1.3.' . $i}};
        }
   case [4,67]  {
         $states{WARNING}{'Disk, ID: ' . ${$result}{$baseOID . '2.1.2.' . $i}}
              = $diskState{${$result}{$baseOID . '2.1.3.' . $i}};
        }
   case [2,64]  {
```

```
            $states{CRITICAL}{'Disk, ID: ' . ${$result}{$baseOID . '2.1.2.' . $i
                }} = $diskState{${$result}{$baseOID . '2.1.3.' . $i}};
            }
        case [3,68]  {
            $states{INFORMATIVE}{'Disk, ID: ' . ${$result}{$baseOID . '2.1.2.' .
                $i}} = $diskState{${$result}{$baseOID . '2.1.3.' . $i}};
            }
    };
  };
};

#function for checking lan ports
sub checkLanPorts(){
 #incoming defined base OID for lan ports
 my $baseOID = shift;
 $result = $session->get_request(-varbindlist => [$baseOID . '1.0'], );
 # retrieving count of lan ports defined on device
 my $amount = ${$result}{$baseOID . '1.0'};
 # requesting table of informations about lan ports
 $result = $session->get_table(-baseoid => $baseOID . '2.1' );
 # checking the state of each lan port
 for (my $i=0;$i<$amount;$i++) {
  # based on lan port MNT status creating report
  switch(${$result}{$baseOID . '2.1.2.' . $i}){
    case 1              {
        $states{OK}{'LAN port - MNT, ID:' . $i} = $lanState{${$result}{$baseOID
            . '2.1.2.' . $i}};
    }
        case [4,7]      {
        $states{INFORMATIVE}{'LAN port - MNT, ID:' . $i} = $lanState{${$result
            }{$baseOID . '2.1.2.' . $i}};
    }
        case 6          {
         $states{CRITICAL}{'LAN port - MNT, ID:' . $i} = $lanState{${$result}{
             $baseOID . '2.1.2.' . $i}};
    }
   };
  # based on lan port RMT status creating report
  switch(${$result}{$baseOID . '2.1.3.' . $i}){
    case 1              {
        $states{OK}{'LAN port - RMT, ID:' . $i} = $lanState{${$result}{$baseOID
            . '2.1.3.' . $i}};
    }
        case [4,7]      {
        $states{INFORMATIVE}{'LAN port - RMT, ID:' . $i} = $lanState{${$result
            }{$baseOID . '2.1.3.' . $i}};
    }
        case 6          {
        $states{CRITICAL}{'LAN port - RMT, ID:' . $i} = $lanState{${$result}{
            $baseOID . '2.1.3.' . $i}};
    }
  };
};
```

Tyto funkce pracují na stejném principu jako funkce *checkComponent*, rozdíl je pouze v tom, že tyto fyzické komponenty mají stanovené rozdílné hodnoty vykazo-

vaných stavů a proto musejí být zpracovány samostatně.

Zbývá poslední funkce tohoto pluginu. Tato funkce je využívána při spouštění pluginu v módech definujících kontrolu pouze vybraného typu komponenty. Jejím smyslem je po provedení kontrol stavů jednotlivých komponent vybraného typu zjistit, kolik komponent daného typu je na diskovém poli a kolik komponent z celkového počtu komponent daného typu je ve stavu OK, WARNING a CRITICAL. Tyto informace jsou následně přidány k celkovému výstupu pluginu. Funkce je v následujícím kódu.

```
sub getCount(){
#function used in modes different from overall
#counting amount of components monitored in specific mode and creating part of
    output which contains information about amount of TOTAL/OK/WARNING/CRITICAL
        $totalAmount = scalar(keys %{$states{OK}}) + scalar(keys %{$states{
            WARNING}}) + scalar(keys %{$states{CRITICAL}});
        $overallStatus .= ', ' . shift . ' T/O/W/C - ' . $totalAmount . '/' .
            scalar(keys %{$states{OK}}) . '/' . scalar(keys %{$states{WARNING
            }}) . '/' . scalar(keys %{$states{CRITICAL}});
};
```

Po provedení kontrol stavů všech komponent všech typů zvolených při spuštění následuje vyhodnocení získaných výsledků. To je realizováno na základě obsahu datových struktur udržujících příslušná hlášení o stavech komponent. Princip je zobrazen v následujícím kódu.

Určení výstupních hodnot, Výpis typu pole, sériového čísla a OK stav, Výpis typu pole, sériového čísla a komponent ve stavu WARNING, Výpis typu pole, sériového čísla a komponent ve stavu CRITICAL, Výpis typu pole, sériového čísla a komponent ve stavu CRITICAL a WARNING, Ukončení pluginu

```
if ((scalar(keys %{$states{WARNING}}) != 0) && (scalar(keys %{$states{CRITICAL
    }}) != 0)){
 $overallStatus .= ', States: CRITICAL: ';
 foreach $key (keys %{$states{CRITICAL}}){
  $overallStatus .= $key . ' - ' . $states{CRITICAL}{$key} . ' / ';
 };
 $overallStatus .= 'WARNING: ';
 foreach $key (keys %{$states{WARNING}}){
  $overallStatus .= $key . ' - ' . $states{WARNING}{$key} . ' / ';
 };
 print $overallStatus . "/n";
 exit 2;
} elsif (scalar(keys %{$states{WARNING}}) != 0){
 $overallStatus .= ', States: ';
 foreach $key (keys %{$states{WARNING}}){
  $overallStatus .= $key . ' - ' . $states{WARNING}{$key} . ' / ';
 };
 print $overallStatus . "/n";
 exit 1;
 } elsif (scalar(keys %{$states{CRITICAL}}) != 0) {
  $overallStatus .= ', States: ';
  foreach $key (keys %{$states{CRITICAL}}){
```

```
    $overallStatus .= $key . ' - ' . $states{CRITICAL}{$key} . ' / ';
  };
  print $overallStatus . "/n";
  exit 2;
} elsif ((scalar(keys %{$states{WARNING}}) == 0) && (scalar(keys %{$states{
    CRITICAL}}) == 0)){
  print $overallStatus . "/n";
  exit 0;
};
```

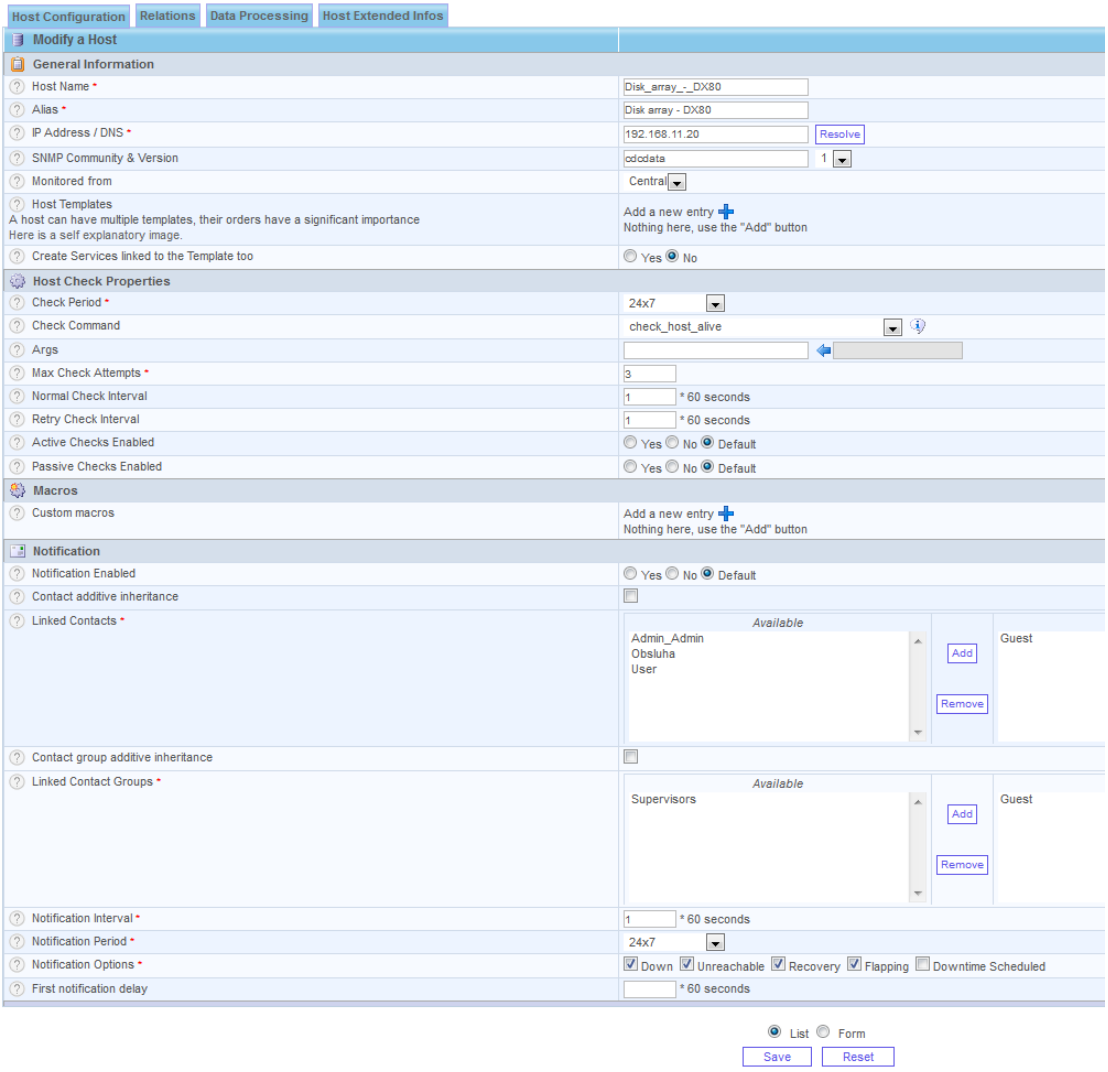
V případě, že datové struktury udržující hlášení o komponentách ve stavech WARNING nebo CRITICAL obsahují nějaké záznamy, pak se vyhodnotí celkový stav zařízení podle obsazenosti příslušné datové struktury. V případě že je obsazena pouze struktura udržující hlášení o komponentách ve stavu WARNING, pak je pomocí exit status kódu pluginu s hodnotou 1 předáván stav WARNING doplněný o informace o zařízení a komponentách tento stav způsobující. Obdobně je to řešeno u stavu CRITICAL, kdy se tento stav předává hodnotou exit status kódu pluginu 2. V případě, že jsou datové struktury udržující hlášení o zařízení ve stavech WARNING i CRITICAL prázdné, vrací se exit status kódem pluginu s hodnotou 0 stav OK a doplňující informace o zařízení. Předáním hodnot monitorovacímu systému plugin končí.

8.5 Nasazení pluginu do monitorovacího systému Centreon

Následující část se zabývá provedením nastavení monitorovacího systému Centreon, čili provedení všech nezbytných akcí pro docílení nasazení pluginu na monitorování vybraného diskového pole.

8.5.1 Vytvoření hosta

Abychom mohli vybrané pole monitorovat, musíme ho nejprve v monitorovacím systému nadefinovat jako hosta. Ve webovém prostředí Centreonu vybereme záložku *Configuration* a podzáložku *Hosts*. Vypíše se nám tabulka, ve které nalezneme všechny definované hosty v monitorovacím systému. Nad tabulkou klikneme na odkaz *Add*, čímž se dostaneme do definice nového hosta. Formulář pro vytvoření nového hosta můžeme vidět na obrázku č. 18.



Obrázek 18: Formulář pro definici hosta v monitorovacím systému Centreon

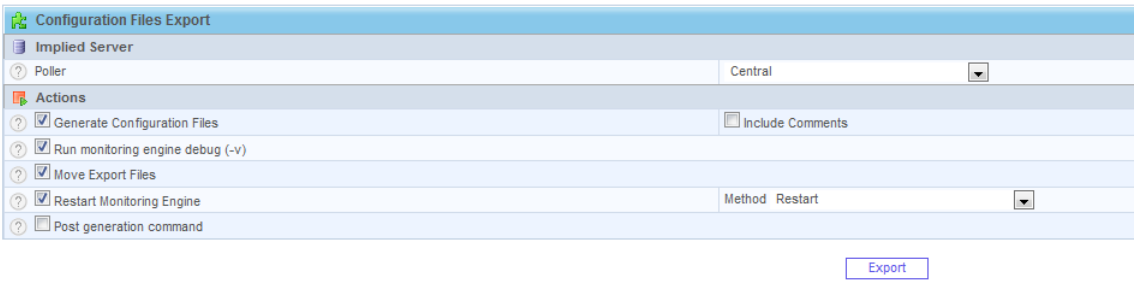
Nejprve definujeme *Host name*, pod kterým se nám tento host bude zobrazovat v monitorovacím systému. Dále definujeme jeho *Alias* a *IP address*. Abychom se mohli na zařízení dotazovat protokolem SNMP, musíme dále určit *SNMP Community & Version*. Ostatní položky v části *General Information* ponecháme na přednastavených hodnotách.

V části *Host Check Properities* budeme nastavovat, jakým způsobem bude host monitorován. Nejprve určíme v *Check Period*, v jakém časovém rozmezí chceme monitoring provádět. Dále můžeme již v této fázi nasadit nějaký plugin pro provádění kontroly zařízení. V této fázi bych doporučoval předdefinovaný plugin *check_host_alive*, který nám bude kontrolovat, zda příjmáme ze zařízení odezvu. Další položkou, kterou je zapotřebí vyplnit, je *Max Check Attempts*, do které vložíme například hodnotu 3. Tato položka nám udává, kolik pokusů má monitorovací systém na dotázání zařízení předtím, než přepne ukazatel informující o stavu za-

řízení z hodnoty OK na WARNING nebo CRITICAL. Zbylé hodnoty v této části ponecháme na výchozím nastavení.

Část *Macros* ponecháme tak jak je a přejdeme na část *Notification*. V této části určujeme, jak má monitorovací systém reagovat na změny stavů zařízení a informovat o takových událostech uživatele. V částech *Linked Contacts* a *Linked Groups* určujeme, kterým uživatelům, popřípadě skupině se budou hlášení odesílat. Můžeme využít předdefinovaného uživatele a skupiny *Guest* nebo si vytvořit kontakt vlastní. Dále nastavíme *Notification Interval*, který nám určuje, jak často má být hlášení odesíláno. Následuje *Notification Period*, zde určujeme, v jakém časovém období je odesílání hlášení povoleno a *Notification Options*, kde vybereme stavy, o kterých se bude hlášení odesílat. Ostatní políčka ponecháme tak, jak jsou a klikneme na tlačítko *Save*. Tím jsme úspěšně dokončili definici nového hosta.

Aby se nám nově vytvořený host promítl do monitorovacího systému, je zapotřebí jej restartovat. To se provádí v záložce *Configuration* a podzáložce *Monitoring Engines*. Zaklikneme možnosti tak, jak jsou na obrázku č. 19 a klikneme na tlačítko *Export*.

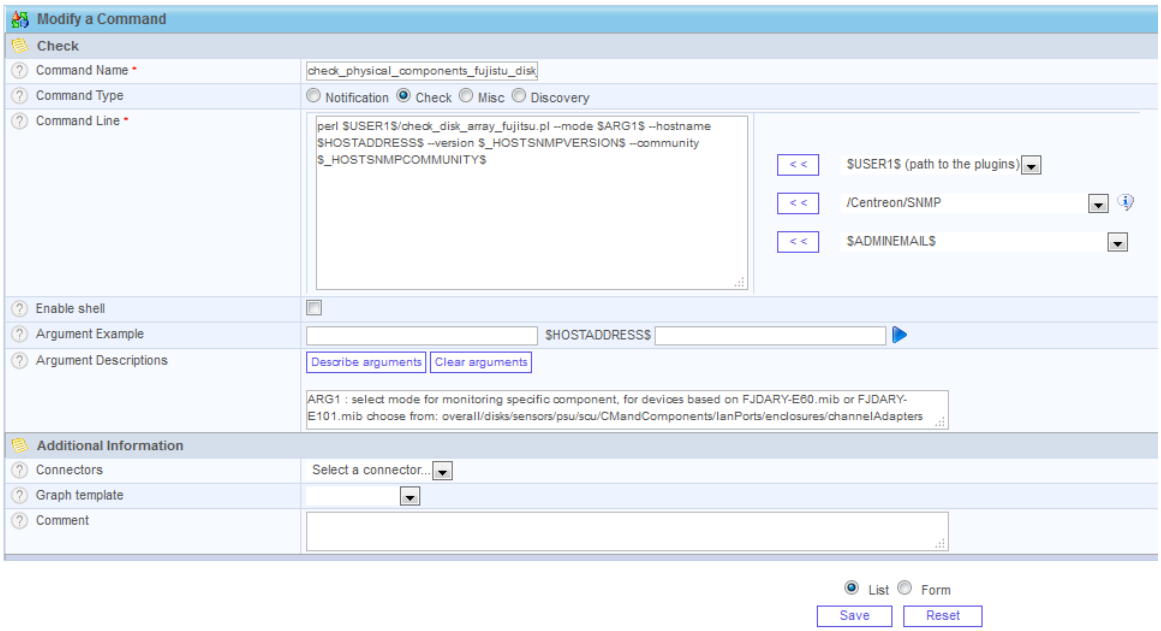


Obrázek 19: Prostředí pro aktualizace konfigurací monitorovacího systému Centreon

Tento krok je třeba provést po každém zásahu do monitorovacího systému, aby se tyto změny promítly.

8.5.2 Vytvoření příkazu

Abychom mohli používat vytvořený plugin, musíme v monitorovacím systému nejprve nadefinovat příkaz, který bude monitorovací systém spouštět při vznesení požadavku na kontrolu fyzických komponent diskových polí. Tabulku již existujících příkazů najdeme v záložce *Configuration* a podzáložce *Commands*. Nad tabulkou opět najdeme odkaz *Add*, kterým se dostaneme do definice nového příkazu. Formulář pro vytvoření nového příkazu vidíme na obrázku č. 20.



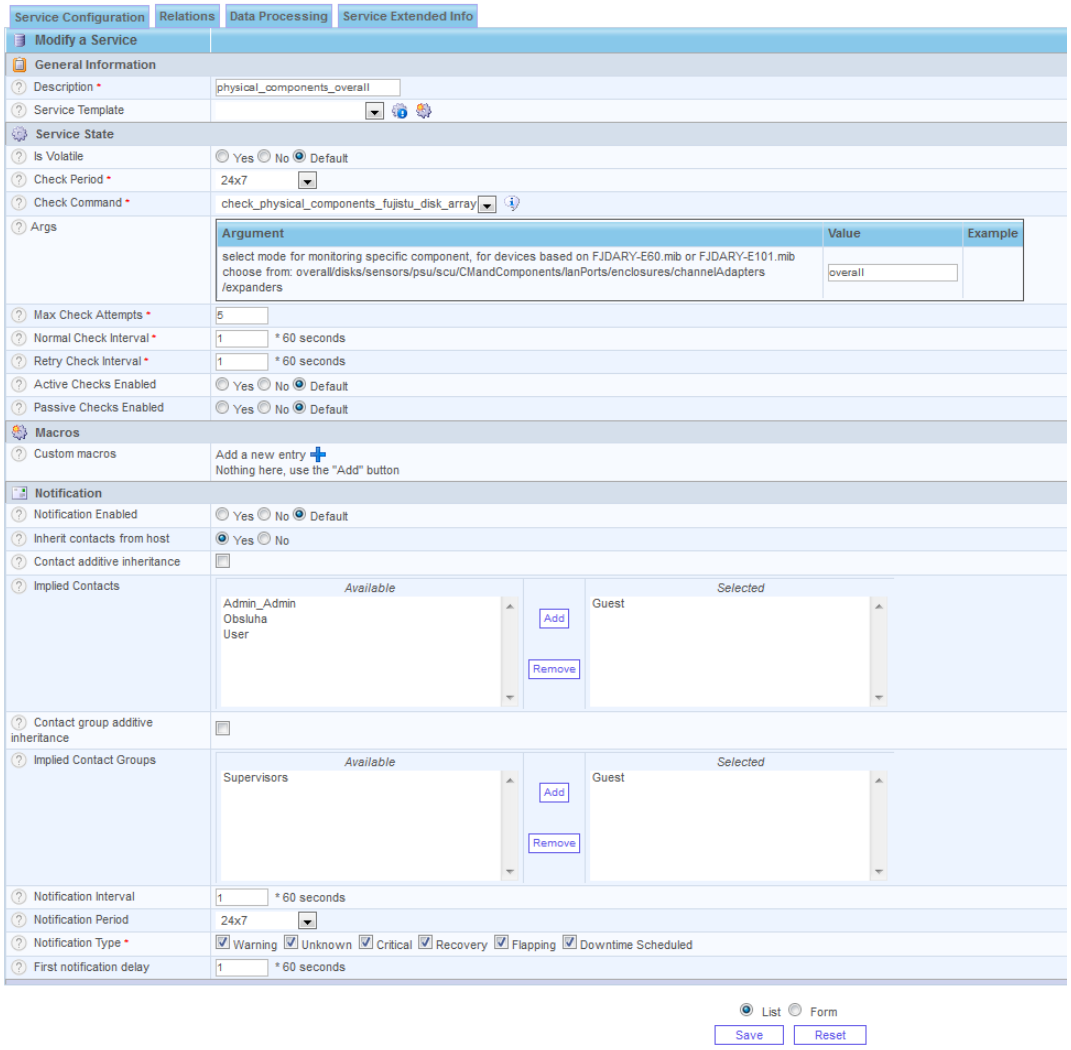
Obrázek 20: Formulář pro definici příkazu v monitorovacím systému Centreon

Začneme pojmenováním příkazu v políčku *Command Name*. Pole *Command Type* ponecháme nastaveno na hodnotu *Check* a pokračujeme nastavením pole *Command Line*. Toto pole představuje příkazový řádek, ve kterém se spouští plugin. Definujeme zde tedy příkaz pro spuštění samotného pluginu. Začneme klíčovým slovem *perl*, které interpretuje plugin naprogramovaný v programovacím jazyku Perl. Z první rolovací nabídky, která je po pravé straně od tohoto pole, vybereme hodnotu *$USER1$*, která nám definuje standardní cestu ke složce, ve které jsou uloženy všechny dostupné pluginy a šipkami ji vložíme do pole. Přidáme lomítko pro oddělení adresáře a z druhé rolovací nabídky vybereme název požadovaného pluginu, v našem případě *check_disk_array_fujitsu.pl* a šipkami jej vložíme do pole. Třetí rolovací nabídka nám představuje parametry, které se do pluginu mohou předávat. Do pole za dosud sestavený příkaz doplníme parametr *−−mode* a ve třetí rolovací nabídce vybereme hodnotu *$ARGn$*. Tato hodnota představuje uživatelem definovaný argument. Díky nastavení tohoto argumentu umožníme uživateli při pozdějším volání příkazu měnit módy pluginu. Pokračujeme přidáním parametru *−−hostname* a následně ze třetí rolovací nabídky vybereme hodnotu *$HOSTADDRESS$*. Tato hodnota představuje IP adresu hosta, na kterém budeme příkaz spouštět. IP adresa se předává přímo z dříve definovaných informací o hostovi, které jsme si definovali při vytváření hosta. Pokračujeme přidáním parametru *−−version* a výběrem hodnoty *$HOSTSNMPVERSION$* ze třetí rolovací nabídky. Tato hodnota představuje verzi SNMP protokolu, kterou host používá a tato informace se také předává přímo z informací o hostovi. Dále doplníme poslední potřebný parametr, kterým je *−−community* a doplníme ho hodnotou *$HOSTSNMPCOMMUNITY$*, která nám udává SNMP komunitu, kterou jsme si definovali na hostovi při jeho vytváření. Tato

hodnota se také přímo doplňuje z informací o hostovi. Tímto máme příkaz pro spuštění pluginu kompletně nadefinovaný. V poli *Argument Description* můžeme popsat námi definovaný argument pro lepší interpretaci významu argumentu při následném použití příkazu. Kliknutím na tlačítko *Save* vytvoříme příkaz a provedeme restartování monitorovacího systému.

8.5.3 Vytvoření monitorovací služby

Finální krok pro nasazení pluginu na požadovaném zařízení je vytvoření monitorovací služby. Tuto možnost nalezneme v záložce *Configuration* a podzáložce *Services*, kde se nám vypíše tabulka obsahující všechny vytvořené služby. Klikneme na odkaz *Add* a dostaneme se do definice nové služby. Formulář pro vytvoření nové monitorovací služby je na obrázku č. 21.



Obrázek 21: Formulář pro definici služby v monitorovacím systému Centreon

Celý proces opět začíná pojmenováním služby v poli *Description*. Dále určíme v poli *Check Period* časové období, ve kterém se má služba provádět. Důležitou částí v definici nové služby je výběr příkazu, který bude služba spouštět. Tuto volbu provádíme v poli *Check Command*. Zde vybereme námi dříve definovaný příkaz *check_physical_components_fujitsu_disk_array*. Po výběru daného příkazu se nám otevře možnost pro doplnění argumentů příkazu. V tomto případě doplňujeme pouze jediný argument *mode*, protože ostatní argumenty příkazu se doplňují automaticky definovanými hodnotami hosta. Vybereme tedy mód, ve kterém chceme plugin spouštět, například *overall*. Pokračujeme políčkem *Max Check Attempts*, jehož význam byl vysvětlen při definici hosta, dále pole *Normal Check Interval*, které nám definuje interval, ve kterém se má spouštět plugin, pokud zařízení vykazuje stav OK. Dále políčko *Retry Check Interval*, jehož hodnota udává interval, ve kterém se má plugin spouštět v případě, že zařízení vykazuje stav jiný než stav OK. V části Notification postupujeme stejně jako při definování nového hosta. Ještě před uložením služby musíme přiřadit tuto službu na konkrétního hosta, na kterém bude tato služba realizována. V prostředí definice nové služby klikneme na záložku *Relations*. Otevře se nám nový formulář, ve kterém můžeme dokončit potřebná nastavení pro úspěšné spuštění služby. V části *Linked With Hosts* vybereme dříve definovaného hosta *Disk_array_-_DX80*, na kterém se bude tato služba spouštět a pomocí tlačítka se šipkami ho přesuneme do pravého okna. Po tomto úkonu klikneme na tlačítko *Save* a restartujeme monitorovací systém. Tímto máme úspěšně definovanou službu a po restartování monitorovacího systému budeme schopni vidět výsledky monitoringu zařízení v záložce *Monitoring*, podzáložce *Services* a v sekci *All services*.

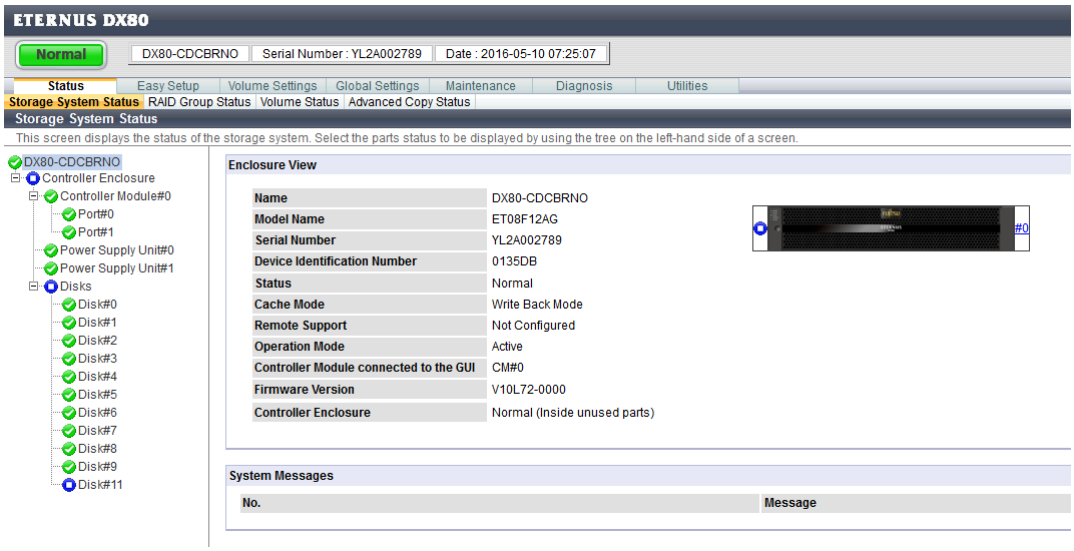
9 Testování

Následující kapitola představuje ověření funkcionality pluginu na reálných zařízeních. Předmětem reálného testování jsou tři disková pole, z nichž každé je založeno na jiné MIB tabulce. Při testování byly tedy pokryty všechny možnosti, pro které měl být plugin navrhnut a implementován. Jelikož je obtížné navodit na takovém zařízení stavy komponent ve stavech WARNING nebo CRITICAL, bylo testování na všech polích provedeno ve stavu, kdy všechny komponenty vykazují stav OK. Reálné navození stavu CRITICAL bylo dosaženo pouze u pole FUJITSU DX80, kde bylo při testování provedeno dočasné odpojení jednoho ze dvou zdrojů zařízení.

Aby byly pokryty i stavy CRITICAL a WARNING, které mohou komponenty v případě problémů vykazovat, byly pro demonstraci funkcionality pluginu tyto stavy a následná chybová hlášení uměle vyvolána úpravou zdrojového kódu pluginu při testovacím monitorování diskového pole FUJITSU ETERNUS DX80.

9.1 Testovací případ č. 1 – Testování komponent pole FUJITSU ETERNUS DX80 – stav OK

První testovací případ ověřuje správnou funkcionalitu pluginu na zařízení, které vykazuje bezproblémový stav. Na obrázku č. 22 je výpis z managmentu monitorovaného diskového pole, který dokazuje, že se pole nachází v bezproblémovém stavu. Na poli je špatně nastaven systémový čas, který je o hodinu pozadu oproti času reálnému. Proto dochází k hodinovému rozestupu mezi časovými údaji poskytovanými diskovým polem a monitorovacím systémem.



Obrázek 22: Reálně vykazované stavy komponent pole FUJITSU DX80 v managmentu pole – stav OK

Pro ověření funkcionality pluginu nahlédneme do přehledu služeb monitorovacího systému Centreon, ve kterém jsme si dříve definovali služby pro monitoring tohoto zařízení a ověříme, zda všechny tyto služby vykazují správné hodnoty. V tomto případě by tedy měly všechny služby vykazovat stav OK, jak je vidět na obrázku č. 23.

	S	Hosts	Services	Status	Duration	Last Check	Tries	Status information
		Disk_array_-_DX80	physical_components_channelAdapters	OK	1w 1d 1m 33s	10/05/2016 08:25:01	1/5 (H)	ETERNUSDXL - ET08F12AG with serial number: YL2A002789, channel adapters T/O/W/C - 1/1/0/0
			physical_components_CMandComponents	OK	1w 1d 1m 33s	10/05/2016 08:25:01	1/5 (H)	ETERNUSDXL - ET08F12AG with serial number: YL2A002789, CM & components T/O/W/C - 3/3/0/0
			physical_components_disks	OK	1w 1d 1m 33s	10/05/2016 08:25:01	1/5 (H)	ETERNUSDXL - ET08F12AG with serial number: YL2A002789, disks T/O/W/C - 11/11/0/0
			physical_components_enclosures	OK	1w 1d 1m 33s	10/05/2016 08:25:01	1/5 (H)	ETERNUSDXL - ET08F12AG with serial number: YL2A002789, enclosures T/O/W/C - 2/2/0/0
			physical_components_expanders	OK	1w 1d 1m 33s	10/05/2016 08:25:01	1/5 (H)	ETERNUSDXL - ET08F12AG with serial number: YL2A002789, expanders T/O/W/C - 1/1/0/0
			physical_components_lanPorts	OK	1w 1d 1m 33s	10/05/2016 08:25:01	1/5 (H)	ETERNUSDXL - ET08F12AG with serial number: YL2A002789, LAN ports T/O/W/C - 2/2/0/0
			physical_components_overall	OK	1w 17h 3m 9s	10/05/2016 08:25:02	1/5 (H)	Overall status of ETERNUSDXL - ET08F12AG with serial number: YL2A002789 is OK
			physical_components_psu	OK	1w 17h 3m 9s	10/05/2016 08:25:01	1/5 (H)	ETERNUSDXL - ET08F12AG with serial number: YL2A002789, power supply units T/O/W/C - 2/2/0/0
			physical_components_scu	OK	1w 1d 1m 33s	10/05/2016 08:25:01	1/5 (H)	ETERNUSDXL - ET08F12AG with serial number: YL2A002789, system capacitor units T/O/W/C - 1/1/0/0
			physical_components_sensors	OK	1w 1d 1m 33s	10/05/2016 08:25:01	1/5 (H)	ETERNUSDXL - ET08F12AG with serial number: YL2A002789, sensors T/O/W/C - 2/2/0/0

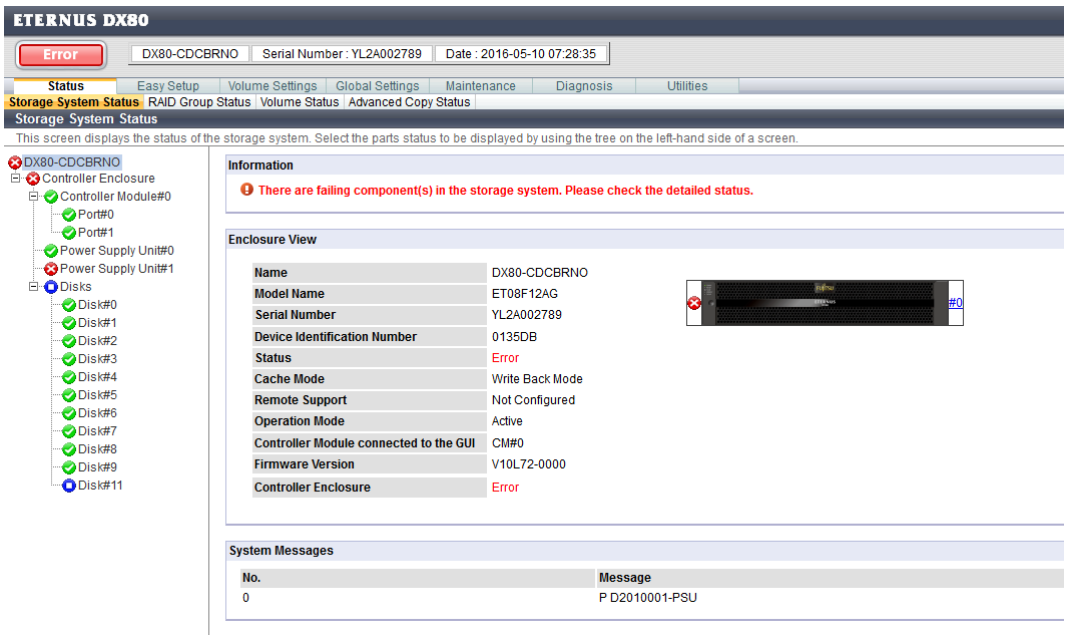
Obrázek 23: Reálně vykazované stavy komponent pole FUJITSU DX80 v monitorovacím systému – stav OK

Jednotlivé služby reprezentující použitelné módy pluginu, mimo stavů informujících o výsledku provedené kontroly (v tomto případě stav OK), vykazovaných na základě stavů kontrolovaných komponent, obsahují i doplňující hlášení. Tato hlášení v případě celkového stavu OK, služby reprezentující jakýkoliv mód mimo módu *overall*, obsahují informace o typu, modelu a sériovém čísle monitorovaného pole doplněné informací o typu komponenty, která se v jednotlivém módu kontrolovala a přehled o celkovém počtu komponent určitého typu a jejich rozložení do jednotlivých stavů, kde zkrácený zápis T/O/W/C znamená Total/Ok/Warning/Critical. V módě *overall* pak hlášení obsahuje informace o typu, modelu a sériovém čísle monitorovaného pole doplněné o celkový stav zařízení a v případě, že některá z komponent vykazuje chybový stav, je pak hlášení doplněno o informace o této komponentně a jejím chybovém stavu.

Na základě průběhu tohoto testovacího případu můžeme říct, že plugin funguje správně na polích používajících MIB tabulku FJDARY-E60.mib při kontrole komponent nevykazujících chybový stav.

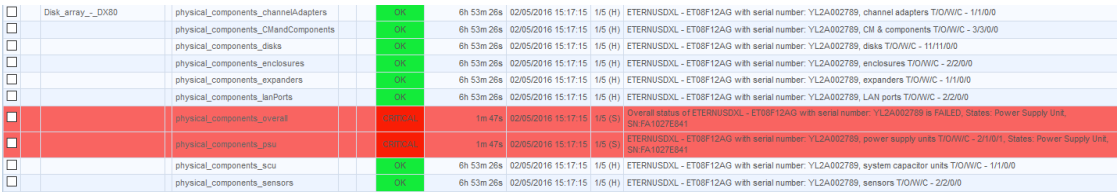
9.2 Testovací případ č. 2 – Testování komponent pole FUJITSU ETERNUS DX80 – stav CRITICAL

Na tomto poli jsme se v rámci druhého testovacího případu ve spolupráci s panem Ing. Holomkem snažili navodit některý z chybových stavů zařízení. Jak již bylo zmíněno, je složité tyto stavy navodit, jelikož to ve většině případů znamená poruchu dané komponenty. Podařilo se nám navodit stav CRITICAL v módu kontrolujícím stav zdrojů napájení zařízení. Tohoto stavu jsme dosáhli odpojením jednoho ze dvou zdrojů napájení zařízení. Výstup managmentu diskového pole po odpojení zdroje napájení je na obrázku č. 24.



Obrázek 24: Reálně vykazované stavy komponent pole FUJITSU DX80 v managmentu pole po odpojení zdroje napájení – stav CRITICAL

Pro ověření, zda plugin tuto chybu komponenty zachytil, se opět podíváme do přehledu služeb monitorovacího systému Centreon. Po provedení naplánované kontroly zařízení by se měl stav služby kontrolující zdroje napájení a stav služby kontrolující celkový stav zařízení změnit ze stavu OK na stav CRITICAL. Výsledek této změny je vidět na obrázku č. 25.



Obrázek 25: Reálně vykazované stavy komponent pole FUJITSU DX80 v monitorovacím systému po odpojení zdroje napájení – stav CRITICAL

Jak je vidět na obrázku č. 25, služba používaná pro kontrolu stavu zdrojů napájení je ve stavu CRITICAL, stejně jako služba kontrolující celkový stav zařízení. Obě tyto služby vykazují doplňující informaci o konkrétní komponentě, která tento stav vyvolala, doplněná o sériové číslo této komponenty pro snažší identifikaci. Po zapnutí zdroje napájení přecházejí zmiňované módy zpět do předešlého stavu OK.

Na základě druhého testovacího případu lze říci, že plugin správně reaguje na změny stavů komponent a vykazuje na základě zjištěných informací správné údaje. Vzhledem ke stejnému principu funcionality kontroly ostatních typů komponent lze říci, že plugin pracuje správně a splňuje požadavky na zařízeních používajících MIB tabulku FJDARY-E60.mib.

9.3 Testovací případ č.3 – Testování komponent pole FUJITSU ETERNUS DX60 S2 – stav OK

Otestování funkcionality pluginu na zařízeních používající MIB tabulku FJDARY-E101.mib bylo provedeno na produkčním zařízení zákazníka firmy CDC Data, s. r. o.. Vzhledem k tomu bylo testování prováděno pouze pokusným spuštěním pluginu v příkazovém řádku na serveru ve společné síti s tímto diskovým polem. Výsledky jsou na obrázku č. 26.

```
root@monitoring-moravia:/usr/lib/nagios/plugins# perl check_eternus.pl --hostname 192.168.10.140 --mode overall --version 1 --community cdcdata
Overall status of ETERNUSDXLS - ET062DDFU with serial number: 4311346587 is OK
root@monitoring-moravia:/usr/lib/nagios/plugins# perl check_eternus.pl --hostname 192.168.10.140 --mode sensors --version 1 --community cdcdata
ETERNUSDXLS - ET062DDFU with serial number: 4311346587, sensors T/O/W/C - 2/2/0/0
root@monitoring-moravia:/usr/lib/nagios/plugins# perl check_eternus.pl --hostname 192.168.10.140 --mode disks --version 1 --community cdcdata
ETERNUSDXLS - ET062DDFU with serial number: 4311346587, disks T/O/W/C - 8/8/0/0
root@monitoring-moravia:/usr/lib/nagios/plugins# perl check_eternus.pl --hostname 192.168.10.140 --mode psu --version 1 --community cdcdata
ETERNUSDXLS - ET062DDFU with serial number: 4311346587, power supply units T/O/W/C - 2/2/0/0
```

Obrázek 26: Reálně vykazované stavy komponent pole FUJITSU ETERNUS DX60 S2 v prostředí příkazové řádky

Pro otestování pluginu byly použity pouze některé módy, jejich princip funkcionality je však totožný a dá se říci, že pokud funguje jeden z módu, pak fungují i módy ostatní. Jak je vidět z výsledků v příkazovém řádku, plugin pracuje i na tomto zařízení správně a bez problémů.

Vzhledem k tomu, že funkcionalita pluginu pro kontrolu zařízení používajících MIB tabulku FJDARY-E60.mib a FJDARY-E101.mib je totožná, můžeme říci, že plugin pracuje správně a splňuje požadavky na zařízeních používajících zmíněné MIB tabulky.

9.4 Testovací případ č. 4 – Testování komponent pole FUJITSU ETERNUS DX100 S3 – stav OK

Otestování funkcionality pluginu na zařízeních používající MIB tabulku FJDARY-E150.mib, jehož implementace byla vytvořena pouze experimetálně bez možnosti průběžného testování při vývoji této části pluginu, bylo také provedeno na produkčním zařízení zákazníka firmy CDC Data, s. r. o.. Vzhledem k tomu bylo testování prováděno pouze pokusným spuštěním pluginu v příkazovém řádku na serveru ve společné síti s tímto diskovým polem. Výsledky jsou na obrázku č. 27.

```
root@mt01:/usr/lib/nagios/plperl ./check_eternus --mode overall --hostname 192.168.30.26 --version 1 --community cdcdata
Overall status of ETERNUSDXLS3 - ET103AU with serial number: 4601443408 is OK, States: LAN port - RMT, ID:0 - linkdown / LAN port - RMT, ID:1 - linkdown /
root@mt01:/usr/lib/nagios/plugins# perl ./check_eternus --mode disks --hostname 192.168.30.26 --version 1 --community cdcdata
ETERNUSDXLS3 - ET103AU with serial number: 4601443408, disks T/O/W/C - 16/16/0/0
root@mt01:/usr/lib/nagios/plugins# perl ./check_eternus --mode enclosures --hostname 192.168.30.26 --version 1 --community cdcdata
ETERNUSDXLS3 - ET103AU with serial number: 4601443408, enclosures T/O/W/C - 1/1/0/0
root@mt01:/usr/lib/nagios/plugins# perl ./check_eternus --mode cmandcomponents --hostname 192.168.30.26 --version 1 --community cdcdata
ETERNUSDXLS3 - ET103AU with serial number: 4601443408, CM & components T/O/W/C - 6/6/0/0
root@mt01:/usr/lib/nagios/plugins# perl ./check_eternus --mode sensors --hostname 192.168.30.26 --version 1 --community cdcdata
ETERNUSDXLS3 - ET103AU with serial number: 4601443408, sensors T/O/W/C - 2/2/0/0
```

Obrázek 27: Reálně vykazované stavy komponent pole FUJITSU ETERNUS DX100 S3 v prostředí příkazové řádky

Ve spolupráci s panem Ing. Holomkem jsme ve firmě CDC Data, s. r. o. alespoň dokázali tuto experimentální implemetaci otestovat na produkčním zařízení jednoho

z jejich zákázníků. I přes experimentální implemetaci plugin na struktuře záznamů této MIB tabulky pracuje správně ve všech módech, mimo módu kontrolující zdroje napájení. Tato chyba však může být způsobena neaktualizovaným firmwarem zařízení, který v současné době není možné aktualizovat.

Nelze tedy tvrdit, že plugin na zařízeních používajících MIB tabulku FJDARY-E150.mib pracuje správně, i když tomu výsledky testu nasvědčují. Tato část funkcionality bude tedy ponechána v experimentální implementaci s možností jejího použití.

9.5 Testovací případ č. 5 – Testování komponent pole FUJITSU ETERNUS DX80 – stavy WARNING a CRITICAL

Pro účely otestování správného předávání stavů a chybových hlášení komponent ve stavech WARNING a CRITICAL, které se nepodařilo reálně navodit na fyzickém zařízení, byly přímo v kódu vytvořeny umělé chybové stavy, které simulují detekci takových stavů pluginem, jejich následnou interpretaci a předání do monitorovacího systému. Výsledek ve webovém rozhraní monitorovacího systému je na obrázku č. 28.

☐	Disk_array_-_DX80	physical_components_channelAdapters		OK	18h 30m 3s	12/04/2016 09:27:06	1/5 (H)	ETERNUSDXL - ET08F12AG with serial number: YL2A002789, channel adapters T/O/W/C - 1/1/0/0
☐		physical_components_CMandComponents		WARNING	12h 29m 40s	12/04/2016 09:27:43	5/5 (H)	ETERNUSDXL - ET08F12AG with serial number: YL2A002789, CM & components T/O/W/C - 5/3/2/0, States: Controller Module, SN:546A8: warning / CM flash-rom, ID:14AXX2: warning /
☐		physical_components_disks		WARNING	12h 29m 40s	12/04/2016 09:26:42	5/5 (H)	ETERNUSDXL - ET08F12AG with serial number: YL2A002789, disks T/O/W/C - 12/11/1/0, States: Disk, ID: 13568: unformatted /
☐		physical_components_enclosures		OK	12h 29m 40s	12/04/2016 09:27:26	1/5 (H)	ETERNUSDXL - ET08F12AG with serial number: YL2A002789, enclosures T/O/W/C - 2/2/0/0
☐		physical_components_expanders		OK	12h 29m 40s	12/04/2016 09:27:06	1/5 (H)	ETERNUSDXL - ET08F12AG with serial number: YL2A002789, expanders T/O/W/C - 1/1/0/0
☐		physical_components_lanPorts		CRITICAL	18h 17m 5s	12/04/2016 09:27:02	5/5 (H)	ETERNUSDXL - ET08F12AG with serial number: YL2A002789, LAN ports T/O/W/C - 3/2/0/1, States: LAN_port - MNT, ID: 64221: linkdown /
☐		physical_components_overall		CRITICAL	18h 17m 20s	12/04/2016 09:26:42	5/5 (H)	Overall status of ETERNUSDXL - ET08F12AG with serial number: YL2A002789 is OK, States: CRITICAL: LAN_port - MNT, ID: 64221: linkdown / WARNING: Disk, ID: 13568: unformatted / CM flash-rom, ID:14AXX2: warning / Inlet Thermal Sensor, ID:55412: warning / Controller Module, SN:546A8: warning /
☐		physical_components_psu		OK	18h 30m 8s	12/04/2016 09:27:02	1/5 (H)	ETERNUSDXL - ET08F12AG with serial number: YL2A002789, power supply units T/O/W/C - 2/2/0/0
☐		physical_components_scu		OK	12h 29m 40s	12/04/2016 09:27:43	1/5 (H)	ETERNUSDXL - ET08F12AG with serial number: YL2A002789, system capacitor units T/O/W/C - 1/1/0/0
☐		physical_components_sensors		WARNING	12h 29m 40s	12/04/2016 09:27:26	5/5 (H)	ETERNUSDXL - ET08F12AG with serial number: YL2A002789, thermal sensors T/O/W/C - 3/2/1/0, States: Inlet Thermal Sensor, ID:55412: warning /

Obrázek 28: Uměle vykazované stavy komponent pole FUJITSU ETERNUS DX80 v prostředí monitorovacího systému

Jak je vidět, některé komponenty vykazují příslušné chybové stavy WARNING a CRITICAL, doplněné o chybová hlášení. V módu *overall* jsou pak vypsána všechna chybová hlášení, které by plugin na poli detekoval. Overall status OK v chybovém hlášení tohoto módu není chybou, jelikož celkový stav zařízení je přebírán přímo z MIB záznamu zařízení a neovlivňují jej tak uměle vytvořené chybové stavy komponent.

Na základě výsledků testování můžeme říci, že naprogramovaný plugin je plně funkční na polích používajících MIB tabulky FJDARY-E60.mib a FJDARY-E101.mib. Vzhledem k omezené možnosti testování ponechávám část pluginu monitorující disková pole používající MIB tabulku FJDARY-E150.mib v experimentální implementaci, i přesto, že test na diskovém poli používající zmíněnou MIB tabulku proběhl dobře s výjimkou nefunkčnosti jednoho módu, která však pravděpodobně není způsobena chybou v pluginu.

10 Ekonomické zhodnocení

Následující kapitola obsahuje ekonomické zhodnocení vypracovaného řešení. Toto zhodnocení budeme uvažovat ve dvou případech, a to nasazení řešení v prostředí firmy CDC Data, s. r. o. a dále zhodnocení nasazení řešení ve fiktivní firmě, která chce zavést monitoring síťových zařízení. V obou případech budeme uvažovat jednoho pracovníka s průměrnou hodinovou mzdou na pozici systémového administrátora nebo správce počítačové sítě v Jihomoravském kraji za rok 2015, kdy tato průměrná mzda činila 270 Kč (Ministerstvo práce a sociálních věcí, 2016).

10.1 Počáteční investice

Budeme-li uvažovat případ nasazení řešení ve firmě CDC Data, s. r. o. budou tyto počáteční náklady nulové. Jelikož se firma monitoringem aktivně zabývá, má vytvořené potřebné hardwarové a softwatové zázemí pro provoz takové služby. Předpokládáme tedy, že firma disponuje serverem, na kterém je nainstalován monitorovací systém Centreon.

V případě firmy, která chce zavést monitoring síťových zařízení, už nějaké počáteční náklady budeme muset brát v potaz. Budeme-li předpokládat, že taková firma disponuje serverem, na kterém lze zprovoznit monitorovací systém Centreon, budou náklady na hardwarové zázemí také nulové. V další fázi musíme na tento server nasadit monitorovací systém Centreon. Tento systém, založený na bázi open source řešení, je v základní plně funkční verzi dostupný zdarma, což je pro firmu velkou výhodou. První investice přichází s požadavkem na nasazení monitorovacího systému. Za předpokladu, že kompletní nasazení systému se časově pohybuje mezi 5–10 hodinami, v závislosti na zdatnosti pracovníka, se pak při hodinové sazbě 270 Kč počáteční investice, spočívající v nasazení monitorovacího systému, pohybuje v rozmezí 1 350–2 700 Kč.

10.2 Náklady na implementaci pluginu

V této části budeme pro oba případy uvažovat stejnou situaci. Pro vytvoření monitorovací služby splňující zadané požadavky na monitoring diskových polí, je zapotřebí vytvořit plugin, který bude požadované informace o zařízení získávat. Shrneme-li celý proces, potřebný pro vytvoření požadované monitorovací služby, obsahující mimo samotné implementace pluginu i další úkony, jako jsou například analýza MIB tabulek zařízení nebo následná konfigurace monitorovacího systému a nasazení pluginu do tohoto systému, pak lze odhadnout časovou náročnost na 30–50 hodin práce, v závislosti na zdatnosti pracovníka. Náklady na implementaci pluginu se tedy při hodinové sazbě 270 Kč pohybují v rozmezí 8 100–13 500 Kč.

10.3 Provoz a údržba

Také v této části budeme uvažovat náklady na provoz a údržbu pro oba případy. Náklady na provoz monitorovacího systému se odvíjí od nákladů na provoz serveru, na kterém monitorovací systém běží. Budeme-li uvažovat server se zdrojem o příkonu 700 W, který firma efektivně využívá a slouží tak i ke dvěma jiným účelům, než pouze jako monitorovací server a rozložíme jeho odběr elektrické energie na tři díly, pak můžeme uvažovat o odběru monitorovacího serveru okolo 200 W s tím, že ostatní služby budou server vytěžovat o něco více. Při průměrné ceně elektrické energie, která činí 5 Kč/kWh a uvažovaném nepřetržitém provozu 24 hodin denně po celý měsíc se poměrná částka monitorovacího systému na využití serveru vyšplhá na 720 Kč měsíčně.

Budeme-li uvažovat obsluhu monitorovacího systému a řešení případných nastalých problémů zjištěných monitorovacím systémem jako součást práce síťového administrátora, na kterou bude mít vyhrazenou hodinu ze standardní pracovní doby a dvě hodiny pohotovostního stavu během víkendů, pak se náklady na údržbu vyšplhají na 10 260 Kč.

10.4 Úspory

Z pohledu firmy CDC Data, s. r. o. se z velké části nejedná o úspory, ale spíše o výnosy. Jelikož tato firma poskytuje monitoring jako externí službu pro firmy bez vlastního monitoringu, účtuje si za tyto služby finanční odměnu. Pro firmu však nasazení monitorovacího systému představuje i jistou část úspor, díky sledování vlastních zařízení společnosti, na základě kterých dokáže včas reagovat na vzniklé problémy nebo dalším problémům předcházet.

V případě firmy, která monitoring nasadila na vlastní náklady, úspory vycházejí z možnosti sledování stavů vlastních síťových zařízení, na základě kterých dokáže ve velmi krátkém čase operativně reagovat na vznikající problémy a předcházet problémům výraznějším, vycházejících z chybné funkcionality některého ze síťových zařízení.

V obou případech je nemožné vyjádřit úsporu v peněžních jednotkách. Výše takových úspor se zpravidla mění na základě včasně vyřešených problémů zařízení a aktuální situaci.

11 Závěr

Na základě analyzovaných uživatelských požadavků firmy CDC Data, s. r. o. byl v rámci bakalářské práce vytvořen plugin určený pro monitoring diskových polí integrovatelný do monitorovacího systému Centreon. Požadovanou funkcionalitu se povedlo implementovat a otestovat na diskových polích používajících MIB tabulky FJDARY-E60.mib nebo FJDARY-E101.mib. Na těchto polích vykazuje požadovanou funkcionalitu. Implementace funkční části pluginu, jejímž předmětem jsou disková pole používající MIB tabulku FJDARY-E150.mib, byla vytvořena pouze experimentálně, jelikož po dobu vývoje pluginu nebylo takové zařízení dostupné. I přes experimetální implemetaci se tuto funkční část později povedlo alespoň jednorázově otestovat na produkčním zařízení zákazníka firmy CDC Data, s. r. o., kde vykazovala správnou funkčnost, vyjma nesprávné funkcionality módu kontrolujícího zdroje napájení. Tato chyba však byla zřejmě způsobena zastaralým firmwarem na monitorováném zařízení. Z důvodu nemožnosti řádného otestování této funkční části pluginu je tato část označena jako experimetální implemetace s ponechanou možností použití.

Všechny požadavky na bakalářskou práci se povedlo zpracovat, mimo spolehlivého otestování implementace části pluginu určené pro monitoring diskových polí založených na MIB tabulce FJDARY-E150.mib. Zpracování této práce bylo pro mne velkým přínosem, zejména díky zadání a zpracování úkolu v reálném prostředí a prohloubení znalostí v profesním směru, kterým bych se chtěl dále ubírat.

12 Seznam zdrojů

Benda, L. *Integrace monitorovacího systému počítačové sítě na bázi open source a komerčních produktů*. Brno, 2011. Bakalářská práce. Mendelova univerzita v Brně. Fakulta provozně ekonomická. Vedoucí práce Ing. Martin Pokorný, Ph.D.

Blokdijk, G. *Management Information Base - Simple Steps to Win, Insights and Opportunities for Maxing Out Success*. [Aspley]: Complete Publishing, 2015. 154 s. ISBN 978-1488895784.

CDC Data s. r. o. *O nás* [online]. 2015. [cit. 2015-12-01]. Dostupné z: http://www.cdc.cz/o-spolecnosti.

CDC Data s. r. o. *Produkty* [online]. 2015. [cit. 2015-12-01]. Dostupné z: http://www.cdc.cz/produkty.

Centreon. *Company*. 2015a [online]. [cit. 2015-12-01]. Dostupné z: https://www.centreon.com/en/company/.

Centreon. *Integration Overview* [online]. 2015b. [cit. 2016-04-14]. Dostupné z: https://documentation.centreon.com/docs/centreon-engine/en/latest/user/configuration/basics/integration_overview.html.

Centreon. *Partners* [online]. 2015c. [cit. 2015-12-01]. Dostupné z: https://www.centreon.com/en/partners/find-a-partner/.

Centreon. *Products* [online]. 2015d. [cit. 2015-12-01]. Dostupné z: https://www.centreon.com/en/products/.

Centreon. *Release notes* [online]. 2016. [cit. 2016-05-11]. Dostupné z: https://documentation.centreon.com/docs/centreon/en/2.7.x/release_notes/centreon-2.7/index.html.

Centreon. *Welcome to Centreon's documentation!* [online]. 2015e. [cit. 2015-12-17]. Dostupné z: https://documentation.centreon.com/docs/centreon/en/2.7.x/index.html.

Dařena, F. *Myslíme v jazyku PERL*. Praha: Grada, 2005. 700 s. ISBN 978-80-247-6390-3.

Dařena, F. *Studijní materiály k předmětu Programovací jazyk Perl* [online]. 2012. [cit. 2015-12-21]. Dostupné z: https://akela.mendelu.cz/~darena/Perl/.

Ellingwood, J. *An Introduction to SNMP* [online]. 2014. [cit. 2015-12-17]. Dostupné z: https://www.digitalocean.com/community/tutorials/an-introduction-to-snmp-simple-network-management-protocol.

Fall, K. a Stevens, R. *TCP/IP Illustrated, Volume 1: The Protocols*. 2nd edition. [Boston]: Addison-Wesley, 2011. 1020 s. ISBN 978-0-321-33631-6.

Fujitsu Ltd. *FJDARY-E60.mib*. 2009.

Fujitsu Ltd. *FJDARY-E101.mib*. 2011.

Fujitsu Ltd. *FJDARY-E150.mib*. 2013.

Habrman, Z. *Monitorování aktivních prvků počítačové sítě*. Zlín, 2013. Diplomová práce. Univerzita Tomáše Bati ve Zlíně. Fakulta aplikované informatiky. Vedoucí práce Ing. Jiří Korbel, Ph.D.

Hartpence, B. *Packet Guide to Core Network Protocols*. 1st edition. [Sebastopol]: O'Reilly Media, 2011. 264 s. ISBN 978-1-4493-0653-3.

Hendel, V. *Computer Network Security: Virtual Private Networks*. [North Charleston]: CreateSpace Independent Publishing, 2015. 120 s. ISBN 978-1519222671.

Chamith, B. *SSH Tunneling Explained* [online]. 2012. [cit. 2016-30-03]. Dostupné z: https://chamibuddhika.wordpress.com/2012/03/21/ssh-tunnelling-explained/.

Ipinsivy. *How to set up your monitoring platform – configuration* [online]. 2013. [cit. 2015-12-17]. Dostupné z: http://blog.centreon.com/how-to-set-up-your-monitoring-platform-configuration/.

Ipinsivy. *How to set up your monitoring platform – part 1* [online]. 2013. [cit. 2015-12-17]. Dostupné z: http://blog.centreon.com/how-to-monitor-your-monitoring-platform-part-1/.

Kocourková, L. *Udržitelnost a rozvoj monitorovacích systémů v komerční společnosti*. Brno, 2013. Diplomová práce. Masarykova univerzita v Brně. Fakulta informatiky. Vedoucí práce doc. RNDr. Jaroslav Ráček, PhD.

Malík, J. *Správa různorodých síťových zařízení pomocí SNMP protokolu*. Brno, 2011. Diplomová práce. Masarykova univerzita v Brně. Fakulta informatiky. Vedoucí práce RNDr. Tomáš Rebok, Ph.D.

Mauro, D. a Schmidt, K. *Essential SNMP*. 2nd edition. [Sebastopol]: O'Reilly Media, 2005. 462 s. ISBN 978-0-596-00840-6.

Ministerstvo práce a sociálních věcí. *Regionální statistika ceny práce*. 2016 [online]. [cit. 2016-08-05]. Dostupné z: http://www.ispv.cz/getattachment/66a1676b-c19b-449a-8ae9-3764187c3d9f/Publikace-ve-formatu-PDF.aspx?disposition=attachment.

Schwartz, R. a Foy, B. a Phoenix, T. *Learning Perl*. 6th edition. [Sebastopol]: O'Reilly Media, 2011. 388 s. ISBN 978-1-4493-0358-7.

Szabo, G. *Monitoring lokálnej siete prostredníctvom Nagiosu*. Brno, 2012. Bakalářská práce. Masarykova univerzita v Brně. Fakulta informatiky. Vedoucí práce doc. RNDr. Eva Hladká, Ph.D.

Town, D. *Net::SNMP* [online]. 2010. [cit. 2016-03-02]. Dostupné z: http://search.cpan.org/~dtown/Net-SNMP-v6.0.1/lib/Net/SNMP.pm.

VOSECKÝ, J. *Implementace monitorovacího systému v síti poskytovatele internetových služeb*. Praha, 2012. Diplomová práce. Bankovní institut vysoká škola Praha. Katedra matematiky, statistiky a informačních technologií. Vedoucí práce Ing. Vladimír Beneš.

W3TECHS. *Usage of operating systems for websites* [online]. 2016. [cit. 2016-03-02]. Dostupné z: http://w3techs.com/technologies/overview/operating_system/all.

Přílohy

A CD – Zdrojový kód pluginu