Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Monitorování systémů a aplikací v podnikovém prostředí s využitím IBM Tivoli Monitoring

Autor: Bc. Jiří Kotalík

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Jiří Kotalík

Informatika

Název práce

Monitorování systémů a aplikací v podnikovém prostředí s využitím IBM Tivoli Monitoring

Název anglicky

Systems and applications monitoring in the enterprise environment using IBM Tivoli Monitoring

---

Cíle práce

Hlavním cílem práce je přiblížení problematiky monitorování systémů a aplikací v podnikovém prostředí. Dílčími cíli jsou:
- charakteristika protokolu SNMP a MIB souborů
- charakteristika monitoringu s agenty a bez agentů
- představení standardů z oblasti logování
- představení IBM Tivoli Monitoring
- představení vhodných nástrojů a technologií pro vytváření agentů
- vytvoření ukázkového monitorovacího agenta v IBM Tivoli Monitoring Agent Builder a instalace
- zhodnocení a závěry

Metodika

Metodika řešené problematiky je založena na studiu a analýze odborných informačních zdrojů a praktických zkušeností s monitorováním v konkrétním prostředí. Teoretická část představí prostředí IBM Tivoli Monitoring, technologie a standardy využívané v oblasti monitorování a při vytváření agentů. Praktická část ukáže, na základě syntézy teoretických a praktických zkušeností, tvorbu agenta, který bude monitorovat zvolené charakteristiky v prostředí IBM Tivoli Monitoring. Poslední kapitola bude věnována shrnutím a závěrům.

Doporučený rozsah práce

50 – 60 stran

Klíčová slova

Agent, SNMP, MIB, skript, situace, platforma, server, monitorovací systém, log

---

Doporučené zdroje informací

František Dařena, Myslíme v jazyku Perl, Grada, 2005, ISBN: 978-80-247-6390-3

Herbert Schildt, Mistrostvi – Java , Computer press, 2014, ISBN: 978-80-251-4145-8

IBM. IBM Tivoli Agent Builder Version 6.3.1 User's Guide. IBM Knowledge Centre. [Online] [Citace: 16. 4 2016.] http://www.ibm.com/support/knowledgecenter/SSTFXA_6.3.0/com.ibm.itm.doc_6.3/agentbuilder63_user.pdf

IBM. IBM Tivoli Monitoring Version 6.3 Administrator's Guide. IBM Knowledge Centre. [Online] [Citace: 16. 4 2016.] http://www.ibm.com/support/knowledgecenter/SSTFXA_6.3.0/com.ibm.itm.doc_6.3/itm63_admin.pdf

IBM. Úvodní stránka produktu Tivoli Monitoring. IBM Knowledge Center. [Online] [Citace: 16. 4 2016.] http://www.ibm.com/support/knowledgecenter/SSTFXA/welcome

Mark G. Sobell, Mistrovství v Linuxu – Příkazový řádek, shell, programování, Computer press, 2007, ISBN 978-80-251-1726-2

Stephen J. Bigelow, Mistrovství v počítačových sítích, Computer press, 2004, ISBN 80-251-0178-9

Předběžný termín obhajoby

2016/17 LS – PEF

Vedoucí práce

Ing. Jiří Vaněk, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 23. 5. 2016

Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 2. 8. 2016

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 19. 03. 2017

Čestné prohlášení

Prohlašuji, že svou diplomovou práci Monitorování systémů a aplikací v podnikovém prostředí s využitím IBM Tivoli Monitoring jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne _____

Poděkování

Chtěl bych touto cestou poděkovat vedoucímu své diplomové práce Ing. Jiřímu Vaňkovi, Ph.D. za podnětné rady a diskuse. Dále bych rád poděkoval prof. RNDr. Jiřímu Horáčkovi, DrSc. a jeho ženě MUDr. Evě Horáčkové za dlouhodobou podporu v mých snahách při studiích. Poděkování patří i Ing. Leoši Stránskému a Bc. Petru Šintákovi za cenné rady a možnost práce s produktem IBM Tivoli Monitoring. Díky patří i celé mé rodině.

Monitorování systémů a aplikací v podnikovém prostředí s využitím IBM Tivoli Monitoring

Souhrn

Diplomová práce je zaměřena na problematiku monitorování systémů a aplikací v podnikovém prostředí. Teoretická východiska charakterizují technologie, které jsou určeny k monitorování a vývoji agentů na zakázku. Mezi charakterizovanými technologiemi jsou protokol SNMP a MIB soubory, logovací systémy, prostředí IBM Tivoli Monitoring, vývojové prostředí IBM Agent Builder, regulární výrazy a doporučené jazyky pro tvorbu agentů na míru. V práci je shrnuta problematika monitorování s agenty a bez agentů. Praktická část představí ukázku modelového zadání, následnou analýzu a tvorbu monitorovacího agenta. Agent je integrován do prostředí IBM Tivoli Monitoring a otestována požadovaná funkcionalita. Práce obsahuje praktická doporučení pro logování, vývoj, integraci a testování agentů.

Klíčová slova: SNMP, MIB, agent, server, monitorovací systém, situace, událost, log

Systems and applications monitoring in the enterprise environment using IBM Tivoli Monitoring

Summary

The diploma thesis deals with the monitoring of systems and applications in the enterprise environment. Its theoretical background characterizes technologies used for the monitoring and for a custom agent development. Introduced are the SNMP and MIB files, logging systems, the IBM Tivoli Monitoring platform, the IDE IBM Agent Builder, regular expressions, and the recommended programming languages. The thesis discusses both the agent and agentless approaches. The practical part shows a model example of the custom agent. After the task analysis, the monitoring agent is created, integrated into the IBM Tivoli Monitoring and tested. Recommendations for the logging, development, integration, and testing the agents are summarized.

Keywords: SNMP, MIB, agent, server, monitoring system, situation, event, log

Obsah

1 Úvod

Monitorování systémů a aplikací je podstatnou částí úspěšného podnikání. Dříve nebo později si každý podnik uvědomí, že sledování prostředků a dostupnosti systémů je nezbytností. Výpadky systémů mohou mít za následek finanční ztráty a nespokojené zákazníky.

Sledovat jednotlivé systémy a reagovat na případné události je obtížným úkolem. Použitím vhodných nástrojů může být monitorování usnadněno. Monitorovací systémy nabízejí centralizovaný způsob odhalování problematických stavů. Vytvořením podmínek a nastavením automatických oznámení lze snadno identifikovat a řešit vzniklé události.

Sledování systémů probíhá pomocí specializovaných protokolů, jako je například SNMP. Protokol SNMP byl navržen ke sbírání dat a nastavování síťových prostředků. Dnes je již ve své třetí verzi. K dotazování jsou využity soubory Management Information Base, v nichž se nacházejí jednotlivé identifikátory sledovaných prvků. Na trhu je široká paleta síťových zařízení, která SNMP podporuje.

Operační systémy Microsoft Windows využívají ke správě a sběru informací technologii Windows Management Instrumentation, která využívá tzv. Common Information Model.

Obchod skýtá velké množství nástrojů ke sledování podnikového prostředí. Jednotlivé prostředky se liší poskytovanými schopnostmi. Mnoho z nich podporuje vytváření různě složitých agentů na míru potřebám zákazníka. Mezi populární nástroje se řadí monitorovací systémy Nagios a Zabbix.

Společnost IBM nabízí produkt IBM Tivoli Monitoring, který je určen pro střední i velké podniky a podporuje široké množství operačních systémů. IBM dodává uživatelům vývojářské nástroje k vytváření agentů na míru. Vývoj agentů probíhá v integrovaném vývojovém prostředí IBM Agent Builder. Agenti mohou být vytvořeni za pomoci různých programovacích jazyků. Mezi nejpoužívanějšími jazyky v oblasti monitoringu jsou Perl, Python a Java. Samozřejmě lze použít i další jazyky.

2 Cíl práce a metodika

2.1 Cíl práce

Hlavním cílem práce je přiblížení problematiky monitorování systémů a aplikací v podnikovém prostředí. Dílčími cíli jsou:

- charakteristika protokolu SNMP a MIB souborů
- charakteristika monitoringu s agenty a bez agentů
- představení standardů z oblasti logování
- představení IBM Tivoli Monitoring
- představení vhodných nástrojů a technologií pro vytváření agentů
- vytvoření ukázkového monitorovacího agenta v prostředí IBM Tivoli Monitoring Agent Builder a instalace
- zhodnocení a závěry

2.2 Metodika práce

Metodika řešené problematiky je založena na studiu a analýze odborných informačních zdrojů a praktických zkušeností s monitorováním v konkrétním prostředí. Teoretická část představí prostředí IBM Tivoli Monitoring, technologie a standardy využívané v oblasti monitorování a při vytváření agentů.

Praktická část ukáže, na základě syntézy teoretických a praktických zkušeností, tvorbu agenta, který bude monitorovat zvolené charakteristiky v prostředí IBM Tivoli Monitoring.

Nejprve bude vytvořeno modelové zadání, ve kterém budou charakterizovány požadavky na monitorování. Následná analýza zadání navrhne postup pro řešení daného problému.

Pro vývoj, testování a integraci agenta bude vytvořeno virtualizované prostředí na platformě Microsoft Hyper-V. Databázový server poběží v kontejneru s využitím prostředí Docker. Agent bude naprogramován ve vhodně zvoleném jazyce a následně zabalen pomocí vývojového prostředí IBM Agent Builder. Tvorbu agenta vhodně doplní ilustrace a praktická doporučení vedoucí k přehlednosti řešené problematiky. Následně bude provedena integrace agenta v prostředí IBM Tivoli Monitoring a vytvořeny požadované

situace. Pro testování funkčnosti agenta budou vytvořeny vzorky dat, které generují požadované události.

Poslední kapitola bude věnována shrnutím a závěrům.

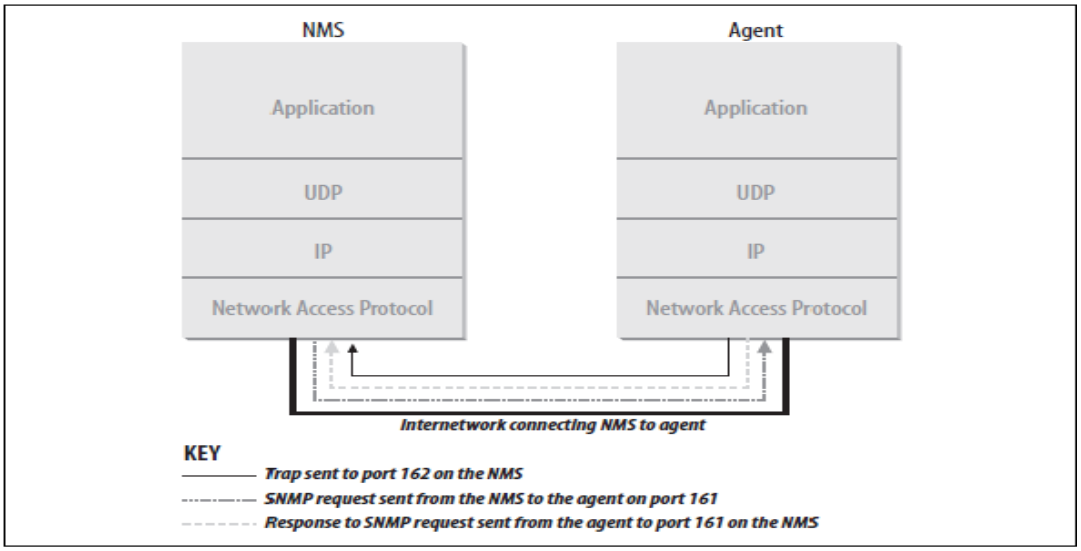
3 Teoretická východiska

3.1 Úvod do SNMP

V dnešní době rozsáhlých sítí směrovačů, přepínačů a serverů může být správa všech zařízení v podnikové síti výzvou. Zjišťování informací, zda prvky sítě běží v pořádku a optimálně, lze získat pomocí SNMP (Simple Network Management Protocol). SNMP byl představen v roce 1998, aby uspokojil potřeby pro správu IP (Internet Protocol) zařízení. Předchůdce protokolu SNMP byl protokol SGMP (Simple Gateway Management Protocol), který byl vyvinut za účelem správy směrovačů. Dnes je SNMP již ve třetí verzi a lze ho využít jak na správu zařízení, tak i software. Potřebné informace lze získávat například z databázových systémů, webových serverů a dalších. Vzdálená správa se odehrává pomocí jednoduchých zpráv.

SNMP využívá ke své práci UDP (User Datagram Protocol), kterým předává data mezi správci a agenty. Jelikož se jedná o nespojovaný protokol na rozdíl od TCP (Transmission Control Protocol), je nutné ověřovat, že zprávy došly ke svému cíli. Ověření probíhá pomocí definovaného časového intervalu, ve kterém se zařízení musí ozvat. Pokud se zařízení neozve, posílá správce nový požadavek. Počet opakování je nastavitelný. Problém vzniká, když zařízení zašle zprávu pomocí TRAP oznámení. Pokud zpráva nedorazí ke správci, agent se to nedozví. Výhodou UDP je malý dopad na výkon sítě.

Používané UDP porty ke komunikaci pomocí protokolu SNMP jsou 161 pro příjem a odesílání požadavků a port 162 k přijímání TRAP oznámení ze vzdáleného zařízení [1] [2].



Obrázek 1: Ukázka komunikace mezi správcem a agentem [1]

3.1.1 IETF a RFC

Uskupení IETF (Internet Engineering Task Force) je zodpovědné za definování standardů v oblasti internetových protokolů. IETF publikuje tzv. RFC (Requests For Comments), které jsou specifikací pro mnoho protokolů v říši IP.

Dokumenty vstupují v prvním kole jako návrh standardu. V dalším stádiu jsou považovány za pracovní koncepty. Koncepty jsou upravovány a mohou být ve finále schváleny jako standardy. Oficiální webová stránka s RFC se nachází na adrese http://www.ietf.org/rfc.html.

SNMPv1 je počáteční verze SNMP. Protokol je definovaný v RFC 1157 a je historickým standardem IETF. Bezpečnost je založena na tzv. communities.

SNMPv2 je rozšíření SNMPv1 o nové příkazy. Definice protokolu pochází z následujících RFC 3416, RFC 3417 a RFC 3418.

SNMPv3 je poslední standardizovanou verzí SNMP. Ke standardizaci došlo v roce 2002. Největším přínosem třetí verze je důraz na bezpečnost a soukromí mezi komunikujícími entitami. Třetí verze je definována v RFC 3410, RFC 3411, RFC 3412, RFC 3413, RFC 3414, RFC 3415, RFC 3416, RFC 3417, RFC 3418 a RFC 2576 [1].

3.1.2 SNMP Communities

SNMPv1 a SNMPv2 používají pojem communities k navázání vzájemné důvěry mezi správci a agenty. Communities jsou obyčejné textové řetězce, které hrají roli hesel. Agent je nakonfigurovaný se třemi textovými řetězci pro operace: read-only, read-write a trap.

Read-only řetězec umožňuje pouze číst informace ze zařízení, ale nelze měnit nastavení. Read-write řetězec zprostředkovává navíc zápis informací do zařízení. Posledním textovým řetězcem je trap, který nastavuje asynchronní notifikace z agenta.

Většina výrobců prodává zařízení s nastavenými výchozími hesly. Ve výchozím stavu je heslo pro read-only operace nastaveno na řetězec public. Pro operace read-write je heslo nastaveno na řetězec private. Před nasazením zařízení do provozu je vhodné nastavit nová hesla. Pro zprávy typu trap je potřeba nastavit místo, kam budou doručovány zprávy z agenta. Hesla jsou po síti posílána jako otevřený text. Hesla lze snadno odposlechnout. Nastavením trap signálu pro nezdařené SNMP ověření lze odhalit útočníka. SNMPv3 řeší

problém bezpečnosti a úniku informací. Umožňuje bezpečné ověřování a komunikaci mezi správci a agenty. Proti útokům lze použít nastavení brány firewall, která povolí komunikaci pouze mezi určenými body a na konkrétním portu [1] [2].

3.1.3 SNMP operace

PDU (Protocol Data Unit) je formát zprávy, který vyžívají agenti a správci ke komunikaci mezi sebou. Každá níže uvedená SNMP operace má standardní PDU formát.

- get
- getnext
- getbulk (SNMPv2 a SNMPv3)
- set
- getresponse
- trap
- notification (SNMPv2 a SNMPv3)
- inform (SNMPv2 a SNMPv3)
- reports (SNMPv2 a SNMPv3)

Operace get je žádost o získání konkrétní hodnoty OID. Správce odešle žádost a čeká na odpověď od agenta. Agent odesílá svou odpověď pomocí getresponse operace.

Operace getnext umožňuje získání skupiny hodnot z MIB. Pro každý MIB objekt je vytvořena samostatná žádost getnext a odpověď getresponse. Getnext prochází vybraný podstrom lexikograficky do hloubky. Procházení končí, když agent ohlásí chybu. Chyba je signálem nepřítomnosti dalších objektů v podstromu.

Operace getbulk je definována od SNMPv2. Umožňuje získat velké části tabulky najednou. Standardně lze získat více objektů použitím operace get, avšak může dojít k omezení na straně agenta. Pokud agent neumí vrátit požadovaný počet objektů najednou, vrací chybovou zprávu. Na druhou stranu operace getbulk požádá agenta o tolik dat, kolik zvládne odeslat. Z toho vyplývá, že agent může odeslat nekompletní data. Getbulk pracuje s pojmy nonrepeaters a max-repetitions. Nonrepeaters udává, že prvních N objektů může být získáno getnext operací. Max-repetitions říká, že se agent má pokusit u zbylých objektů o získání až M hodnot pomocí operace getnext.

Operace set je využívána ke změně hodnoty spravovaného objektu nebo k vytvoření nové řádky v tabulce. Objekt musí mít nastavena práva read-write, nebo write-only. Správce odešle příkaz ke změnění hodnoty. Agent zkontroluje, zda má objekt právo k zápisu. Pokud ano, dojde ke změně a agent odesílá potvrzení. Pokud nastavení selže, agent odesílá informaci s chybovou zprávou:

- noError(0) – Nenastala žádná chyba při provádění požadavku.
- tooBig(1) – Odpověď na požadavek byla příliš velká, aby se vešla do jedné odpovědi.
- noSuchName(2) – Požadované OID nebylo nalezeno.
- badValue(3) – Nastavená hodnota objektu není typově konzistentní.
- readOnly(4) – Není používáno. Ekvivalentní s noSuchName.
- genError(5) – Pokud chyba neodpovídá výše zmíněným.
- noAccess(6) – Nepovolený pokus o zápis do proměnné. Pokud má proměnná nastaven přístup na not accessible.
- wrongType(7) – Objekt byl nastaven na hodnotu špatného typu, dle definice.
- wrongLength(8) – Hodnota objektu přesáhla maximální rozsah.
- wrongEncoding(9) – Pokus o nastavení objektu hodnotou ve špatném kódování.
- wrongValue(10) – Hodnota objektu byla nastavena na neznámou hodnotu. Nastává při nastaveném výčtu možností pro objekt.
- noCreation(11) – Pokus o nastavení nebo vytvoření neexistující proměnné v MIB.
- inconsistentValue(12) – Proměnná v MIB je v nekonzistentním stavu a nelze ji nastavit.
- resourceUnavailable(13) – Nedostupnost systémových prostředků k vykonání operace set.
- commitFailed(14) – Došlo k jakékoliv chybě při nastavení.
- undoFailed(15) – Nastavení objektu selhalo. Agent nemohl provést obnovení původních hodnot.
- authorizationError(16) – SNMP operace nebyla autorizována. Špatný community string.
- notWritable(17) - Proměnná neakceptuje operaci set, přestože má.
- incosistentName(18) – Pokus o nastavení proměnné selhal, protože proměnná byla v nekonzistentním stavu [1].

version	community	PDU type	request ID	error status	error index	OID	value

Obrázek 2: Ukázka formátu PDU protokolu SNMP [1]

Trap je zpráva, kterou odesílá agent správci, pokud došlo k nějaké události. Agent má nastavenu IP adresu správce tak, aby věděl, kam má poslat trap oznámení. Správce neodesílá potvrzení o přijetí zprávy. Agent se tedy nedozví, zda byla zpráva doručena. SNMPv2 umožňuje ověřování doručení trap signálu.

Když správce obdrží trap signál, je potřeba, aby věděl, jak má interpretovat příchozí informace. Trap signál je identifikován sedmi standardními číselnými identifikátory (0-6). Identifikátor s číslem 6 je speciální kategorie určená pro podnikové prostředí, kde různí výrobci a uživatelé specifikují vlastní trap signály. Identifikátor 6 vyjadřuje všechny ostatní nestandardní trap signály.

Podniky a uživatelé definují své specifické trap signály v podstromu iso.org.dod.internet.private.enterprises uvnitř MIB. Standardní názvy trap signálů a číselných identifikátorů jsou následující:

- coldStart(0) – Indikace restartování agenta. Všechny spravované hodnoty jsou resetovány. Počítadla jsou nastavena na 0. ColdStart například indikuje to, že do sítě bylo přidáno nové zařízení.
- warmStart(1) – Indikuje, že agent se reinicializoval. Spravované proměnné nejsou resetovány.
- linkDown(2) – Pokud rozhraní selže. První proměnná je index zařízení, které selhalo, v tabulce zařízení.
- linkUp(3) – Pokud rozhraní nastartuje. První proměnná je index zařízení, které nastartovalo, v tabulce zařízení.
- authenticationFailure(4) – Identifikace pokusu o nepovolený přístup k zařízení. Špatně zadaný community string.
- egpNeighboursLoss(5) – Indikuje, že EGP sousedi selhali.
- enterpriseSpecific(6) – Indikuje, že trap signál není standardní a musí být identifikován ze zaslané zprávy [1].

3.1.4 SNMPv3

Bezpečnost protokolu SNMP ve verzi první a druhé je jeho největší slabinou. Autentizace mezi správcem a agentem v první a druhé verzi SNMP probíhá pomocí hesel, která jsou odesílána v otevřeném textu. Hesla lze odposlechnout po síti a zneužít.

SNMPv3 je navržen tak, aby byl bezpečný. Bezpečnost je jediná změna oproti první a druhé verzi. Podporuje stejné operace a nemění protokol. Mění se pouze názvosloví pro některé prvky. Správci a agenti se nyní nazývají SNMP entitami. Každá entita se skládá ze SNMP enginu a několika SNMP aplikací. Nový koncept je důležitý z hlediska definice architektury. Architektura pomáhá oddělit jednotlivé prvky SNMP systému tak, aby bylo možné implementovat bezpečnostní mechanismy [1].

3.1.4.1 SNMPv3 Engine

Engine je složen ze čtyř částí: Dispatcher, Message Processing Subsystem, Security Subsystem a Access Control Subsystem.

Dispatcher zajišťuje příjem a odesílání zpráv. Zjišťuje verzi přijaté zprávy, a pokud je verze podporována, předává zprávu do Message Processing Subsystem. Dispatcher předává zprávy také jiným subsystémům.

Message Processing Subsystem připravuje zprávy k odeslání a extrahuje data z přijatých zpráv. Message Processing Subsystem může obsahovat několik modulů. Může obsahovat subsystém ke zpracování zpráv SNMPv1, SNMPv2, SNMPv3 a dalších, které jsou ještě ve vývoji.

Security Subsystem zprostředkovává autentizaci a služby soukromí. Autentizace probíhá buď pomocí community řetězců (SNMPv1 a SNMPv2), nebo ověřováním uživatelů (SNMPv3). Pro ověřování uživatelů jsou požity hashovací funkce MD5 nebo SHA. Služby soukromí poskytují šifrování SNMP zpráv pomocí DES, triple DES nebo AES algoritmu.

Access Control Subsystem je zodpovědný za řízení přístupu k MIB objektům. Lze řídit přístup uživatelů ke konkrétním objektům, a jaké operace s nimi mohou provádět [1].

3.1.4.2 SNMP aplikace

Třetí verze SNMP je rozdělena do několika základních aplikací: Command generator, Command responder, Notification originator, Notification receiver a Proxy forwarder. Velkou výhodou oproti verzi první a druhé je, že sadu základních aplikací lze rozšířit o další (RFC 3411).

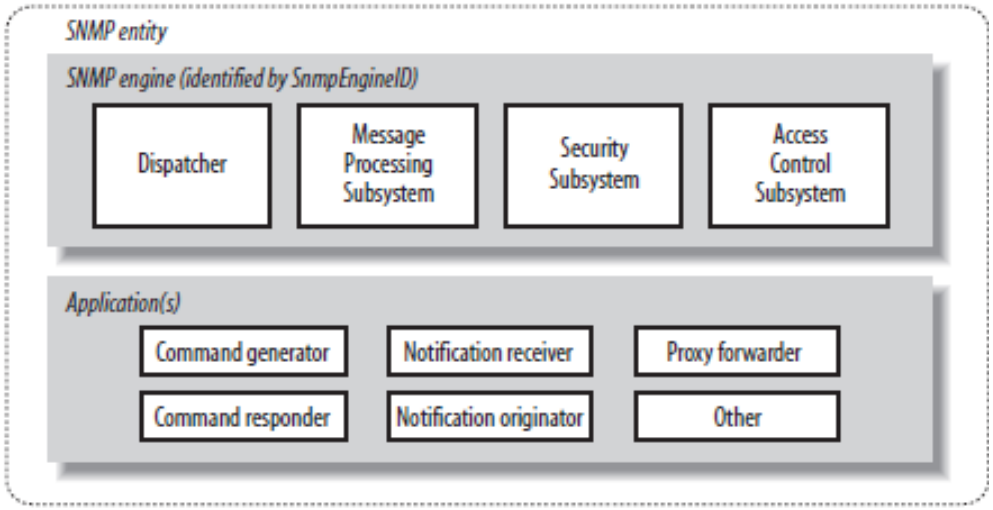
Command generator vytváří požadavky get, getnext, getbulk, set a zpracovává jejich odpovědi. Aplikace je implementována uvnitř správce tak, aby se mohl dotazovat zařízení v síti.

Command responder zajišťuje odpovědi na operace get, getnext, getbulk a set. Aplikace je implementována monitorovanou SNMP entitou.

Notification originator vytváří trap signály a upozornění. Implementace je součástí monitorované SNMP entity.

Notification receiver je implementován správcem. Přijímá trap signály a informační zprávy.

Proxy forwarder usnadňuje posílání zpráv mezi SNMP entitami [1].



Obrázek 3: Zobrazení prvků SNMPv3 entity [1]

3.1.4.3 Bezpečnostní model

K autentizaci SNMPv3 zpráv jsou využívány hashovací funkce MD5 a SHA1. S algoritmem HMAC (keyed Hashing for Message Authentication) tvoří mechanismus k autentizaci a ověření integrity dat.

MD5 vytváří 128 bitový hash a SHA1 160 bitový hash. Obě hashovací funkce mají fixní velikost a nemohou být použity samostatně k autentizaci. Algoritmus HMAC ve spojení právě s MD5 a SHA1 počítá výsledný hash zprávy. Před výpočtem výsledného řetězce je na konec dat přidáno tajné heslo. Tajné heslo musí být známo oběma stranám, které spolu komunikují. Dle RFC musí mít heslo alespoň osm znaků.

Šifrování SNMP dat je zajištěno pomocí CBC-DES algoritmu. Obě komunikující strany musí znát tajné šifrovací heslo. USM tabulka uživatelů ukládá hesla a další podrobnosti v přenášeném paketu v sekci msgPrivacyParameters.

Každá SNMP entita spravuje uživatelskou tabulku, kde jsou uloženy informace o všech uživatelích, kteří mají přístup k systému pomocí SNMP. Tabulka obsahuje následující informace: [1]

- Username - Textový řetězec reprezentující uživatelské jméno, tzv. security name.
- Authentication protocol - Informace o autentizačním protokolu, pokud je použit. Možné hodnoty jsou: usmNoAuthProtocol, usmHMACMD5AuthProtocol, usmHMACSHAAuthProtocol.
- Authentication key – heslo pro autentizaci. Musí mít alespoň osm znaků.
- Privacy protocol - informace o protokolu zajišťující soukromí zpráv. Možné hodnoty jsou usmNoPrivProtocol a usmDESPrivProtocol.
- Privacy key - heslo k zajištění soukromí. Musí mít alespoň osm znaků.
- usmUserSpinLock - zámek pro synchronizaci přístupu k řádkám v tabulce.

VACM (View Access Control Model) je použit pro řízení přístupu k jednotlivým objektům v MIB. Pokud odesílatel nemá právo na konkrétní požadavek, je vrácena chyba. VACM využívá čtyři tabulky, které jsou využity pro různé aspekty přístupu:

- Context Table - Tabulka udržující seznam spravovaných objektů majících přístupová omezení. Tabulka vacmContextTable je indexována podle položky contextName.
- Tabulka vacmSecurityToGroupTable je využívána k ukládání informací o skupinách. Skupina je tvořena kombinacemi žádné, nebo více položek securityModel a položkou securityName. Kombinace udává, ke kterým spravovaným objektům lze přistupovat. Indexace tabulky probíhá pomocí securityModel a securityName.
- Access table je určena k uložení přístupových práv k definovaným skupinám. Indexace probíhá pomocí položek groupName, contextPrefix, securityModel a securityLevel.
- View Tree Family Table uchovává MIB pohledy. Pohled určuje, ke kterým objektům může být přistupováno. Každá řádka tabulky specifikuje podstrom MIB souboru a masku přístupu. [1]

V praxi při využití SNMPv3 jsou důležité následující konfigurační parametry:

- Uživatelské jméno – Textový řetězec osoby zodpovědné za spravovanou SNMP entitu.
- Úroveň zabezpečení – Některé aplikace vyžadují explicitně nastavit bezpečnostní úroveň. Bezpečnostní úroveň je specifikována kombinací použitého protokolu pro autentizaci a soukromí. Možné hodnoty jsou: noAuthNoPriv, authNoPriv, authPriv.
- Autentizační protokol – Použitý protokol pro autentizaci. MD5 nebo SHA1 dle RFC.
- Autentizační heslo – heslo použité ve spojení s autentizačním protokolem. Musí být alespoň osm znaků dlouhé.
- Protokol soukromí – použitý protokol k šifrování dat v SNMP paketech. Dle RFC specifikace je použit DES.
- Heslo soukromí – heslo použité ve spojení s protokolem soukromí. Musí být alespoň osm znaků dlouhé.

3.2 Struktura spravovaných informací v MIB

Informace o datech a jejich struktuře, které poskytuje zařízení pomocí SNMP, jsou uloženy v MIB (Management Information Base) souborech. MIB soubory využívají notaci SMI (Structure Management Information) druhé verze, která je definovaná v RFC 2578.

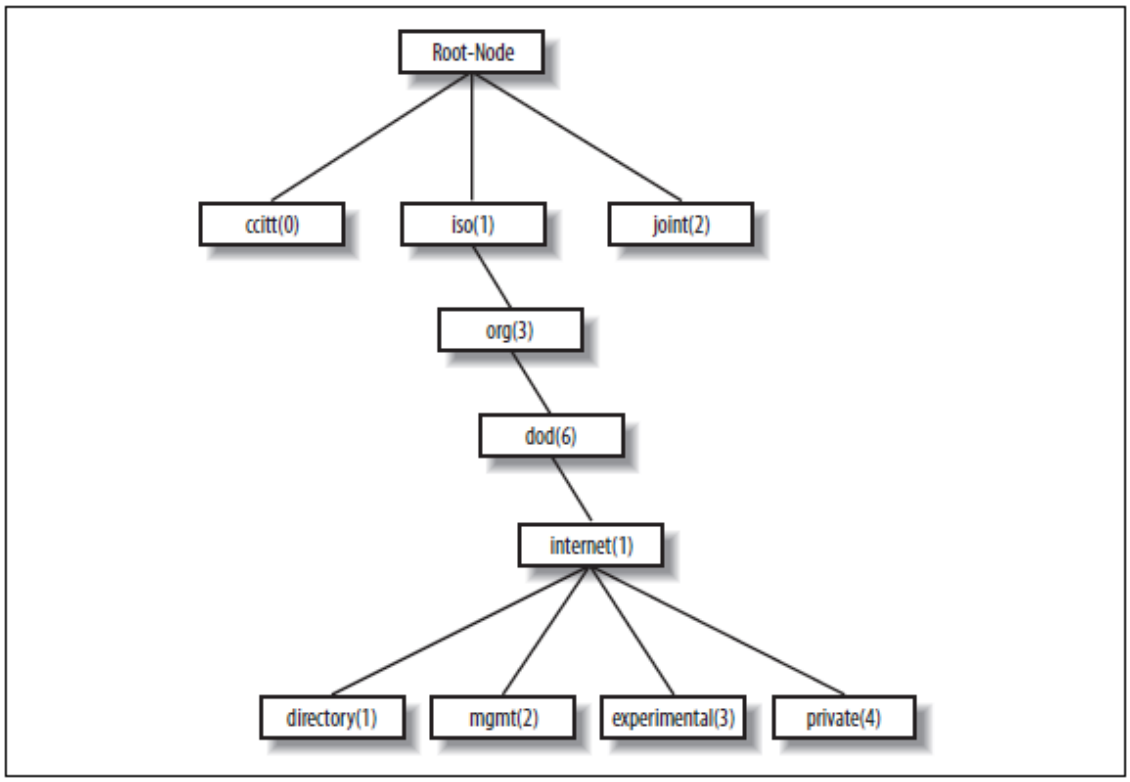
Každý objekt má jedinečný název, který jej definuje. Jedinečný název je označován jako tzv. OID (Object Identifier). Identifikátory objektů se vyskytují v číselné a pro lidi snadno pochopitelné textové podobě. V obou případech jsou názvy dlouhé a nepohodlné.

Datové typy spravovaných objektů jsou definovány pomocí podmnožiny ASN.1 (Abstract Syntax Notation One). ASN.1 je způsob specifikace, jak jsou data reprezentována a přenášena mezi správci a agenty v kontextu SNMP. Specifikace je platformně nezávislá a není potřeba se obávat o správnou posloupnost přenášených bytů.

Spravovaná instance objektu je zakódována do řetězce oktetů využívajícího formát BER (Basic Encoding Rules). Formát BER, definuje, jak jsou objekty zakódovány a dekódovány, aby mohly být přenášeny po transportním médiu, jako je ethernet [1].

3.2.1 Hierarchie OID

Spravované objekty jsou uspořádány do stromové struktury. Jednotlivé objekty jsou reprezentovány posloupností čísel podle uzlů ve stromu. Uzly jsou odděleny tečkami, aby byla zobrazena hierarchická posloupnost. Existuje přívětivější forma v podobě textových řetězců, které jsou opět odděleny tečkami.



Obrázek 4: Zobrazení hierarchie OID v MIB souboru [1]

Na nejvyšší úrovni stromu se nachází uzel, který se nazývá kořen stromu. Pokud má uzel potomky, jedná se o podstromy. Uzly bez potomků se nazývají listy. Kořen má tři potomky: ccitt(0), iso(1) a joint(2). Listy ccitt(0) a joint(2) se nevztahují k SNMP. Podstrom iso(1).org(3).dod(6).internet(1), také označovaný 1.3.6.1, je nejdůležitější pro organizace. Větev directory(1) není v současné době využívána. Větev mgmt(2) definuje standardní množinu objektů pro správu sítě. Větev experimental(3) je určena pro účely testování a výzkumu. Větev private(4) aktuálně obsahuje jeden podstrom enterprises(1), kde si výrobci hardware a software mohou definovat vlastní objekty. Obecné vytvoření identifikátoru objektu se řídí následující syntaxí: [1]

název_objektu OBJECT IDENTIFIER ::= { *cesta_k_rodičovskému_objektu číslo_nového_objektu*
}

Ukázka vytvoření podstromu internet s využitím výše zmíněné notace je následující: [3]

internet OBJECT IDENTIFIER ::= { iso org(3) dod(6) 1 }
directory OBJECT IDENTIFIER ::= { internet 1 }
mgmt OBJECT IDENTIFIER ::= { internet 2 }
experimental OBJECT IDENTIFIER ::= { internet 3 }
private OBJECT IDENTIFIER ::= { internet 4 }

Pokud je vytvořen identifikátor daného uzlu, je dalším krokem definice objektu, která využívá následující syntaxe: [1] [4]

<*název*> OBJECT-TYPE
SYNTAX <*datový typ*>
ACCESS < *read-only / read-write / write-only / not-accessible*>
STATUS <*mandatory / optional / obsolete*>
DESCRIPTION *"Textový popis objektu“* ::= { <*unikátní OID, které definuje tento objekt*>
}

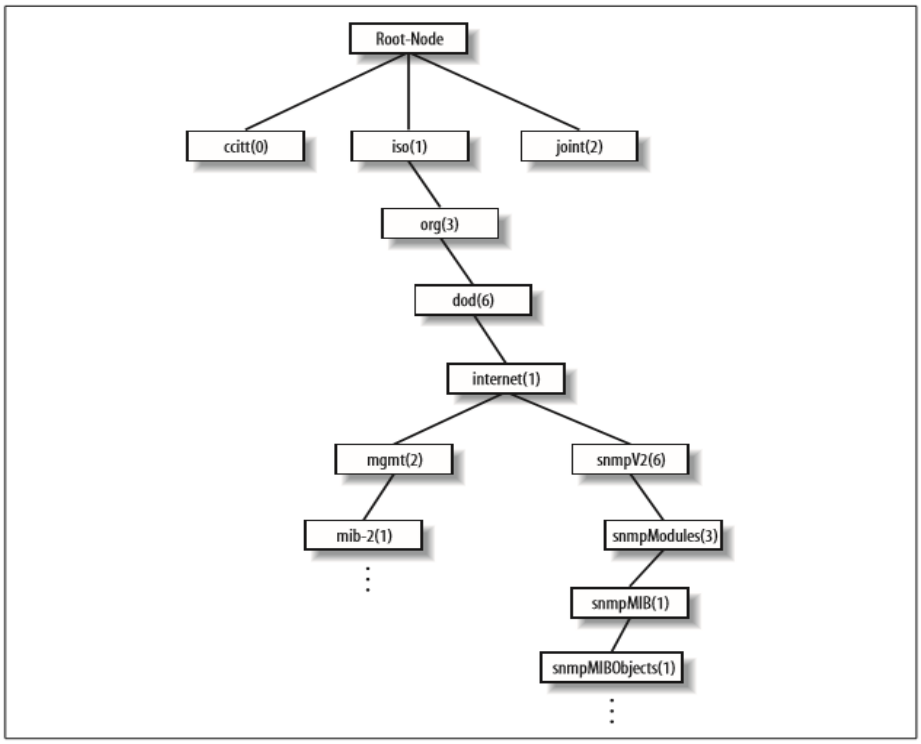
3.2.2 Datové typy SMIv1 a SMIv2

Structure Management Information verze prvé nabízí několik typů, které lze přiřadit v definici objektu [1]:

- INTEGER – 32 bitové číslo často používané jako výčtový typ v kontextu spravovaného objektu. Výčet musí začínat 1 dle RFC1155.
- OCTET STRING – řetězec bytů reprezentující text Counter – 32 bitové číslo s minimem 0 a maximem 2^{32}. Hodnota se stále zvyšuje. Nikdy by nemělo dojít ke snížení při běžných operacích. Pouze při dosažení maxima se změní hodnota na 0. Při restartování zařízení se zapíše do proměnné 0.

- OBJECT IDENTIFIER – řetězec, který je složen z číslic a teček, identifikující spravovaný objekt ve stromové struktuře.
- SEQUENCE – definuje seznam obsahující žádný, nebo více datových typů ASN.1 normy.
- SEQUENCE OF – definuje spravovaný objekt vytvořený z datového typu SEQUENCE.
- IpAddress – reprezentuje IPv4 adresu. SMIv1 a SMIv2 nedefinuje IPv6 adresy.
- NetworkAddress – stejný typ jako IpAddress, ale může reprezentovat různé typy adres.
- Gauge – 32 bitové číslo. Minimální hodnota je 0 a maximální 2^{32}. Nesmí překročit maximální hodnotu jako datový typ Counter. Rychlost rozhraní bývá měřena tímto typem.
- TimeTicks – 32 bitové číslo reprezentující čas v setinách. Zpravidla bývá tímto typem měřen čas od spuštění zařízení.
- Opaque – umožňuje specifikovat ASN.1 BER kódování.

SMIv2 rozšiřuje objekt SMIv1 přidáním větve snmpV2 do podstromu internet. Přidává nové názvy datových typů a dělá několik změn.



Obrázek 5: Zobrazení hierarchie OID v MIB souboru s rozšířením snmpV2

Přidané názvy datových typů ve verzi SMIv2 jsou:

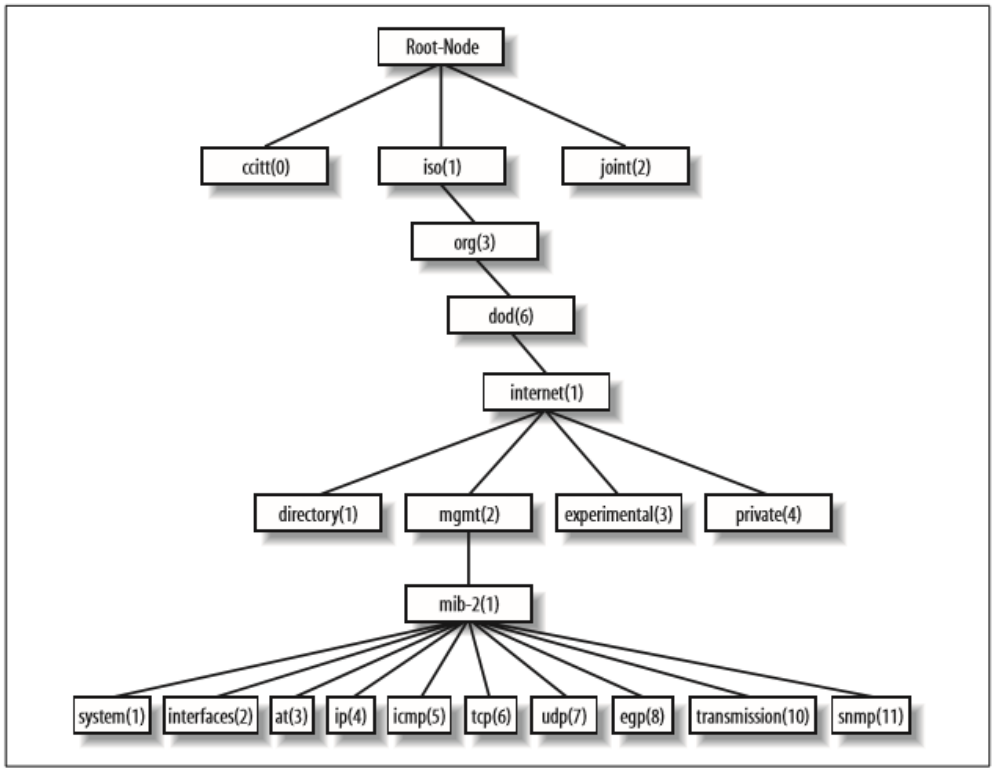
- Integer32 – stejné vlastnosti jako datový typ INTEGER
- Counter32 - stejné vlastnosti jako datový typ Counter
- Gauge32 - stejné vlastnosti jako datový typ Gauge
- Unsigned32 - reprezentuje rozsah $0\text{-}2^{32}\text{-}1$
- Counter64 - podobné Counter32, ale maximální hodnota je 2^{64}-1. Counter64 se využívá v situacích, kdy Counter32 přeteče a vrátí se na 0.
- BITS - výčet pozitivně pojmenovaných bitů.

Definice objektů ve SMIv2 se mírně liší od verze první. Umožňuje přidat nepovinná pole, která nabízí větší kontrolu řízení přístupu k objektu. Lze rozšiřovat tabulku přidáním sloupců a zlepšit popis. [1]

<*název*> OBJECT-TYPE
SYNTAX <*datový typ*>
UnitsParts <*nepovinný textový řetězec popisující jednotky*>
MAX-ACCESS <*read-only/read-write/write-only/not-accessible/accessible-for-notify* >
STATUS < *mandatory / optional / obsolete / current / deprecated* >
DESCRIPTION *"Textový popis objektu"*
AUGMENTS { <*jméno tabulky, která je rozšiřována*> }
::= { <*unikátní OID reprezentující objekt*> }

3.2.3 MIB-2

MIB-2 je velmi důležitá skupina, protože každé zařízení podporující SNMP musí také podporovat MIB-2.



Obrázek 6: Zobrazení podstromu mib-2 [1]

- system (1.3.6.1.2.1.1) - definuje seznam objektů, které se týkají systémových operací, jako je doba běhu systému, systémové jméno, systémový kontakt.
- interfaces (1.3.6.1.2.1.2) - udržuje informace o stavu každého zařízení na spravovaném systému. Udržované informace se týkají počtu bytů, které přišly a odešly ze spravovaného zařízení, počet chyb a další.
- at (1.3.6.1.2.1.3) - skupina překladu adres je zastaralá a poskytována pouze pro zpětnou kompatibilitu.
- ip (1.3.6.1.2.1.4) - uchovává informace spojené s IP, včetně IP směrování.
- icmp (1.3.6.1.2.1.5) - sleduje věci jako ICMP chyby, zahození a další.
- tcp (1.3.6.1.2.1.6) - sleduje mimo jiné stav TCP spojení.
- udp (1.3.6.1.2.1.7) - sleduje UDP statistiky, vstupní a výstupní datagramy.

- egp (1.3.6.1.2.1.8) - poskytuje informace o statistikách EGP (Exterior Gateway Protocol) a udržuje EGP tabulku sousedů.
- transmission (1.3.6.1.2.1.9) – prozatím žádný objekt není nyní definován v této skupině.
- snmp (1.3.6.1.2.1.10) - měří výkon SNMP implementace spravované entity a sleduje statistiky, jako je počet přijatých a odeslaných SNMP paketů [1].

3.3 Monitorování s agenty a bez agentů

Ve světě monitorování se rozlišují dva základní přístupy. Monitorování s agenty a bez agentů. IBM Tivoli Monitoring poskytuje agenty pro operační systémy, které monitorují dostupnost a výkon koncových stanic v podniku. Agent musí být nainstalován na koncové stanici, kterou monitoruje. Příkladem může být Windows OS agent.

IBM Tivoli Monitoring také umožňuje monitorování bez agentů. Jedná se o standardního agenta, který monitoruje operační systémy běžící na vzdálených uzlech. Sledované uzly nemají nainstalovány žádné plně funkční OS agenty. Monitorování vzdálených uzlů probíhá pomocí specifického API, SNMP, CIM nebo WMI. Používaná rozhraní poskytují informace o operačním systému a základních funkcích aplikací. Není potřeba instalovat žádné komponenty systému IBM Tivoli Monitoring.

Ve vývojovém prostředí IBM Agent Builder lze vytvořit agenty na míru, které využívají zmíněných API a jsou odděleny od agentů na instalačním médiu.

Funkcionalita monitorování bez agentů probíhá stejně jako s agenty. Nasbíraná data jsou distribuována na Tivoli Enterprise Monitoring Server a Tivoli Enterprise Portal Server. Lze vytvářet standardní pohledy, situace, politiky a provádět vzdálené nasazení. Výhodou je, že agent monitorující vzdálené uzly může sledovat informace z jiné platformy, než na které je nainstalovaný. Čili agent nainstalovaný na systému Windows může sledovat server s Linuxem. IBM Tivoli Monitoring umožňuje na jednom systému nainstalovat až deset instancí agenta sledujících vzdálené systémy. Každá instance může monitorovat až sto vzdálených systémů.

Mezi základní metriky, které sbírá agent, patří využití fyzických a logických disků, využití sítě, informace o virtuální a fyzické paměti, informace o systému, agregované informace o využití procesoru a spuštěné procesy. Standardně jsou připraveny situace pro hlášení přesažených hranic [5].

3.3.1 Common Information Model

CIM (Common Information Model) je multiplatformní standard spravovaný DMTF (Distributed Management Task Force). CIM je rozšiřitelný, objektově orientovaný datový model, který obsahuje informace o různých částech podniku. Vývojáři mohou vytvářet třídy a dědit od tříd, které reprezentují pevné disky, aplikace, síťové prvky nebo uživatelsky definované technologie. Definice jednotlivých tříd využívá podobných konstrukcí jako C++. Odvozené třídy přejímají stejné atributy a metody svých předků. Celý řetězec dědičností je obsažen v poli __Derivation. Třída může obsahovat vlastnosti pro popis dat a metody pro popis chování. CIM definuje tři úrovně tříd:

- Core – třídy Core reprezentují spravované objekty, které jsou aplikovány ve všech oblastech správy. Poskytují základní slovník pro analýzu a popis spravovaných systémů. Příkladem jsou třídy __Parameters a __SystemSecurity.
- Common – třídy Common představují spravované objekty, které jsou aplikovány ve specifických oblastech. Rozšiřují základní třídy Core, jsou však nezávislé na dané implementaci nebo technologii. Příkladem může být třída CIM_UnitaryComputerSystem.
- Extended – třídy Extended reprezentují objekty, které jsou technologicky specifickými rozšířeními tříd Common. Rozšíření se typicky týká specifických platforem jako je UNIX nebo Win32 prostředí. Příkladem Extended třídy je Win32_ComputerSystem [6].

3.3.2 Windows Management Instrumentation

WMI (Windows Management Instrumentation) je implementace WBEM (Web Based Enterprise Management) od společnosti Microsoft. Využívá standardu CIM pro reprezentaci systémů, aplikací, sítí, zařízení a dalších spravovaných komponent v podnikovém prostředí. Pomocí WMI je možné sbírat data ze vzdálených systémů běžících na platformě Windows. Spojení je provedeno pomocí DCOM (Distributed Component Object Model) nebo WinRM (Windows Remote Management) [7].

3.4 Standardy v oblasti logování

Zaznamenávání informací o podstatných stavech běžícího software je bezesporu jedna z nejdůležitějších věcí při správě. Pokud se vyskytne chyba, správce systému nebo podpora musí zjistit, co způsobilo daný problém a pokusit se o nápravu. Neřešením chybových stavů může dojít ke ztrátě dat, zisku a jiným komplikacím. Prevence je tedy namístě. Je velice nápomocné, když aplikace, operační systém a další systémové služby zaznamenávají důležité události, jako je nedostatek paměti, nadměrný přístup k pevnému disku, bezpečnostní informace nebo interní chyby. Systémový administrátor může využít logovací systém a uložené informace k identifikaci a nápravě chybového stavu. Zaznamenávání událostí je výhodné i při vývoji nového software. Zkracuje se doba nutná k ladění a vývoji komponent.

Každý logovací systém by měl dodržovat základní konvence pro ukládání záznamů. Dodržováním konvencí je usnadněna identifikace problémů a případné monitorování. Zaznamenávány by měly být pouze relevantní informace, které jsou užitečné pro diagnostiku problému. Důležitý je také návrh samotných zpráv. Dobře navržené zprávy mohou redukovat náklady na technickou podporu a zjišťování příčiny problému. Zprávy by měly obsahovat následující informace:

- co se stalo
- co výsledek znamená pro koncového uživatele
- co může uživatel udělat, aby se chyba neopakovala.

Zde je důležité si uvědomit, že cílová skupina nemusí rozumět použité terminologii. Pokud je zpráva určena pro různé skupiny příjemců, je vodné vytvořit speciální zprávy pro každou skupinu. Mezi základní skupiny můžeme považovat vývojáře, administrátory a koncové uživatele. Ideálním stavem je nepoužívat generické zprávy, které zastřešují několik možných chybových stavů. Vhodným řešením je separace každého chybového stavu a vytvoření jedinečných zpráv. Pokud má řešení problému více kroků, je vhodné odkázat na pomoc nebo podporu. Zprávy by neměly obsahovat tabulátory nebo čárky. Logovací systém může využívat čárky a tabulátory jako oddělovače. Záznamy je důležité označit časovým razítkem, aby došlo k přesnému ukotvení v čase. Jednotlivé záznamy by měly mít identifikátor závažnosti a původ vzniku. Podstatné je, aby záznamy ze stejného informačního okruhu byly ukládány do logicky rozlišitelných souborů. V praxi se může stát, že vývojáři

různých komponent software ukládají záznamy do jednoho souboru. Separace jednotlivých zpráv může být obtížným úkolem při monitorování. Dobrým zvykem bývá rotace logovacích souborů při dosažení maximální velikosti souboru [8] [9].

3.4.1 Windows Event Log

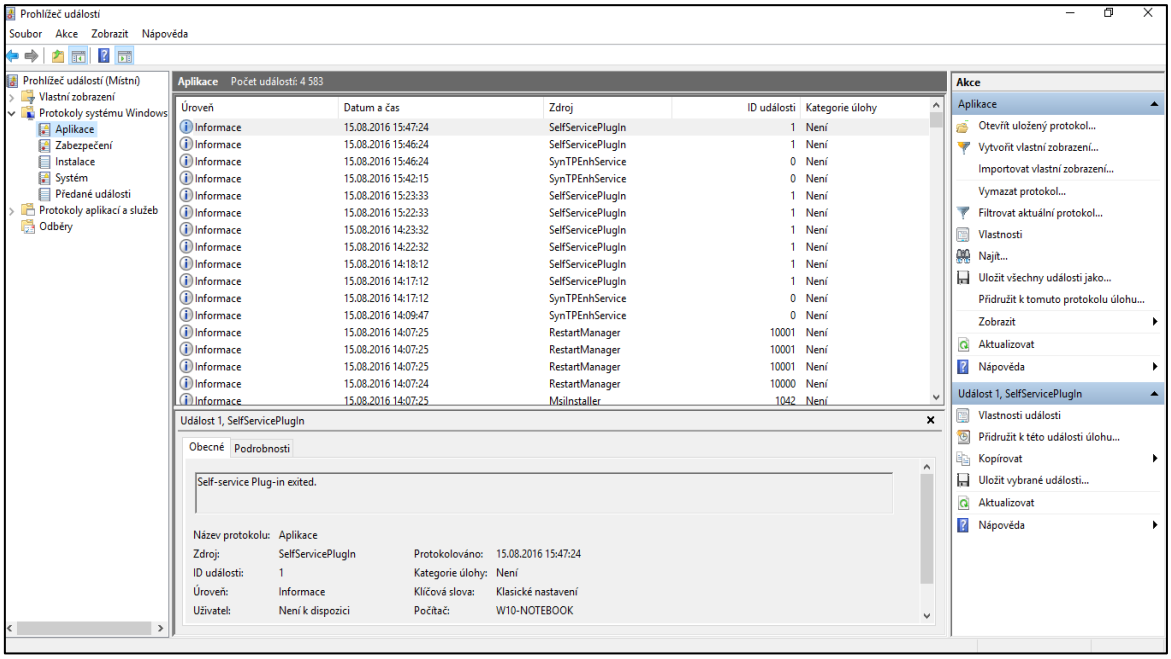
Základním systémem pro správu událostí v Microsoft Windows je Windows Event Log. Windows Event Log obsahuje dvě kategorie logů: Protokoly systému Windows a Protokoly aplikací a služeb. K prohlížení logů je možné využít Prohlížeč událostí nebo v příkazové řádce nástroj wevtutil.

Protokoly systému Windows obsahují logy Aplikací, Systému, Zabezpečení, Instalace a Předaných událostí. Všechny zmíněné podkategorie mají dopad na chod celého systému.

Kategorie Protokoly aplikací a služeb ukládá události z jednotlivých aplikací nebo komponent. Uložené události nemají dopad na celý systém, jako je tomu v kategorii Protokoly systému Windows. Obsahuje čtyři podtypy logů: Admin, Operational, Analytic a Debug. Události typu Admin poskytují informace o tom, jak řešit konkrétní problém. Cílovou skupinou jsou administrátoři, koncoví uživatelé a technická podpora. Události typu Operational obsahují informace, které je nutno dále analyzovat a interpretovat. Podtyp Analytic obsahuje data trasování problému. Obvykle bývá velmi rozsáhlý. Debug je určen především pro vývojáře aplikací. Analytic a Debug jsou ve výchozím stavu skryté a vypnuté.

Informace o každé události odpovídají XML schématu. Uživatelé mohou vytvořit XML dotazy proti logům s událostmi. Prohlížeč událostí poskytuje grafické rozhraní pro snadné pokládání dotazů.

Jednotlivým událostem, které se vyskytnou, lze přiřazovat úlohy. Správce systému může vytvořit automatizovaný skript, který určitým způsobem reaguje a vyřeší konkrétní problém [10].



Obrázek 7: Ukázka Prohlížeče událost v systému Microsoft Windows - vlastní zpracování

3.4.2 Syslog

Syslog je standard, jak síťová zařízení posílají události logovacímu serveru. Protokol Syslog je podporovaný velkou škálou síťových zařízení, jako jsou firewally, Linuxové servery, některé tiskárny, webové servery a další. Například směrovač může zaslat zprávu o uživatelích, kteří se přihlásili do konzole, nebo zamítnutý pokus o přihlášení do webového rozhraní. Systémy Microsoft Windows nepodporují nativně Syslog, ale existují nástroje, které umožňují sbírat data z Windows Event Logu a přeposílat je Syslog serveru.

Syslog server je vhodný systém pro konsolidaci logů z mnoha zařízení. Běžnými komponentami implementací Syslog serveru jsou:

- Syslog listener - naslouchá na UDP portu 514. Zprávy nejsou potvrzovány. Nelze tedy určit, jestli zprávy dorazily do cílového zařízení. Některá zařízení zasílají data pomocí TCP na port 1468, aby nedošlo ke ztrátě dat.
- Databáze – rozsáhlé sítě generují velké množství dat. Pro rychlé načítání dat používá Syslog server databázové úložiště.
- Software pro správu a filtrování – pro snazší správu, filtrování a upozornění na zaznamenané události slouží uživatelsky přívětivé rozhraní. Správci zařízení se dozví o problému ihned, jakmile se vyskytne.

Nevýhodou Syslog serveru je, že zprávy nemají definovaný formát. Každý vývojář formátuje zprávy dle svého rozhodnutí. Některé zprávy jsou obyčejné textové a některé jsou v binární podobě. Jelikož je využívána nespojovaná komunikace, nemusí zprávy dorazit k cíli.

K využití v jazyce C je nutné naimportovat hlavičkový soubor syslog.h. Pro otevření připojení k logovacímu systému je využita funkce *openlog* a záznam je proveden pomocí funkce syslog. K identifikaci, odkud pochází zpráva, je použito označení *facility*. Facility má až 24 různých hodnot. Zprávy mohou například pocházet z jádra systému, uživatelské úrovně, démona, bezpečnostního programu a dalších. Každá zpráva má přiřazenu úroveň závažnosti [11] [12] [13]:

- LOG_EMERG – systém je v nepoužitelném stavu
- LOG_ALERT – musí být okamžitě provedena akce
- LOG_CRIT – kritický stav
- LOG_WARN – varování
- LOG_NOTICE – běžné, ale podstatné oznámení
- LOG_INFO – informační zpráva
- LOG_DEBUG – ladicí zprávy.

3.5 Nástroje pro vytváření agentů

Pro vytváření agentů na zakázku je dobré znát několik prostředků, které umožňují a zjednodušují jejich vývoj.

V oblasti monitorování se velmi často stává, že je potřeba sledovat soubory logů konkrétních aplikací. Soubory je nutné prohledávat a zjišťovat, jestli obsahují konkrétní data, a pracovat se specifickými částmi konkrétního řetězce. Pro práci se soubory, které obsahují textová data, jsou nejvhodnějším prostředkem regulární výrazy.

Při vývoji složitějších agentů, kdy je potřeba naprogramovat rozsáhlejší logika, se velmi často používají jazyky Perl, Java, nebo specifické jazyky pro cílovou platformu. Pro konkrétní jazyk je vhodné mít připravené integrované vývojové prostředí, kde je možné testovat, spouštět a vytvářet na platformě spustitelné soubory.

Systém IBM Tivoli Monitoring vyžaduje pro tvorbu agentů vývojové prostředí IBM Tivoli Monitoring Agent Builder. V několika základních průvodcích lze vytvářet různé typy agentů bez nutnosti programování. Pro složitější agenty, kteří nejsou mezi šablonami, je nutné vytvořit vlastní spustitelný program, který je zabalen v IBM Agent Builderu a nainstalován do monitorovacího systému. Praktická část ukáže, jak lze agenta vytvořit a nainstalovat [14].

3.5.1 Regulární výrazy

Regulární výrazy jsou mocným prostředkem pro snadnou práci s textovými řetězci. Základními prvky regulárních výrazů jsou metaznaky a literály.

Pro vyhledávání v textu se nejdříve specifikuje regulární výraz, který je vzorem hledaného řetězce. Při průchodu textem se postupně vyhodnocuje splnění regulárního výrazu. Pokud dojde ke shodě, je nahlášen výskyt.

Regulární výrazy lze použít v mnoha programovacích jazycích. Pro účely monitoringu a vytváření agentů na zakázku se využívají multiplatformní jazyky, jako jsou například Perl a Java. Oba implementaci regulárních výrazů obsahují. Manipulace s textem může být také prováděna za pomoci specializovaných programů, jako jsou grep, egrep, fgrep, awk, sed a další. V dnešní době můžeme nalézt rozsáhlou podporu regulárních výrazů i v textových editorech [15].

Příklady regulárních výrazů:

^ - počátek řádky

$ - konec řádky

[…] - uzavírá třídu možných znaků na dané pozici

[a-z] - rozsah možných znaků

[^…] - uzavírá třídu znaků, které se nesmí vyskytovat na dané pozici

[^a-z] - rozsah zakázaných znaků

* - žádný nebo více výskytů předcházejícího výrazu

+ - jeden nebo více výskytů předcházejícího výrazu

? – žádný nebo jeden výskyt předcházejícího výrazu

. - jakýkoliv znak; ve víceřádkovém režimu i odřádkování

() - uzavření do zachycující skupiny

(?:) - uzavření do nezachytávající skupiny

{n} - n krát opakování předcházejícího výrazu; pro n kladné

{n,m} - n až m-krát opakování předcházejícího výrazu; pro n, m nezáporná

x|y - buď výraz x, nebo výraz y

\d - desítková číslice [0-9]

\D - jakýkoliv znak vyjma číslic [^0-9]

\s - neviditelné znaky

\S - jakýkoliv znak vyjma neviditelného [^\s]

\w - jakýkoliv alfanumerický znak včetně podtržítka

\W - jakýkoliv nealfanumerický znak [^\w]

\t - tabulátor

\r - návrat vozíku

\n - nová řádka

[15].

Pro testování regulárních výrazů je vhodné používat integrované vývojové prostředí, jako je Eclipse. V Eclipse Marketplace lze nalézt zásuvný modul QuickREx. Po instalaci lze ihned zkoušet regulární výrazy v praxi. Na internetu je možné nalézt online nástroje pro testování, které pracují přímo ve webovém prohlížeči. Na odkazech http://www.regexp.cz/ a http://myregexp.com/ si lze takové nástroje otestovat.

3.5.2 Perl

Pro vývoj agentů na míru je vhodné použít jazyk, který je přenositelný a lze v něm co nejsnáze vyjádřit potřebný algoritmus. Specifické agenty, které využívají připojení k databázi, je vhodnější programovat v jazyce Java za použití JDBC (Java Database Connectivity). Monitorování logovacích souborů a obecně práce s textovými informacemi je snazší v jazyce Perl.

Mezi oblíbenými jazyky v oblasti monitorování se usídlili Perl a Java. Perl je interpretovaný jazyk, který byl vytvořen Larry Wallem v roce 1987. Dnes se původní Perl nachází ve své páté verzi a je stále vyvíjen. Existuje však Perl verze šest, který je v mnoha ohledech odlišným jazykem. Některé syntaktické konstrukce z šesté verze nejsou zpětně kompatibilní s pátou verzí. Jedná se o odlišný jazyk, který je vyvíjen komunitou lidí po celém světě. Pátá verze umožňuje objektový přístup a používat výkonné datové struktury. Perl se řadí mezi jazyky, ve kterých lze jednu a tu samou věc vyjádřit mnoha způsoby. Skripty není nutné kompilovat a linkovat s knihovnami.

Perl bývá standardně přítomen ve světě UNIXU a Linuxových distribucí. Při použití v systémech Microsoft Windows je nutné doinstalovat balík, který obsahuje běhové prostředí nutné pro vývoj a ladění skriptů. Mezi nejoblíbenější distribuce Perlu patří Strawberry Perl a Active Perl.

Pro stahování nových modulů, které lze následně používat ve skriptech, slouží webová stránka http://www.cpan.org. Zde mohou být nalezeny moduly a jejich dokumentace. Stažení samotných modulů je provedeno pomocí nástroje cpan v příkazové řádce. Moduly bývají zpravidla aktuální a připravené k použití. Nástroj cpan je standardně dodáván s distribucí Strawberry Perl. Active Perl obsahuje vlastní nástroj ppm, který stahuje moduly z repositáře na adrese http://code.activestate.com.

Samotný vývoj skriptů je vhodné provádět v integrovaném vývojovém prostředí. Nejvíce osvědčenou platformou pro vývoj v jazyce Perl se ukázalo prostředí Eclipse, které lze stáhnout z odkazu https://www.eclipse.org/downloads/. Samotné prostředí je nutné rozšířit o modul Epic Perl, který je ke stažení na adrese http://www.epic-ide.org/download.php. Editor nabízí zvýrazňování syntaxe a doplňování kódu. V prostředí lze nastavovat různé konfigurace spuštění pro testovací účely. Nechybí ani podpora ladění a krokování. Pro krokování a ladění je nutné mít nainstalovaný modul PadWalker v běhovém prostředí Perlu. K distribuci výsledných skriptů jako spustitelných souborů

bez nutnosti instalace běhového prostředí Perlu slouží modul PAR::Packer, který nainstaluje nástroj pp do příkazové řádky. Do výsledného souboru jsou zabaleny všechny potřebné knihovny pro běh skriptu na cílové platformě [16].

Nedílnou součástí jazyka Perl jsou regulární výrazy. K porovnání řetězce se vzorem se používají operátory =~ nebo !~. Výraz "Řetězec" =~ /vzor/ vrací hodnotu 1, pokud byl v řetězci nalezen vzor. Při nenalezení hledaného vzoru je vrácen prázdný řetězec. Konstrukce pro nahrazování podřetězce novým řetězcem je "Řetězec" =~ s/regulární výraz/nový řetězec/. Za posledním lomítkem u regulárního výrazu mohou být následující modifikátory [15]:

e - nahrazovaný řetězec je výraz pro vyhodnocení – pouze u nahrazování
g – globální zaznamenání všech shodných řetězců
i – nezávislost na velikosti znaků
s – množina značící se tečkou zahrnuje znak nového řádku
x – umožňuje speciální syntaxi regulárních výrazů s komentáři
m – víceřádkový režim
o – vzor je přeložen pouze jednou

Ukázka kódu:

```
my $retezec = "řetězec";
if ($retezec =~ /regulární výraz/){
        print "Shoda!";
}
```

3.5.3 Java

Java je jedním z nejrozšířenějších programovacích jazyků na světě. V roce 1995 se Java dočkala svého prvního vydání. Jazyk je plně objektový. Vytvořené aplikace jsou multiplatformní a běží ve virtuálním stroji. Program je přeložen do tzv. bytekódu, který je interpretován JVM (Java Virtual Machine). Nyní se Java nachází ve své osmé verzi a v roce 2017 je očekáváno vydání deváté verze.

Pro vývoj v jazyce Java je možné použít více integrovaných vývojových prostředí. Mezi nejznámější patří Eclipse, Netbeans a IntelliJ IDEA. K zachování konzistence s produkty IBM Tivoli Monitoring je vhodné používat Eclipse. IBM staví svá vývojová

prostředí nad platformou Eclipse, do kterých implementuje zásuvné moduly. Vývojová prostředí společnosti IBM určená pro monitorování jsou například IBM Agent Builder nebo produkty z řady Rational – RFT a RPT.

Použití regulárních výrazů je možné po naimportování balíčku *java.util.regex.**. Samotná kompilace regulárního výrazu probíhá pomocí funkce Pattern.compile("regulární výraz", modifikátory), která vrací objekty typu Pattern. Dalším krokem je předání řetězce, ve kterém se má hledat shoda. Vrácenému objektu funkce Pattern.compile se zašle zpráva matcher("řetězec"), která vrací objekt typu Matcher. Nalezení podřetězce se testuje posláním zprávy matches vrácenému objektu typu Matcher [17].

Ukázka kódu:

```
Import java.util.regex*;
Pattern pattern = Pattern.compile("regulární výraz",modifikátory);
Matcher matcher  = pattern.matcher("řetězec");
If (matcher.matches()){
System.out.println("Shoda!");
}
```

3.5.4 IBM Agent Builder

Vývoj agentů na míru, které jsou určeny pro prostředí IBM Tivoli Monitoring, probíhá v integrovaném vývojovém prostředí IBM Agent Builder. IBM Agent Builder je postaven nad platformou Eclipse, do které je doinstalován zásuvný modul pro vývoj agentů. Vývojové prostředí Eclipse může být spuštěno na mnoha platformách. Podporované jsou Windows, Linux, AIX a další. Aktuální vývojové prostředí lze stáhnout z webové adresy http://www-01.ibm.com/support/docview.wss?uid=swg24041130. Momentálně se Agent Builder nachází ve verzi 6.3.3. Mimo vývojového prostředí je nainstalována i potřebná IBM Java ve verzi 1.7. Agenty lze cílit na platformy, které jsou podporovány OS agenty.

Mezi podporované platformy patří [14]:

- AIX
- HP-UX
- Linux
- Solaris
- Windows

Instalace vlastního agenta musí být provedena na systém, kde je nainstalován OS agent v příslušné 32/64-bitové verzi.

Než započne samotné vytváření agenta, je vhodné mít rozmyšlený název agenta, kód agenta, druh agenta, cílovou platformu a datové typy sbíraných dat.

Po spuštění vývojového prostředí je možné v menu File->New->Agent vytvořit nového agenta nebo zvolit ikonu pro vytvoření nového agenta.

Obrázek 8: Zobrazení nabídky pro vytvoření nového agenta - vlastní zpracování

Otevře se průvodce, ve kterém jsou zvoleny parametry agenta. Prvními parametry jsou název projektu a umístění. Samotné pojmenování agenta se nachází na další obrazovce průvodce. Důležité je mít na mysli, že jméno agenta bude zobrazeno v monitorovacím systému. Název by měl být výstižný a jedinečný. Agent Builder vyžaduje zadání informace o Copyrightu agenta a pomocí zaškrtávacích nabídek vybrat cílové platformy.

Na další obrazovce průvodce lze nastavit jméno služby pro agenta, které se zobrazí ve správci služeb Manage Tivoli Monitoring Services.

Každý agent musí být označen jednoznačným produktovým kódem v daném prostředí. Některé produktové kódy jsou rezervované společností IBM. Produktový kód je složen ze tří znaků a povolené vzory jsou: K00-K99, K{0-2}{A-Z}, K{4-9}{A-Z}. Pokud se autor agenta rozhodne pro prodej a sdílení agenta s ostatními, je potřeba zaslat informaci na e-mailovou adresu toolkit@us.ibm.com a požádat o rezervaci daného produktového kódu [14].

Nedílnou součástí je nastavení identifikátoru společnosti a jednoznačného identifikátoru agenta. Identifikátor agenta je řetězec složený z alfanumerických znaků anglické abecedy. Kombinovaná délka řetězce identifikátoru společnosti a identifikátoru agenta nesmí překročit 11 znaků.

Následující položkou je verze agenta. Verze je složena ze tří čísel, která jsou oddělena tečkami. Formát je tedy V.R.R. Symbol V reprezentuje hlavní verzi agenta a R.R vydání a opravu. V editoru agenta lze nastavit opravnou verzi tak, že nedojde ke změně hlavní verze. Pokud na monitorovaném systému poběží více instancí agenta a je potřeba je rozlišit, je možné zaškrtnou pole Support multiple instance of this agent.

Agent má již nastavené identifikační parametry a základní charakteristiky. Nyní průvodce nabídne zdroje dat pro monitorování. Agent může mít definovaný větší počet zdrojů dat. Jednotlivé zdroje dat patří do následujících kategorií:

- A proces
  - A proces
  - A Windows service
- Data from a server
  - WMI
  - Perfmon
  - CIM
  - SNMP
  - SNMP Events
  - JDBC
  - JMX
  - HTTP
  - SOAP
- Network management data
  - Ping
- Logged data
  - A Log File
  - AIX Binary Log
  - Windows Event Log

- Command or script
  - A command return code
  - Output from a script
- Custom programs
  - Socket
  - Java API

Zvolením zdroje se otevře průvodce konfigurací pro daný zdroj. Každý z datových zdrojů má své specifické konfigurační parametry. Například pro monitorování procesu je nutné vyplnit název procesu, a jestliže je požadována podrobnější identifikace procesu, pak je možné zadat i přesnou podobu předaných parametrů příkazové řádky.

Monitorování pomocí JDBC vyžaduje specifikaci JDBC ovladačů pro cílovou databázi, přihlašovací údaje k databázi a SQL dotaz, který vrací monitorovaná data. Nedílnou součástí je cesta k JVM (Java Virtual Machine) a možná specifikace parametrů.

U monitorování SNMP je nutná identifikace hostitele a port, na kterém probíhá komunikace. Standardně se jedná o port 161. Uživatel vybírá verzi SNMP a od toho se odvíjí následná nastavení. Pro verzi první a druhou je nutná specifikace hesla – community string. Pro třetí verzi je potřeba vyplnit uživatelské jméno, úroveň zabezpečení, autentizační protokol, autentizační heslo, protokol soukromí a soukromé heslo.

Agent monitorující Windows Event Log vyžaduje název množiny logů na sledovaném systému. Filtrovat lze podle typů, zdrojů a identifikátorů událostí. Pokud je agent zastaven, je možné nastavit, co bude provedeno s offline událostmi.

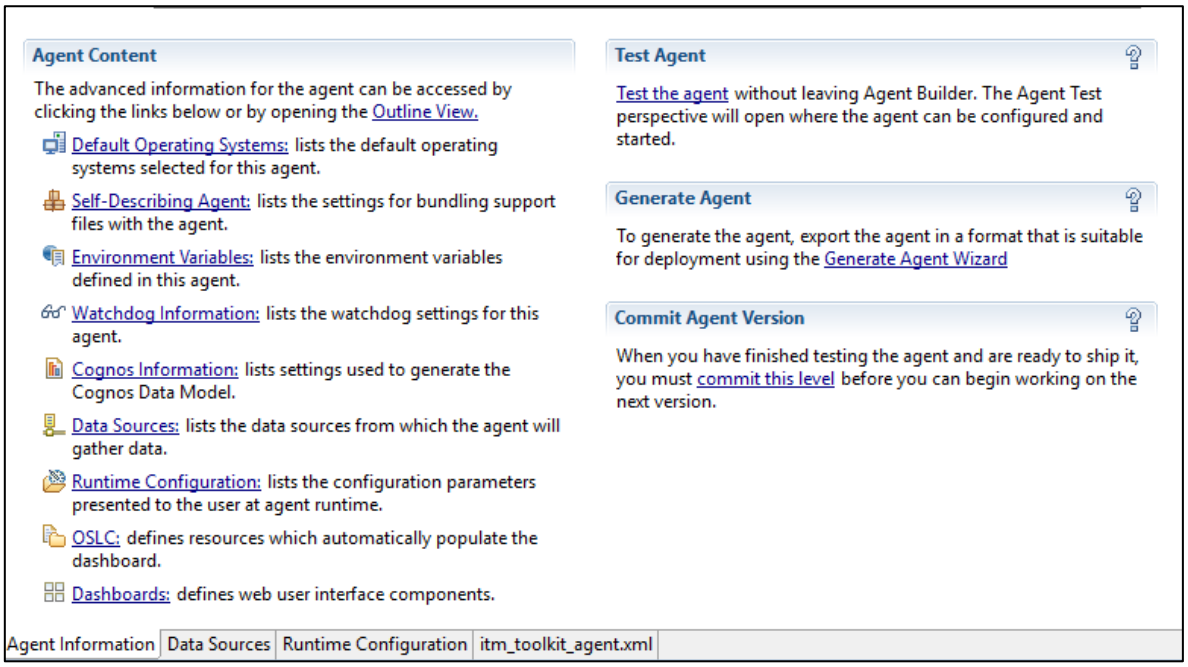
Pro monitorování logů je nutné zvolit, jaké soubory budou monitorovány a specifikovat konkrétní podobu jednotlivých záznamů a polí v nich obsažených. Jednotlivé záznamy mohou být zpracovány jako vzorkované události nebo jako ryzí události.

Ve vývojovém prostředí Agent Builder je možné vytvořit agenty, které zpracovávají vstupy z TCP/IP soketu. Nejdříve je nutné vytvořit agenta, který bude naslouchat na určitém portu. Klientská aplikace se následně připojí ke zvolenému portu a pomocí zpráv, které dodržují XML standard, posílá data do systému IBM Tivoli Monitoring.

Největší volnost získává vývojář se zvolením šablony Output from a script. Pokud ostatní šablony v prostředí Agent Builder nesplňují požadavky na potřeby monitorování, lze vytvořit téměř jakýkoliv program, jehož výstup bude předán do monitorovacího systému.

Datové zdroje vrací řádky, které se skládají z několika sloupců. Každý sloupec má definovaný název, datový typ a nápovědu. Pro rozlišení jednotlivých sloupců je nutné zvolit oddělovač. Povolené datové typy jsou řetězec, číslo a časová známka. U řetězců lze nastavit maximální délku. Číselné hodnoty mohou být 32 nebo 64 bitové. Lze nastavit i přesný rozsah a konkrétní účel hodnoty.

Pokud datový zdroj vrací více řádků, znamená to, že každý řádek reprezentuje právě jednu monitorovanou entitu. Pro unikátní označení monitorované entity je nutné označit alespoň jeden sloupec jako klíčový. Rozlišením jednotlivých entit lze pro každou vypočítat rozdílové hodnoty a míru růstu. Sumarizační a prořezávací agent sbírá souhrnná data pro každou entitu zvlášť. Označení je provedeno pomocí zaškrtávacího tlačítka u konkrétního sloupce [14].



Obrázek 9: Zobrazení konfiguračních možností agenta – vlastní zpracování

Před zabalením a vygenerováním instalačního balíčku agenta je možné nastavit další vlastnosti. Položka Self-Describing Agent říká, jestli se mají do instalačního balíčku přibalit soubory podpory pro jednotlivé komponenty produktu IBM Tivoli Monitoring. Ve výchozím stavu jsou přibaleny podpory pro Tivoli Enterprise Monitoring Server, Tivoli Enterprise Portal Server a Tivoli Enterprise Portal Browser.

Editor agenta umožňuje nakonfigurovat vlastní proměnné prostředí, které jsou předány při spuštění agenta.

Pokud by se vytvořený agent choval nějakým způsobem abnormálně, existuje možnost nastavit Watchdog. Nastavení nabízí, jak často se má kontrolovat zdraví agenta. Standardně je nastavena frekvence 180s. Pokud se agent dostane do nestandardního stavu, lze nastavit počet restartování agenta. Pokud je ve 24 hodinách přesažen limit restartování, je informace o selhání zaslána správci systému. Agenta je možné omezit maximální hodnotou alokované paměti.

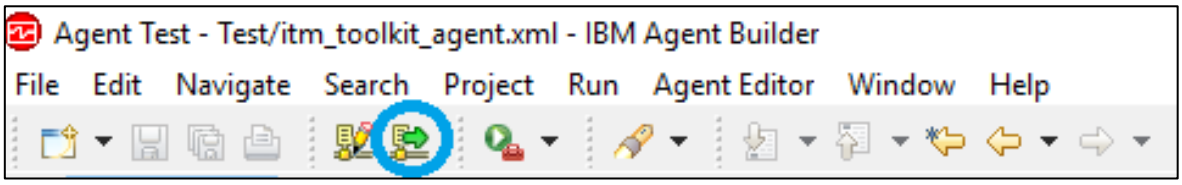
Pokud mají být produkovány zprávy pro Cognos, je možné nastavit datový zdroj, který propojuje Tivoli Common Reporting s IBM Tivoli Data Warehouse. Standardně je nastaveno TDW. Pro úplně kvalifikované názvy tabulek v databázi Tivoli Data Warehouse se musí nastavit název schématu [14].

Při prvotním spuštění agenta je velmi často vyžadována specifická konfigurace a vložení potřebných parametrů pro chod agenta. Editor agenta obsahuje položku Runtime Configuration. Lze vytvořit několik položek, které se zobrazí při konfiguraci agenta. Jednotlivé položky jsou seskupeny do sekcí, kde každá sekce má název a popis. Samotná konfigurační položka má název, popis a jakého typu je. Základní typy jsou: String, Numeric, Password, Choice, Read Only Text, Separator, File Browser a Checkbox. V konfiguračním průvodci se položka zobrazí tak, aby dodržovala zvolený typ. Například typ Password zaručuje, že se vkládané znaky hesla zobrazují jako černé tečky. Podstatnou částí konfigurační položky je název proměnné prostředí. Vyplněním konfigurační položky se vložená hodnota uloží do definované proměnné prostředí a její název je předán agentovi při spuštění.

Po nastavení všech potřebných informací lze přistoupit k testování agenta. Stisknutím tlačítka Test The Agent se otevře perspektiva pro spuštění a testování agenta. Uživatel si může upravit proměnné prostředí a nastavit uživatelské parametry běhového prostředí. Agent provede svou činnost a zobrazí výstupy v podokně editoru. Rozšiřující podrobnosti o spuštění agenta lze vidět v podokně Performance Object Status. Zde se zobrazí informace o úspěšnosti spuštění, návratová hodnota agenta, typ objektu, datum posledního sběru dat, průměrná doba sběru dat a další diagnostické hodnoty, které budou zobrazeny v Tivoli Enterprise Portal.

Když je agent otestován a připraven k distribuci, je dalším krokem vygenerování instalačního balíku. Instalační balík obsahuje soubory agenta a podpory pro jednotlivé

servery infrastruktury. Vytvoření instalačního balíku je provedeno pomocí nabídky Agent Editor -> Generate Agent nebo zvolením ikony pro generování agenta [14].



Obrázek 10: Zobrazení nabídky pro vygenerování instalačního balíčku – vlastní zpracování

3.6 IBM Tivoli Monitoring

Produkty IBM Tivoli Monitoring provádějí dohled na výkon a dostupnost distribuovaných systémů aplikací. Tyto produkty jsou založeny na množině služeb, které se hromadně nazývají Tivoli Monitoring Services. Poskytují bezpečnost, datové přenosy, notifikační mechanismy, uživatelskou prezentaci dat a komunikační služby mezi prvky architektury agent-server-klient.

IBM Tivoli Monitoring podporuje širokou škálu platforem. Základní infrastruktura může být nainstalována na Microsoft Windows Server 2008 až po Microsoft Windows Server 2012. Nelze instalovat na Core edice. Portálový klient a OS agent jsou podporováni od Microsoft Windows Server 2003 až po desktopovou verzi Microsoft Windows 8 [5].

Mimo svět Microsoft Windows je možné nainstalovat monitoring na platformu AIX V6.1 a V7.1, RedHat Enterprise Linux verze 5 a 6, SuSE Linux Enterprise Server verze 10 a 11. Samotný Monitoring Server a Automation Server je možné dále nainstalovat na Solaris V10 a V11, z/OS 1.12 a 1.13. OS monitoring agenti podporují všechny výše zmíněné platformy a navíc HP-UX 11i v2 a v3, i5/OS 5.4, IBM i 6.1, IBM i 7.1 a VMWare ESX Server od verze 3.0.1 až po verzi 4.0 [5].

Typické podnikové prostředí obsahuje následující komponenty:

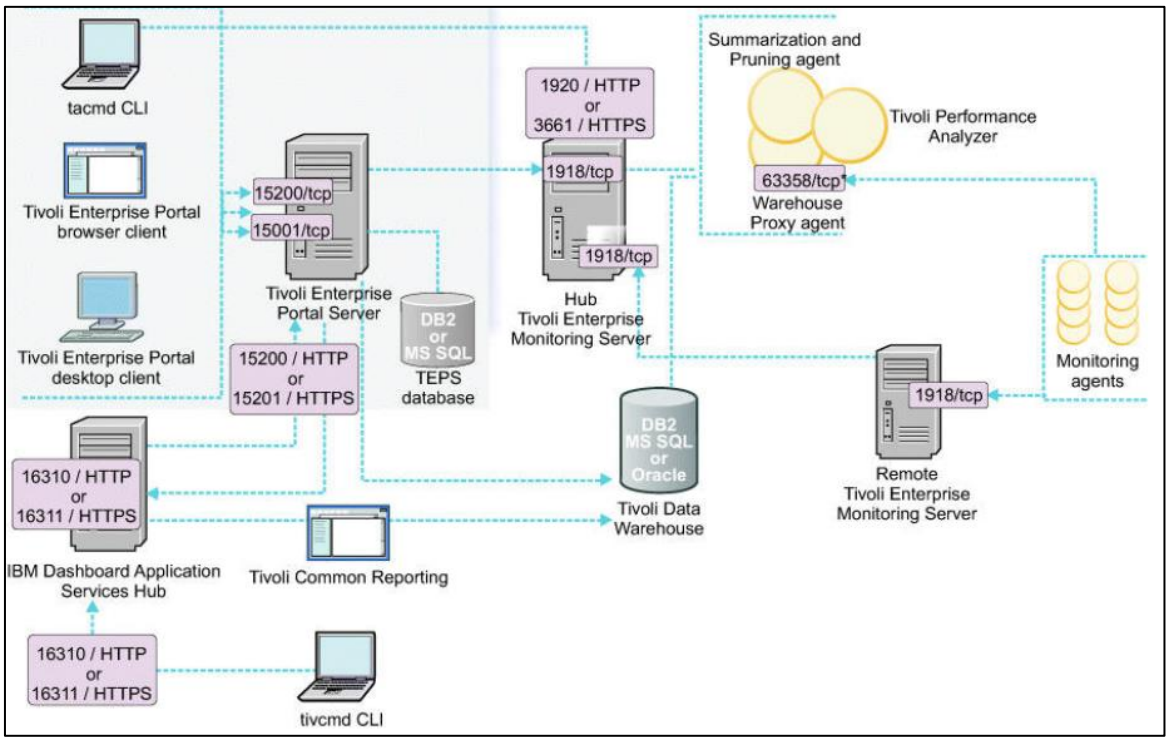
- Jeden nebo více Tivoli Enterprise Monitoring Server, které hrají roli sběrného místa a kontrolního bodu pro upozornění z nasazených agentů. Monitorovací server také spravuje stav připojení jednotlivých agentů. Jelikož Monitorovací server vykonává velké množství funkcí, je nutné distribuovat zatížení mezi více serverů. Jeden ze serverů je určen jako tzv. hub monitoring server. Ostatní servery jsou označovány jako remote servery. Každý remote server musí být nainstalován na vlastním počítači a mít nastaveno unikátní jméno v rámci monitorovacího prostředí.
- Tivoli Enterprise Portal Server, který poskytuje jádro prezentační vrstvy pro získávání, manipulaci, formátování a analýzu dat. Portal Server získává data z Monitoring Serveru jako odpověď na uživatelské dotazy z portálového klienta. Vrácená data jsou uživateli prezentována v portálovém klientovi. Portal Server využívá k ukládání konfiguračních dat databázi IBM DB2, Derby nebo Microsoft SQL Server.
- Jeden nebo více Tivoli Enterprise Portal Client založených na jazyce Java. Klientská aplikace pracuje v režimu desktop a browser. Mezi podporované prohlížeče patří

Microsoft Internet Explorer od verze 8 a Mozilla Firefox verze 10 a 17. Klienta lze spustit v prohlížeči nebo pomocí Java Web Start.

- Tivoli Enterprise Monitoring Agents instalované na sledovaných systémech. Nasbíraná data z agentů jsou distribuována do Monitoring Serveru. Agenti se rozlišují na OS agenty, agentless agenty a ostatní agenty. OS agenti sledují výkon a dostupnost operačního systému. Agentless agenti sledují operační a jiné systémy pomocí SNMP, WMI, CIM a podporovaných API. Agenti sledující určité subsystémy monitorovaného prostředí jsou považováni za kategorii ostatní.
- Jedna nebo více instancí tacmd. Jedná se o nástroj příkazové řádky, který slouží pro automatizaci úkonů v Tivoli Enterprise Portal. Příkazy jsou zasílány Tivoli Enterprise Monitoring Serveru nebo Tivoli Enterprise Portal Server.
- Eclipse Help Server pro prezentaci nápovědy portálu a všech monitorovacích agentů.

Podnikové prostředí může být rozšířeno o následující nepovinné komponenty:

- Tivoli Data Warehouse pro ukládání historických dat, která pochází od nainstalovaných agentů. Data mohou být ukládána v databázi IBM DB2, ORACLE nebo Microsoft SQL Server. Pro ukládání dat v databázi musí být nainstalován Warehouse Proxy Agent. O prořezávání a agregaci dat se stará Summarization and Prunning Agent.
- Event Integration Facility je synchronizační komponenta, která aktualizuje události mezi Netcool/OMNIbus ObjectServer nebo Tivoli Enterprise Console a Tivoli Enterprise Monitoring Serverem.
- IBM Dashboard Application Services Hub je komponenta Jazz for Service Management, která zprostředkovává vizualizaci a oznamování v podobě dashboardu. Pro přístup k dashboardu je využíván webový prohlížeč.
- Sdílený registr uživatelů. Jedná se o LDAP server, jako je Tivoli Directory Server nebo Microsoft Active Directory, který slouží k autentizaci uživatelů v Portal Serveru, IBM Dashboard Application Services Hubu a Netcool/OMNIbus Web GUI. Zprostředkovává single signon funkcionalitu.
- Tivoli Performance Analyzer je komponenta pro predikci budoucího stavu prostředků [5].



Obrázek 11: Zobrazení komponent prostředí IBM Tivoli Monitoring [5]

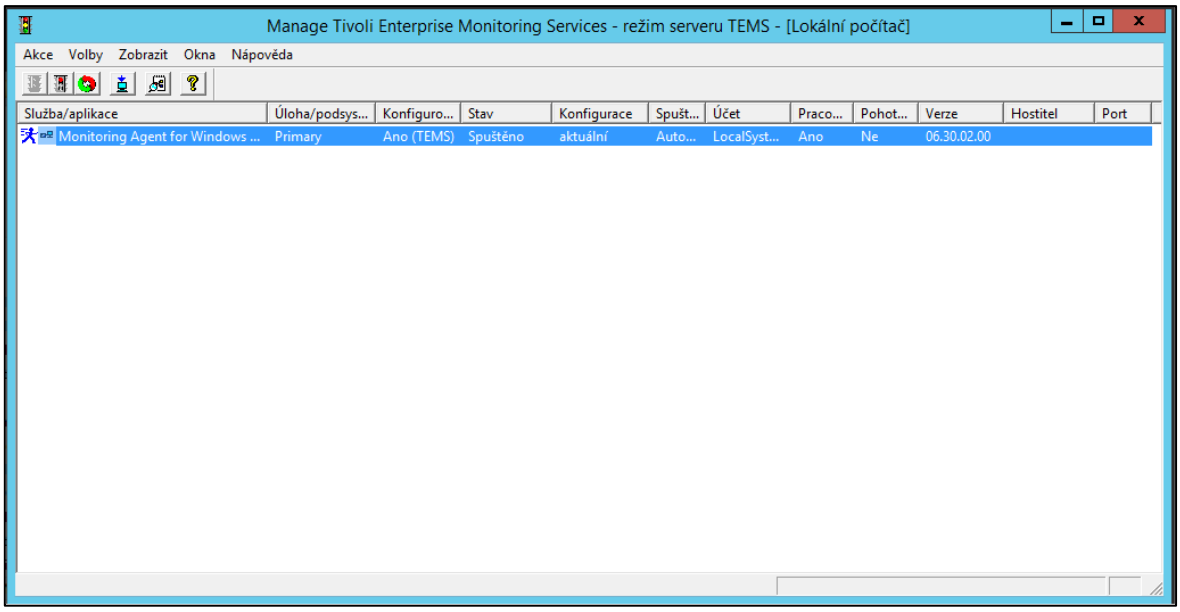
Instalace Tivoli Enterprise Portal obsahuje vnitřní databázi Apache Derby. Apache Derby je mířena na menší až střední podniky využívající maximálně dvacet portálových klientů. Vyžaduje vyšší výkon procesoru a větší množství paměti RAM než IBM DB2.

Pro větší podniky je určena databáze IBM DB2. Instalační médium obsahuje databázi IBM DB2, kterou lze požít pouze pro účely monitoringu. Mezi podporované databáze určené pro Tivoli Enterprise Portal a Tivoli Data Warehouse patří ORACLE 11g, MS SQL Server 2008, IBM DB2 9.7 a jejich vyšší verze [5].

Každá komponenta IBM Tivoli Monitoring má své hardwarové požadavky, které jsou přesně identifikovány v instalační dokumentaci. Tivoli Enterprise Monitoring Server a Tivoli Enterprise Portal Server potřebují dohromady ke svému hladkému běhu 1600 MB paměti RAM a na každých 1000 spravovaných systémů potřebují 220 MB paměti RAM. Instalace TEMS a TEPS zabere 6,7 GB diskového prostoru. Jednotlivé komponenty jsou schopny využívat vícejádrových procesorů. Pro komunikaci mezi komponentami je vhodné mít připojení alespoň 100Mbps [5].

Pokud je nainstalována základní infrastruktura jednotlivých komponent prostředí, je možné přistoupit k instalaci OS agenta. Aby komponenty Tivoli Enterprise Monitoring

Server a Tivoli Enterprise Portal Server rozuměly datům, která získávají od nainstalovaných agentů, je nutné doinstalovat tzv. podpory agentů. Podpory standardních agentů společnosti IBM jsou součástí instalačního média. Vyvinutý agent prostřednictvím IBM Agent Builder má v instalačním balíčku několik instalačních skriptů s podporami. Po instalaci podpor je možné začít sbírat data a vyvářet situace. Instalací OS agenta je nainstalován agent a podpůrný framework pro komunikaci a správu agentů. Pomocí aplikace Manage Tivoli Enterprise Monitoring Services je možné konfigurovat, spouštět a zastavovat jednotlivé agenty.

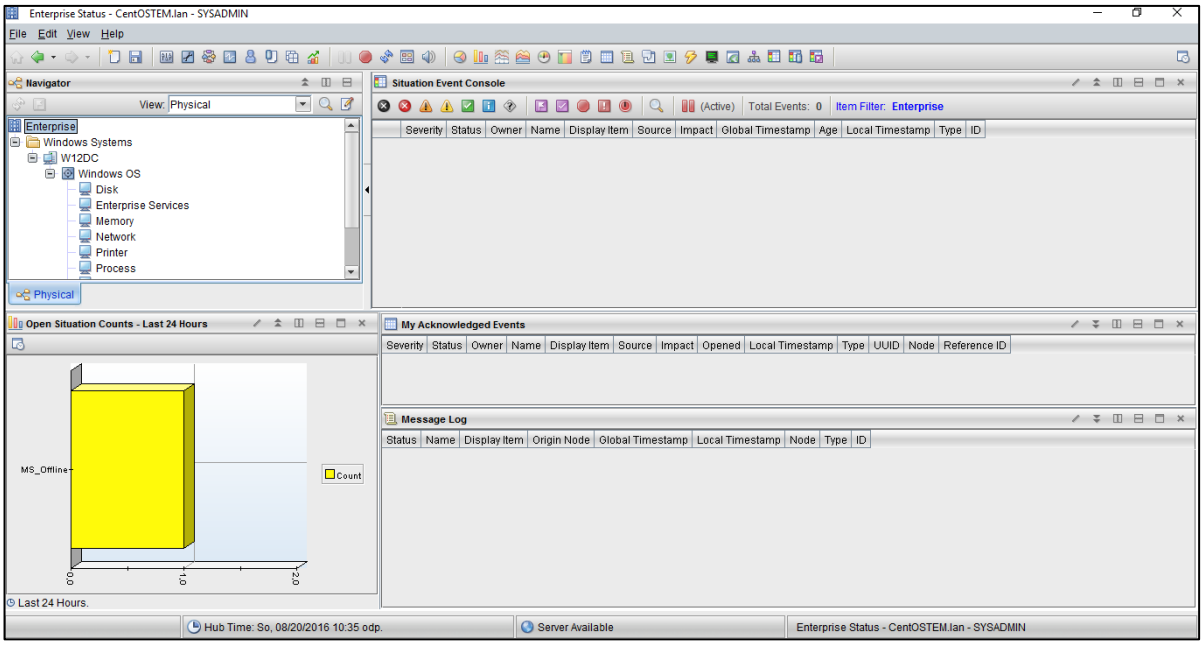


Obrázek 12: Zobrazeni správce služeb prostředí Tivoli Enterprise Monitoring – vlastní zpracování

Uživatelé se přihlašují do prostředí monitoringu prostřednictvím webového prohlížeče, desktopového klienta nebo Java Web Start. Do vyhledávácího řádku nebo za příkaz javaws.exe stačí vložit následující URL:

http://<Tivoli Enterprise Portal Server>:15200/tep.jnlp

Spustí se klientská aplikace. Správce je vyzván k zadání uživatelských údajů. Po přihlášení je možné monitorovat jednotlivé systémy a agenty.



Obrázek 13: Zobrazení klientské aplikace Tivoli Enterprise Portal – vlastní zpracování

V okně klienta je vidět několik odlišných pohledů. Dohromady tvoří pracovní plochu. Správce může vytvářet a přepínat mezi jednotlivými pracovními plochami. Hlavním prvkem je navigátor. Zde se zobrazují různé pohledy na monitorované systémy. Uživatel se setká s dvěma základními pohledy: fyzickým a logickým. Fyzický pohled zobrazuje všechny aktuálně monitorované systémy. Fyzický pohled nelze upravovat. Mění se pouze, když je nainstalován nebo odinstalován monitorovací agent. Logické pohledy zobrazují vybrané systémy. Správce si může vytvářet logické pohledy podle svých potřeb. Každý prvek v navigátoru zobrazuje jeden monitorovaný systém. Ve stromovém pohledu jsou zobrazeni jednotliví agenti a jejich skupiny atributů. Zvolením konkrétní skupiny atributů se zobrazí pohled na relevantní data ke konkrétní skupině. Způsob zobrazení dat může být změněn v nástrojové liště. Nasbíraná data mohou být zobrazena jako graf nebo tabulka. Data jsou zdrojem pro tvorbu situací. Situace je podmínka, která určuje, zda se má generovat událost. Prostředí IBM Tivoli Monitoring rozlišuje dva druhy události: vzorkované a ryzí. Vzorkovaná událost nastane vždy, když je prováděno testování a je splněna podmínka situace. Vzorkovaná situace má určený interval, kdy se má podmínka testovat. Vyřešením problému se vzorkovaná událost nemusí ručně zavírat. Při dalším testování podmínky se sama uzavře. Příkladem vzorkované události může být překročení hranice využívané paměti RAM. Po snížení využívaného množství paměti RAM se událost uzavře.

Pokud nastane ryzí událost, je nutné, aby došlo k ručnímu uzavření. Správce vyřeší daný problém a v kontextové nabídce ji uzavře. Příkladem může být přidání nového záznamu v monitorovaném logu. Vyhodnocování situace probíhá automaticky a není potřeba nastavit vzorkovací frekvenci.

Podstatnou částí je zobrazení Situation Event Console. Zde se zobrazují události, které nastaly na monitorovaných systémech podnikového prostředí. Každý řádek tabulky vyjadřuje jednu událost. Události mají závažnost, stav, vlastníka, název, zdroj, časovou známku, stáří, typ, identifikátor a další. Pokud je přihlášeno více uživatelů, je vidět, kdo řeší jednotlivé události. Jednotliví správci mohou vidět své potvrzené události v seznamu My Acknowledged Events [18] [19].

4 Vlastní práce

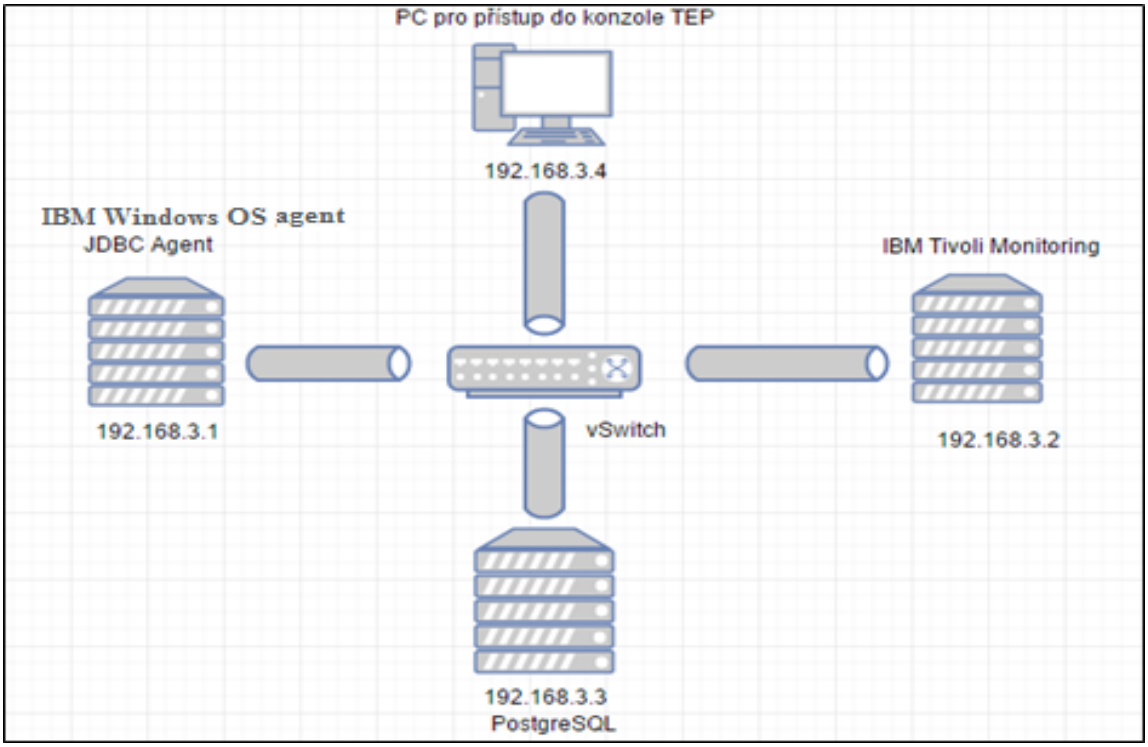
Praktická část se zabývá aplikací teoretických poznatků. Shrnuje samotné vytvoření ukázkového agenta v prostředí IBM Tivoli Monitoring a nasazení v testovacím prostředí.

Pro ukázku je vytvořen agent vycházející z potřeby monitorování vybraných řádků z databáze PostgreSQL. Nalezené řádky následně generují ryzí události. Cílovou platformou agenta je Microsoft Windows.

4.1 Charakteristika testovacího prostředí

Pro praktickou část je vytvořeno virtualizované prostředí založené na platformě Microsoft Hyper-V. Jednotlivé virtuální servery jsou propojeny pomocí interního virtuálního přepínače. Databáze PostgreSQL je spuštěna na serveru s využitím kontejnerové virtualizace Docker for Windows.

Server, hostující centrální komponenty IBM Tivoli Monitoring, má přidělenou statickou IP adresu 192.168.3.2. Testovací databáze PostgreSQL se nachází na serveru s IP adresou 192.168.3.3. Vytvořený agent je nainstalován na serveru s IP adresou 192.168.3.1 a pro přístup do konzole TEP je použita stanice s IP adresou 192.168.3.4.

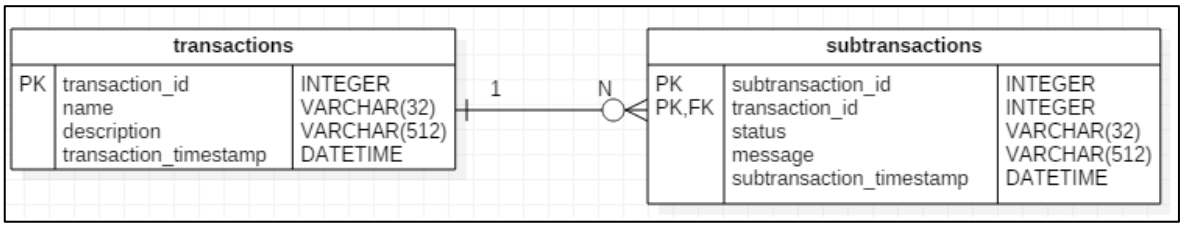


Obrázek 14: Schéma virtuální sítě a rozmístění jednotlivých prvků – vlastní zpracování

4.2 Specifikace zadání

V databázi crm_database jsou vytvořeny tabulky transactions a subtransactions. Každá transakce má N subtransakcí a každá subtransakce patří právě jedné transakci.
Cílem je zjištění, které subtransakce selhaly v rámci celé transakce, a následné informování uživatele. Selhání subtransakce lze identifikovat pomocí sloupce status. Pokud se ve sloupci status nachází řetězec ERROR, je transakce považována za chybovou. SQL dotaz, který vrátí řádky s chybovým řetězcem, je následující:

```
SELECT
   tr.transaction_id, name, description, transaction_timestamp, subtransaction_id, status,
   message, subtransaction_timestamp
FROM transactions AS tr JOIN subtransactions AS sub ON
   (tr.transaction_id=sub.transaction_id)
WHERE sub.status LIKE 'ERROR'
```



Obrázek 15: ER diagram znázorňující vztah mezi transakcemi a subtransakcemi – vlastní zpracování

4.3 Analýza zadání

IBM Agent Builder umožňuje vytvoření agenta využívající JDBC pro komunikaci s databázemi. Avšak standardní JDBC agent podporuje pouze databáze Microsoft SQL Server, IBM DB2 a ORACLE. Agent produkuje pouze vzorkované události [14].

Z výše uvedeného je patrné, že je potřeba vytvořit agenta, který využívá výstupu z více datových zdrojů. Vytvořený agent se skládá ze dvou částí. První část využívá datového zdroje – Output from a script. Druhá část využívá datového zdroje – A Log File.

Celý proces běhu probíhá následovně. Nejprve je zavolán program, který zapíše požadované řádky z databáze do souboru a následně vypíše diagnostické informace o svém běhu na standardní výstup. Jednotlivé sloupce jsou odděleny nakonfigurovaným

oddělovačem. Zapsaná data v souboru jsou zaznamenána monitorovacím systémem, a pokud splní situační podmínku, vygenerují ryzí událost. Diagnostická data o běhu programu se také zobrazí v monitorovacím systému a chybovému stavu se vytvoří vzorkovaná událost.

4.4 Vytvoření agenta

Pro vytvoření programu je využit programovací jazyk Java a rozhraní JDBC API. Komunikace s databází probíhá pomocí tzv. JDBC ovladače, který je nutné stáhnout z webových stránek výrobce databáze. Výkonný program je vytvořen v integrovaném vývojovém prostředí Eclipse a vyexportován jako JAR balíček. Spuštění programu probíhá pomocí dávkového skriptu s extenzí BAT. Dávkový skript zajišťuje, že je vyexportovaný JAR balíček spuštěn se správně nakonfigurovanou verzí IBM Java 1.7 a příslušným ovladačem pro JDBC rozhraní. Cesta k IBM Java 1.7 je uložena v proměnné %IBM_JAVA_17% a umístění JDBC ovladače se nachází v proměnné %DRIVER%. Proměnná %DRIVER% je součástí class path při spuštění Java aplikace.

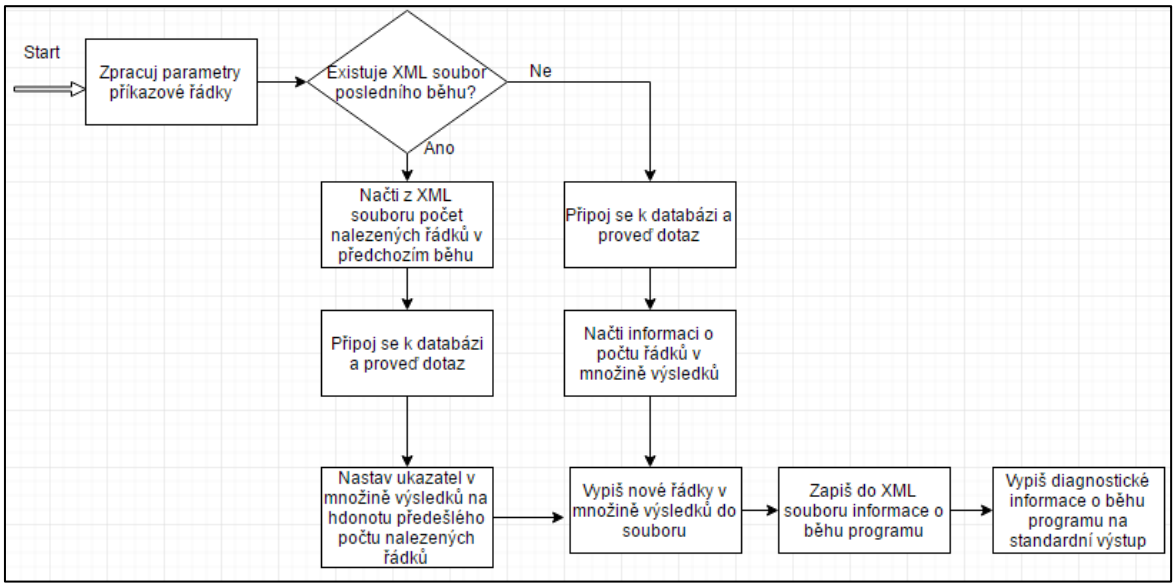
Rozlišení souběžně běžících instancí stejného agenta na jednom serveru je provedeno předáním názvu instance v proměnné %CTIRA_SUBSYSTEM_ID%. Prvotní konfigurace je provedena při prvním startu agenta.

Program zapisuje následující informace do XML souboru: počet nalezených řádků, výsledek programu, počáteční a koncový čas běhu. Spouštěcí skript, program agenta, XML výstup a soubor přesměrovaného chybového výstupu jsou uloženy ve složce %CANDLE_HOME%\TMAITM6\scripts na stanici, kde je agent nainstalován.

Na standardní výstup jsou vypsány sloupce oddělené nakonfigurovaným oddělovačem:

- TimeStmp - Tivoli Time Stamp
- ScriptResult – String: Maximum size 7 (SUCCESS/ERROR)
- Message - String: Maximum size 7
- TotalNumberOfRows - 32 bit number
- PreviousNumberOfRows - 32 bit number
- Delta - 32 bit number

Následující diagram zobrazuje zjednodušenou podobu algoritmu pro vytvořený program. Pro přehledné zobrazení nejsou v diagramu zakresleny chybové stavy, avšak v programu jsou vyřešeny pomocí výjimek. Pokud dojde k výjimce v jakémkoliv kroku algoritmu, je vypsán chybový stav a podrobná informační zpráva na standardní výstup. Monitorovací systém zaznamená chybovou zprávu a vytvoří vzorkovanou událost.



Obrázek 16: Zjednodušený diagram algoritmu JDBC agenta - vlastní zpracování

Parametr	Význam
-user	Uživatelské jméno
-password	Uživatelské heslo
-db-type	Typ databáze (MSSQL|ORACLE|DB2|POSTGRES)
-db-hostname	Název hostitele databáze
-db-name	Název databáze
-db-port	Port databáze
-delimiter	Oddělovač sloupců na výstupu
-query	SQL dotaz
-jdbc-connection-string	Vlastní JDBC řetězec
-ignore-existing	Ignorování existujících řádků při prvním spuštění. Výchozí - true
-listener	Název databáze je listener - ORACLE
-instance	Název instance agenta

Tabulka 1: Parametry JDBC agenta

Program zpracovává několik parametrů, které jsou součástí spouštěcí řádky. Agent umožňuje navíc flexibilitu v tom, že lze nastavit komunikace s databázemi Microsoft SQL

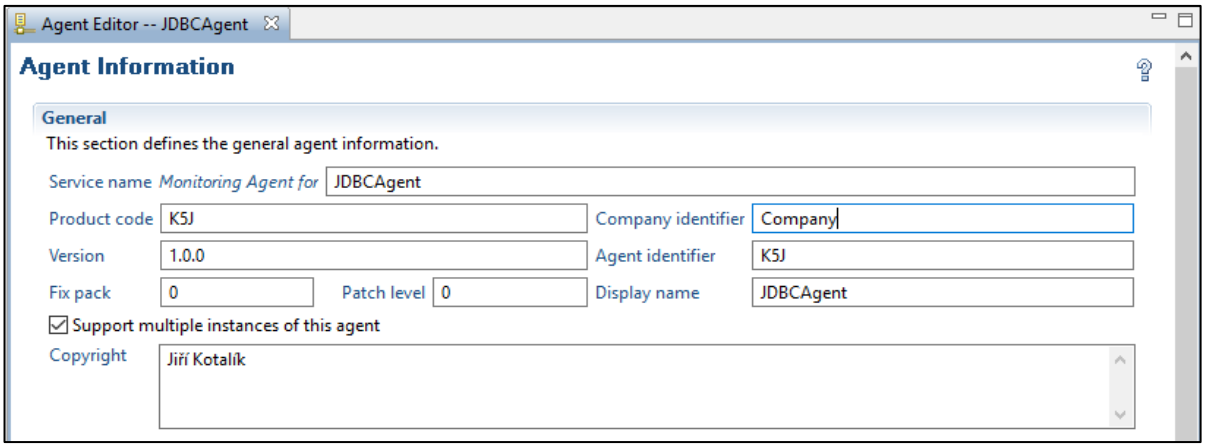
Server, IBM DB2, ORACLE nebo použít vlastní JDBC řetězec a příslušné ovladače. Za jednotlivými parametry jsou názvy proměnných, kde se nachází již reálné hodnoty parametrů. Při zpracování parametrů dojde k extrakci hodnot z předaných proměnných prostředí.

Obsah dávkového souboru JDBCAgent.bat, který spouští jar balíček JDBCAgent.jar s potřebnými parametry:

```
@echo off
"%IBM_JAVA_17%" -Dname="JDBCAgent-%CTIRA_SUBSYSTEM_ID%"
-cp "%CANDLE_HOME%/TMAITM6/scripts/JDBCAgent.jar;%DRIVER%"
cz.kotalik.jiri.jdbc.agent.JDBCAgent -user USER_NAME -password PASSWORD
-db-type DATABASE_TYPE -db-hostname DATABASE_HOSTNAME
-db-name DATABASE_NAME -db-port DATABASE_PORT
-delimiter JDBC_AGENT_DELIMITER
-query JDBC_AGENT_QUERY
-jdbc-connection-string JDBC_STRING
-ignore-existing IGNORE_EXISTING_ROWS -listener DATABASE_LISTENER
-instance CTIRA_SUBSYSTEM_ID
2>scripts/JDBCAgent-%CTIRA_SUBSYSTEM_ID%.err
```

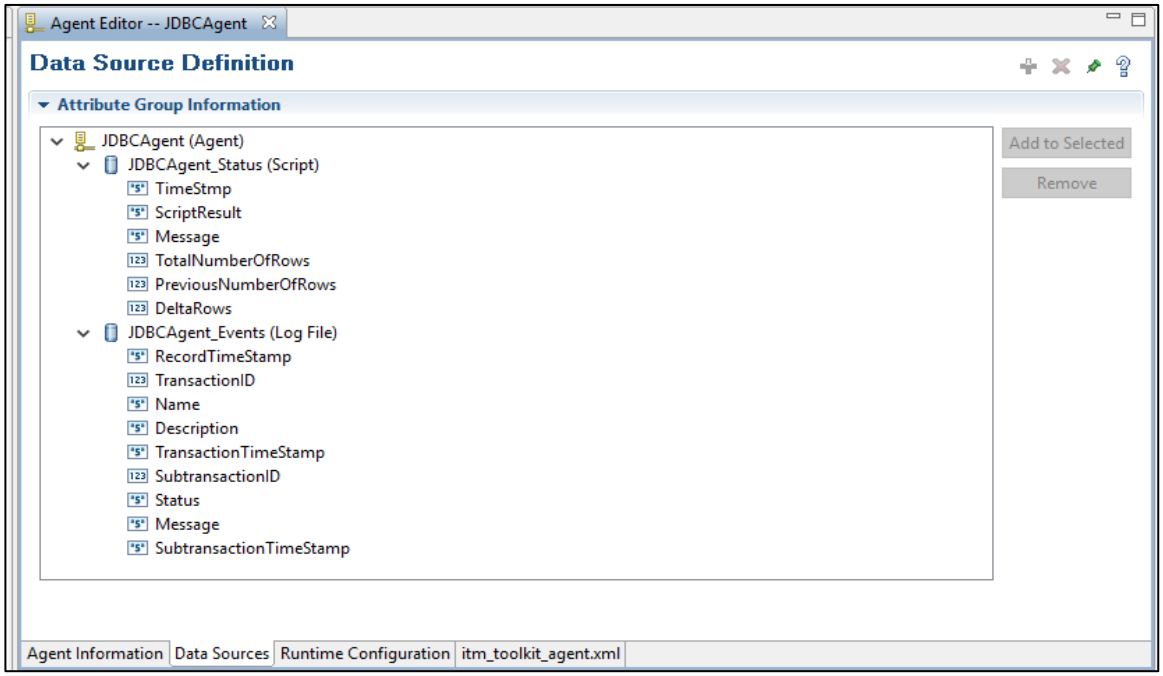
Vytvoření a zabalení monitorovacího agenta je provedeno pomocí vývojového prostředí IBM Agent Builder.

V menu je zvolena možnost vytvoření nového agenta. Otevře se průvodce, kde jsou vyplněny základní informace. Agent je pojmenován JDBCAgent a má přidělen kód K5J. Zaškrtnutím políčka Support multiple instances of this agent je možné spustit více instancí stejného agenta na jedné stanici.



Obrázek 17: Přehled vyplněných informací - vlastní zpracování

Ze zadání jsou patrné názvy jednotlivých sloupců a příslušné datové typy, které budou generovány jednotlivými zdroji. Oddělovač sloupců bude nastaven na řetězec <;>. Základem je použití dvou datových zdrojů. Prvním zdrojem je výstup ze skriptu označený JDBCAgent_Status. Druhým zdrojem je log soubor pojmenovaný JDBCAgent_Events.

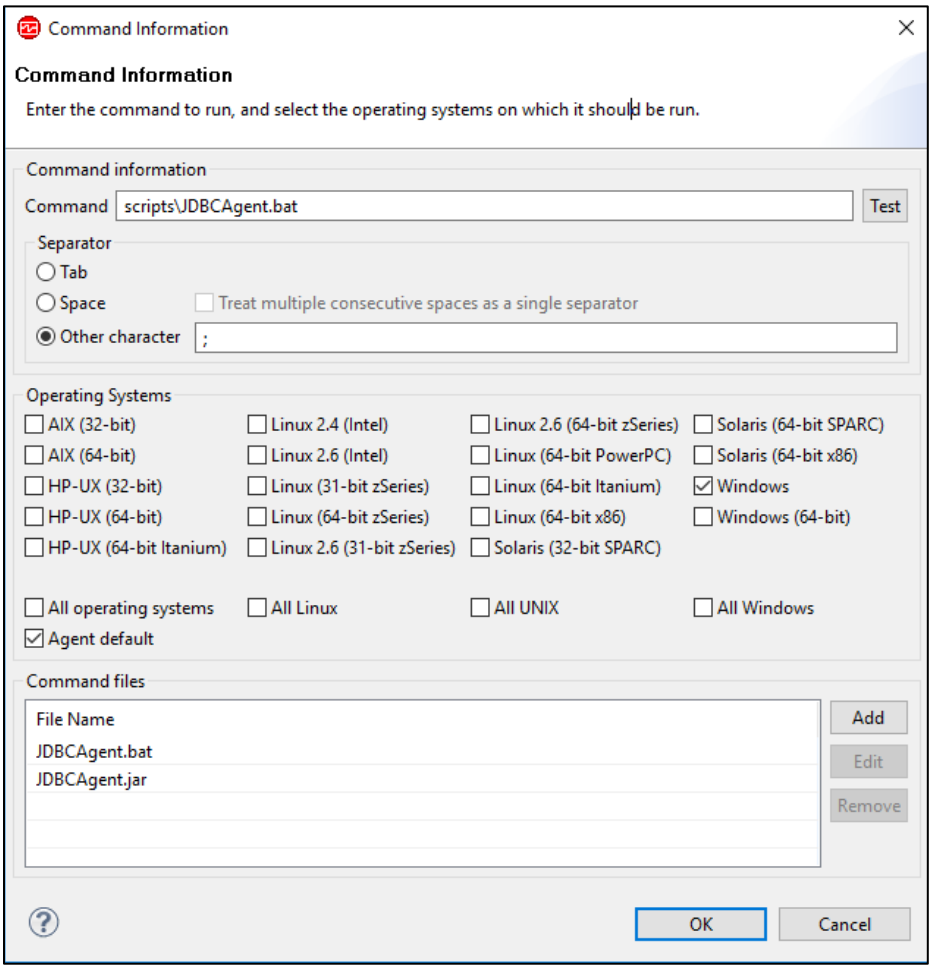


Obrázek 18: Zdroje monitorování pro JDBC agenta v prostředí IBM Agent Builder – vlastní zpracování

Datový zdroj, výstup ze skriptu, je nastaven následovně. Vždy je vrácena pouze jedna řádka. Cílová platforma je nastavena na Windows. V položce Command files jsou přidány

soubory JDBCAgent.jar a JDBCAgent.bat. Poslední položkou je příkaz, který se má vždy vykonat, když je agent zavolán. Příkaz je následující: *scripts\JDBCAgent.bat*. Při každém zavolání agenta bude spuštěn skript JDBCAgent.bat, který je umístěn ve složce scripts.

Položka separátor je ignorována, protože jednotlivé oddělovače jsou nastaveny při definici sloupců.

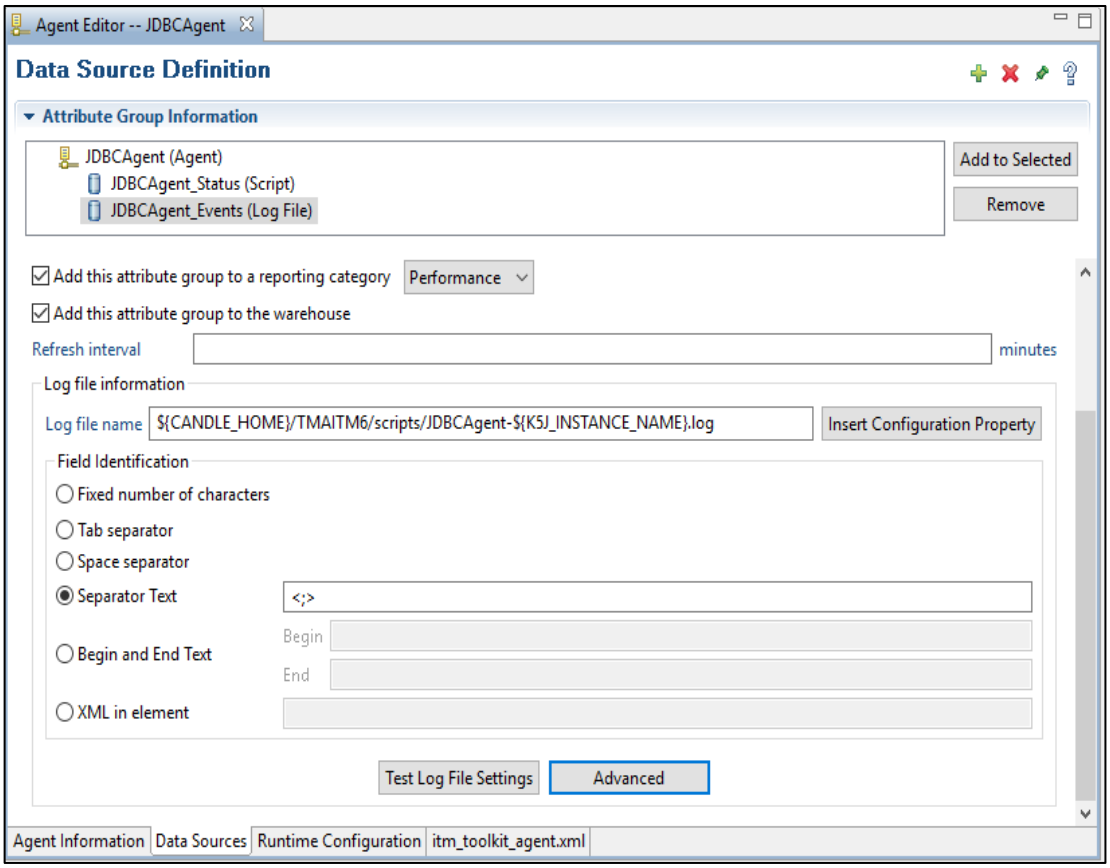


Obrázek 19: Nastavení zdroje JDBCAgent_Status – vlastní zpracování

Nyní, když je nastaven datový zdroj výstup ze skriptu, je možné nakonfigurovat datový zdroj log soubor.

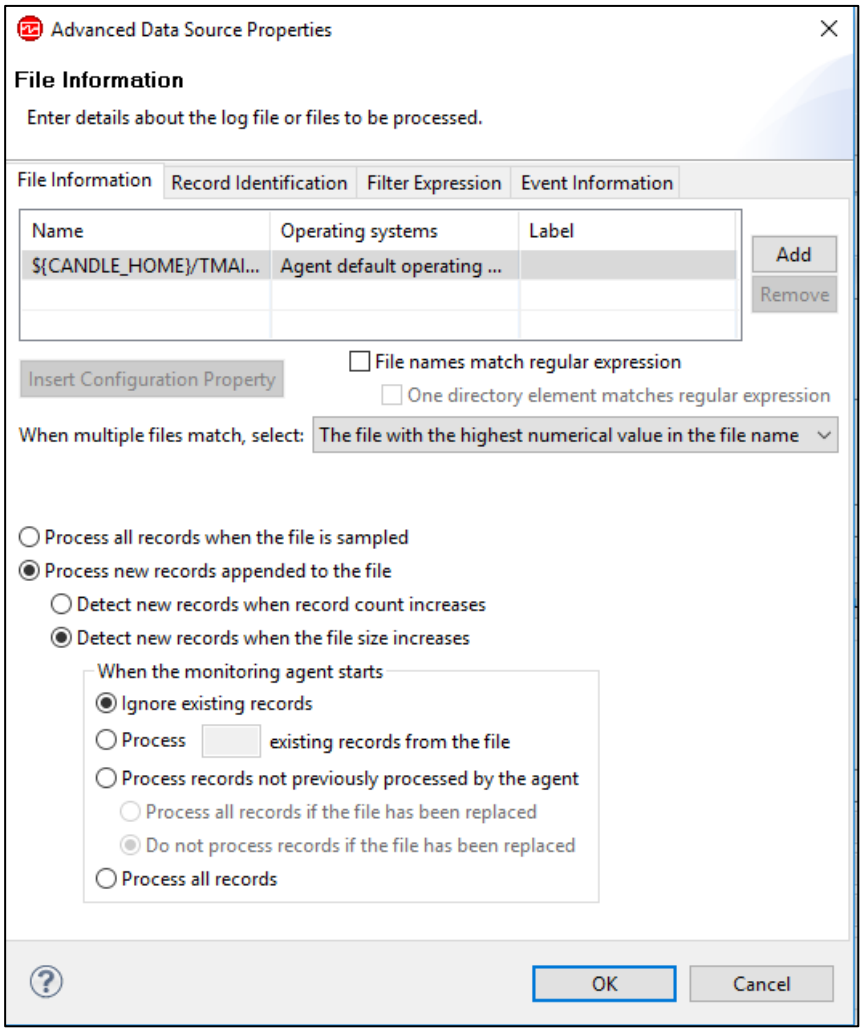
Cesta k log souboru je následující:
${CANDLE_HOME}/TMAITM6/scripts/JDBCAgent-${K5J_INSTANCE_NAME}.log

Značení ${název proměnné}vrací nastavenou hodnotu proměnné prostředí. Uvnitř proměnné ${CANDLE_HOME} je uložena cesta, kde se nachází instalace komponent IBM Tivoli Monitoring. V proměnné *${K5J_INSTANCE_NAME}* je uložen název instance nakonfigurovaného agenta. Jako separátor jednotlivých sloupců v řádku je opět řetězec <;>.



Obrázek 20: Nastavení zdroje JDBCAgent_Events – vlastní zpracování

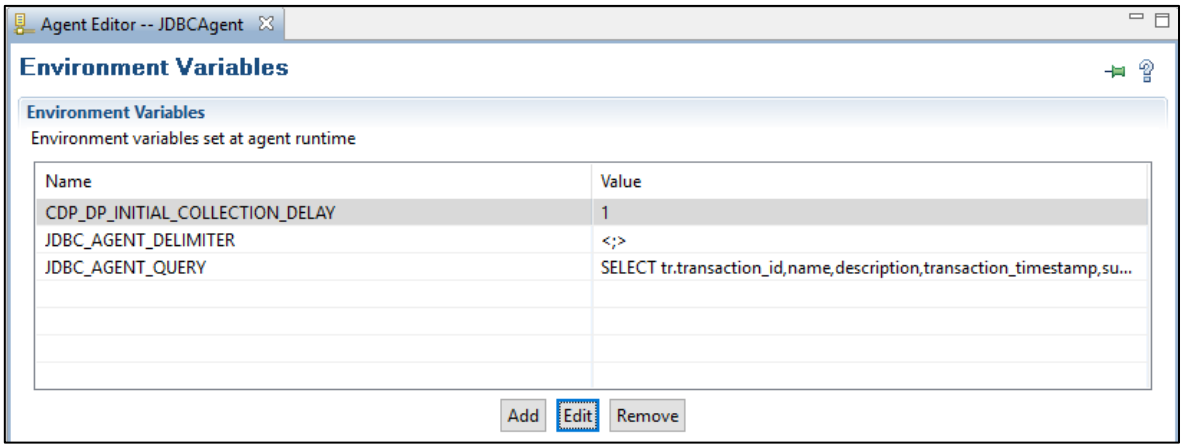
Pro rozšířenou konfiguraci je nutné kliknout na tlačítko Advanced. Otevře se okno s podrobnější konfigurací. Aby došlo ke generování pouze ryzích událostí, je nutné nastavit položku Process new records appended to the file. V monitorovacím systému se tedy zobrazí pouze nově přidané řádky v log souboru. Detekce nových záznamů je založena na sledování velikosti log souboru. Je nutné zvolit položku Detect new records when the file size increases. Poslední položkou je Ignore existing records, která říká, že se při startu ignorují již existující položky.



Obrázek 21: Rozšířené nastavení zdroje JDBCAgent_Events – vlastní zpracování

Pokud jsou všechny datové zdroje vytvořeny a nakonfigurovány, lze přistoupit k přípravě běhového prostředí. Nastavením proměnných prostředí je možné měnit chování samotného agenta. Definování vlastních proměnných je provedeno v sekci Environment Variables.

Standardně je připravena proměnná CDP_DP_INITIAL_COLLECTION_DELAY, jejíž hodnota je nastavena na 1. Hodnota udává, o kolik sekund se má při startu agenta zpozdit první sběr dat. Přidáním vlastní proměnné JDBC_AGENT_DELIMITER bude agent používat nastavenou hodnotu jako oddělovač sloupců při zápisu do log souboru a na standardní výstup. Hodnota oddělovače je nastavena na řetězec <;>. SQL dotaz je uložen v proměnné JDBC_AGENT_QUERY.



Obrázek 22: Nastavené proměnné prostředí – vlastní zpracování

Před prvním spuštěním agenta je dobré uživateli nabídnout možnost konfigurace běhového prostředí. Pro zobrazení formuláře s konfiguračními parametry je nutné připravit jednotlivé obrazovky průvodce, které spolu souvisí. Příprava průvodce je provedena v části Runtime Configuration Information.

Základní obrazovka průvodce zobrazí uživateli vstupní pole typu choice, kde je zvolen typ databáze. Na výběr jsou možnosti:

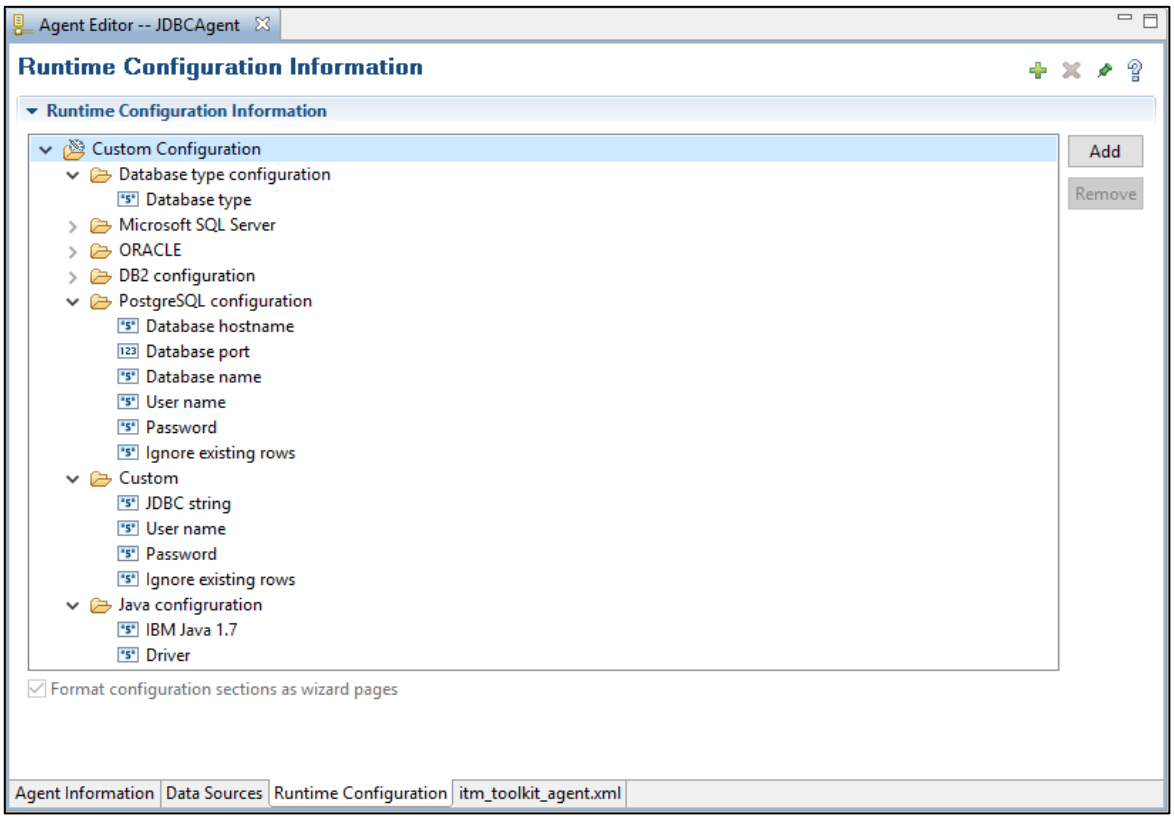
- Microsoft SQL Server – MSSQL
- ORACLE – ORACLE
- IBM DB2 – DB2
- PostgreSQL – POSTGRES
- Custom – CUSTOM

Výběrem typu databáze je uživateli zobrazena další obrazovka nastavení související s konfigurací konkrétního typu databáze.

Příkladem může být nastavení databáze PostgreSQL. Uživateli jsou zobrazena následující pole:

- Database hostname – hostitelské jméno/IP adresa databázového hostitele
- Database port – port, na kterém naslouchá databázový systém
- User name – uživatelské jméno pro přístup do databáze
- Password – uživatelské heslo pro přístup do databáze
- Ignore existing rows – výchozí je logická hodnota true

Každé vstupní pole je reprezentováno jménem proměnné prostředí. Jednotlivá vstupní pole mají definovaný datový typ a jsou označena jako povinná. Například pole Database hostname je reprezentováno proměnnou DATABASE_HOSTNAME a je typu String. Pole reprezentující heslo je typu Password. Vepisované heslo je automaticky zobrazeno zástupnými znaky. Poslední obrazovku tvoří nastavení Java prostředí a JDBC ovladače. Vstupní pole IBM Java 1.7 a Driver jsou typu File Browser. Obě pole vyvolají dialogové okno pro výběr souboru. Vyplněné hodnoty polí jsou uloženy do příslušných proměnných prostředí a předány skriptu JDBCAgent.bat.



Obrázek 23: Příprava průvodce nastavení agenta - vlastní zpracování

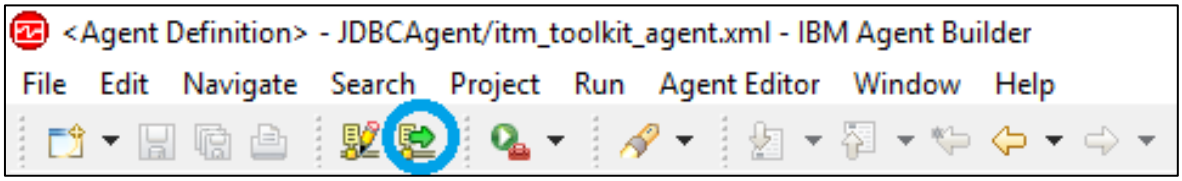
Jednotlivé obrazovky je nutné propojit tak, aby měly mezi sebou návaznost v závislosti na volbě uživatele. Provázání je provedeno definováním podmínek ve vygenerovaném XML souboru, který reprezentuje konfigurační sekce průvodce. Všechny podmínky jsou potomky uzlu *<AgentConfig:agent>*. Podmínka obsahuje jednoznačný identifikátor a tělo podmínky je tvořeno definicí vlastní podmínky. Položka *name* obsahuje proměnnou, která se má porovnávat s hodnotou uloženou v položce *value*. Syntaxe podmínky je následující:

<condition ID="POSTGRES">
<condition-item name="$(DATABASE_TYPE)" value="POSTGRES"/>
</condition>

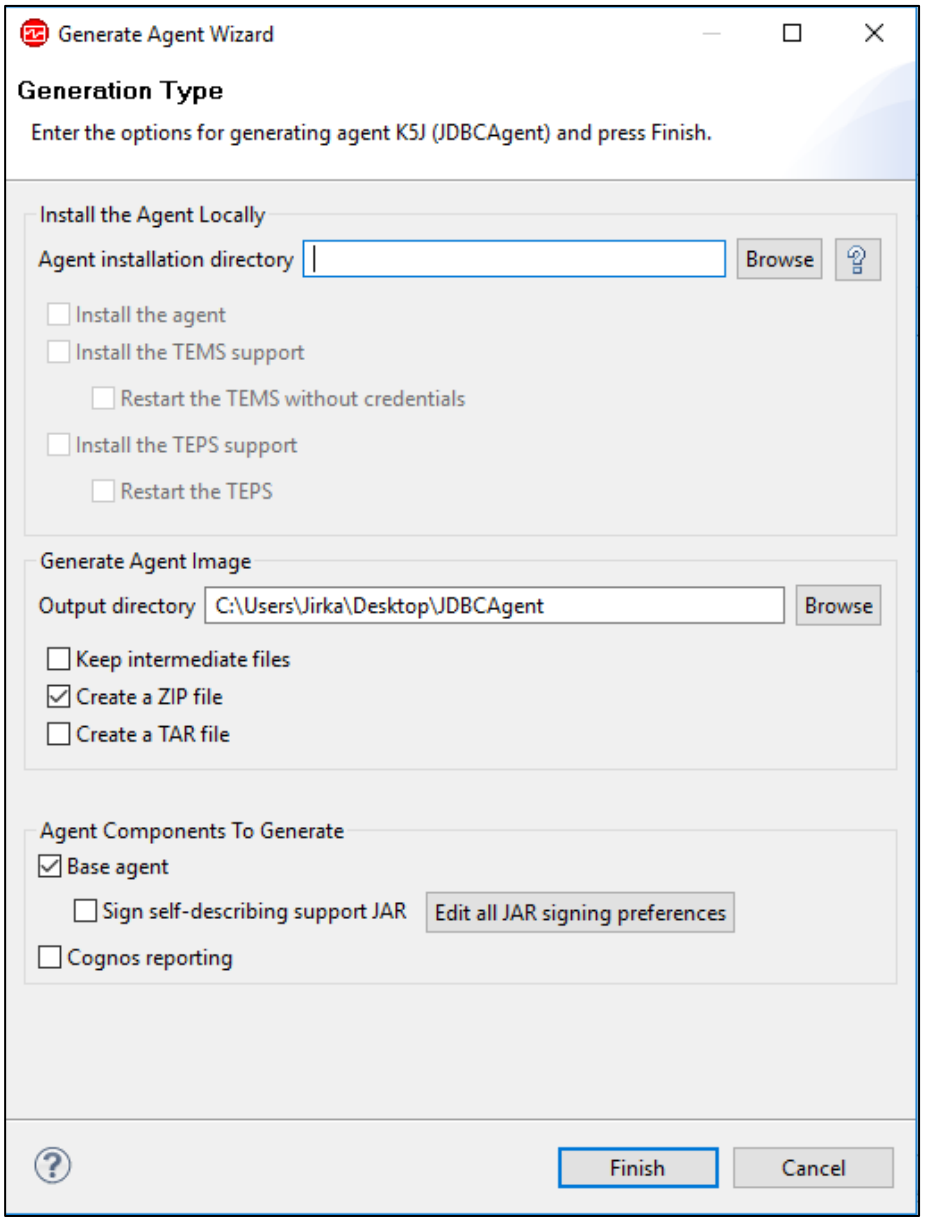
Proměnná DATABASE_TYPE byla přiřazena vstupnímu poli typu Choice. Uživatel si vybere požadovanou databázi a dle výběru je přesměrován na požadovanou konfigurační sekci.

Přidáním podmínky do konfigurační sekce se zobrazí pouze ta sekce, splňující výše definovanou podmínku. Syntaxe sekce s využitím podmínky je následující:
<section condition="POSTGRES" name="POSTGRESQL_CONFIGURATION">

Nyní je agent připraven k exportování do instalačního balíku. Použitím nabídky Agent Editor -> Generate Agent je otevřeno dialogové okno pro uložení instalačního balíku. Agent je vyexportován v podobě ZIP archivu. Pro export může být použito i připravené tlačítko v nástrojové liště.



Obrázek 24: Zobrazení nabídky pro vygenerování instalačního balíčku – vlastní zpracování



Obrázek 25: Dialogové okno exportu agenta – vlastní zpracování

Jelikož má agent nastavenou cestu spuštění programu na *scripts\JDBCAgent.bat*, je nutné upravit vygenerovaný instalační skript. Ve vyexportovaném archivu se nachází skript *installIraAgent.bat*. Zde je nutné přidat řádku:

if not exist "%TMA_DIR%\scripts" mkdir "%TMA_DIR%\scripts"

Nyní je zaručeno, že dojde při instalaci k vytvoření složky scripts ve složce reprezentované proměnnou prostředí TMA_DIR.

Upravením části skriptu, kde jsou kopírovány soubory *JDBCAgent.jar* a *JDBCAgent.bat,* je docíleno toho, že zmíněné soubory se zkopírují do složky *%TMA_DIR%\scripts.*

Upravená část instalačního skriptu je následující:

```
if "winnt" == "%WIN_ARCH%" (
@call :copyFile "%SOURCE_DIR%ira\agent\all_windows\scripts\JDBCAgent.jar"
"%TMA_DIR%\scripts\JDBCAgent.jar"   &@if errorlevel 1 goto end
 @call :copyFile "%SOURCE_DIR%ira\agent\all_windows\scripts\JDBCAgent.bat"
"%TMA_DIR%\scripts\JDBCAgent.bat"   &@if errorlevel 1 goto end )
```

Protože byla změněna standardní instalační cesta, je nutné upravit skript pro odinstalování agenta. Archiv obsahuje ve složce *ira\agent\winnt* soubor *k5j_uninstall.vbs.* Ve skriptu jsou změněny následující řádky ve funkci *delAgent*:

```
rc = rc + shObj.run("%comspec% /c del /f /q " + candleHome +
"\TMAITM6\scripts\JDBCAgent.jar >>" & candleHome & "\logs\k" & pcode &
"_uninstall.log 2>&1", 0, true)
rc = rc + shObj.run("%comspec% /c del /f /q " + candleHome +
"\TMAITM6\scripts\JDBCAgent.bat >>" & candleHome & "\logs\k" & pcode &
"_uninstall.log 2>&1", 0, true)
```

Při spuštění skriptu pro odinstalování agenta dojde ke správnému odstranění všech nainstalovaných souborů. Nyní je možné přejít k instalaci agenta. Upravené soubory jsou uloženy zpět do archivu pro snadnou distribuci.

4.5 Instalace a konfigurace agenta

Každý agent musí mít nainstalovánu podporu pro Tivoli Enterprise Monitoring Server a Tivoli Enterprise Portal Server. Nejprve je rozbalen archiv do vybrané složky.

V hlavní složce se nachází instalační soubory *installIraAgentTEMS.sh* a *installIraAgentTEPS.sh*. Jelikož jsou komponenty TEMS a TEPS nainstalovány na stejném serveru, je možné spustit oba instalační skripty v terminálu. Instalace musí probíhat v kontextu uživatele root.

Instalační příkazy jsou následující:

./installIraAgentTEMS.sh /opt/IBM/ITM

./installIraAgentTEPS.sh -r /opt/IBM/ITM

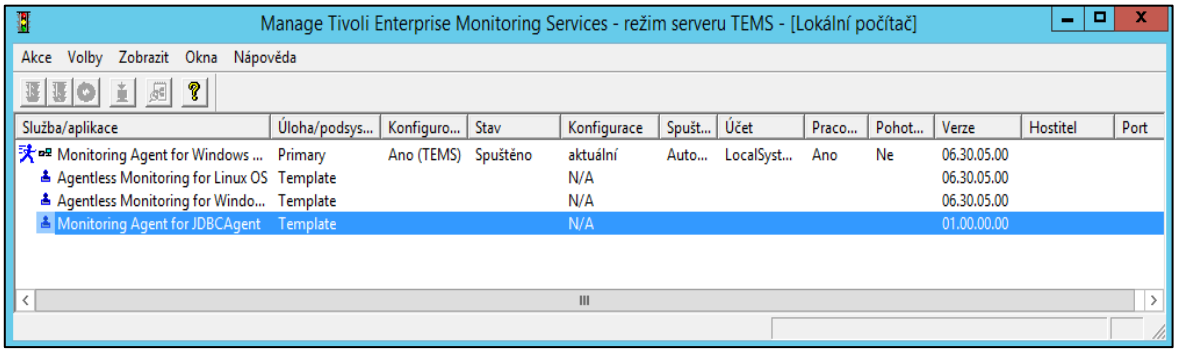
Standardně dojde k restartování služeb TEMS a TEPS. Pro ověření správné instalace je použit příkaz *cinfo*, který zobrazí nainstalované podpory.

Instalace agenta je provedena na serveru, kde je nainstalován IBM Windows OS Agent. Archiv obsahuje upravený instalační skript agenta.

Instalace spočívá ve spuštění následujícího příkazu v kontextu lokálního administrátora:

.\installIraAgent.bat C:\IBM\ITM

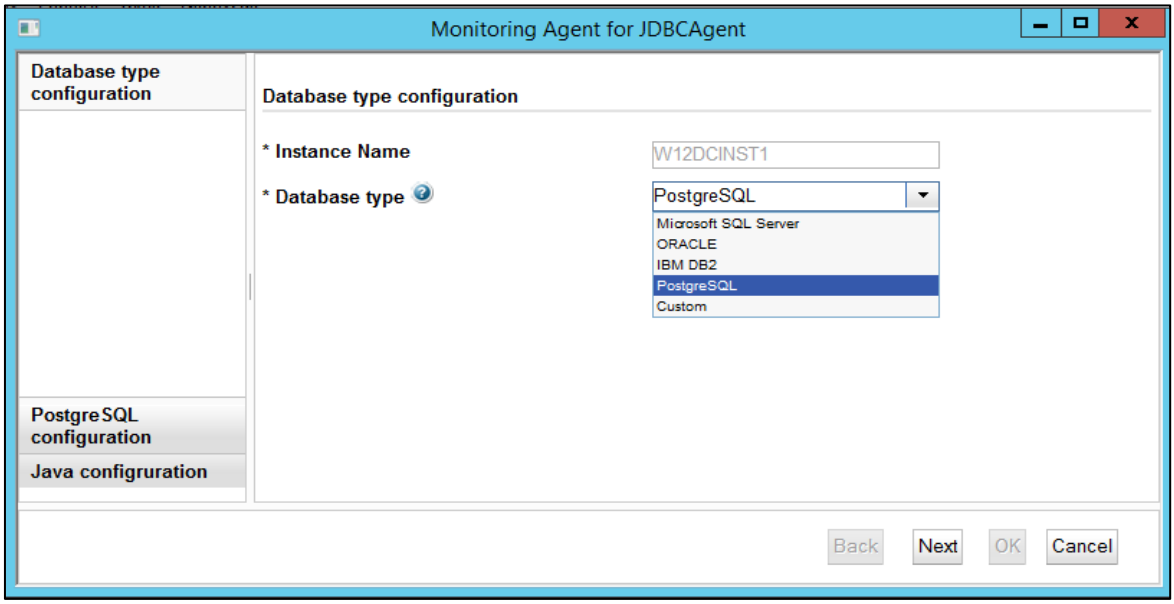
Spuštěním nástroje Manage Tivoli Enterprise Monitoring Services je zobrazena tabulka s nainstalovanými agenty. Z obrázku níže je patrné, že vytvořený JDBCAgent se správně nainstaloval a je připraven ke konfiguraci.



Obrázek 26: Manage Tivoli Enterprise Monitoring Services – vlastní zobrazení

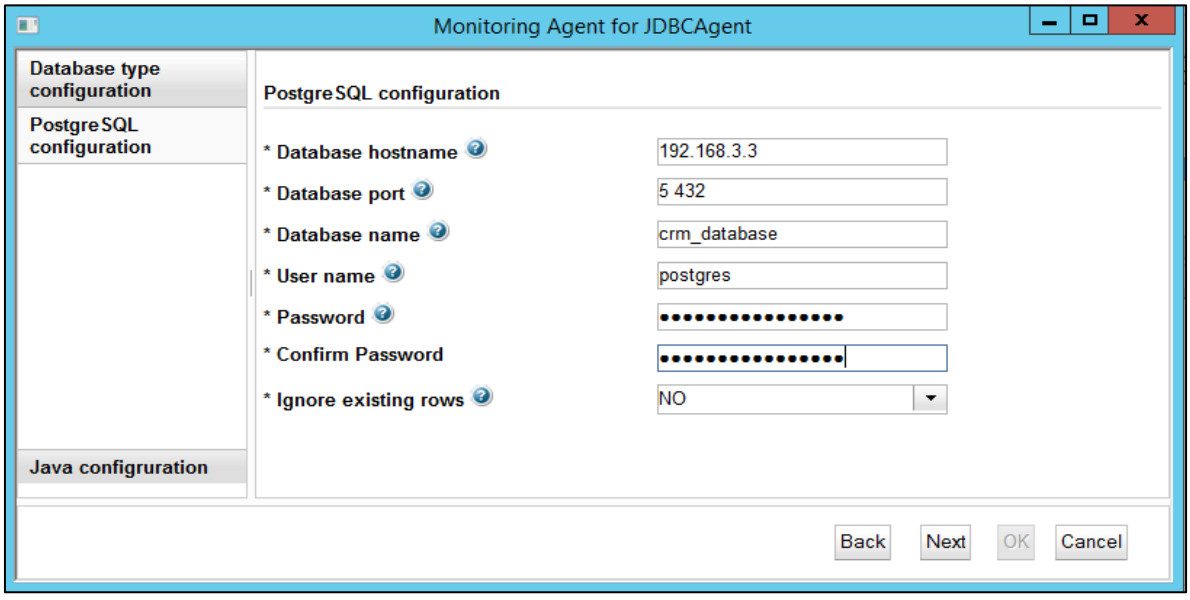
Stiskem pravého tlačítko na vyznačené položce JDBCAgent a vybrání položky Konfigurovat pomocí předvoleb je spuštěn průvodce nastavením.

Pro jednoznačnou identifikaci je vyplněn název instance agenta. Následuje výběr požadované databáze. Potvrzením je zobrazena související obrazovka s nastavením konkrétní databáze.



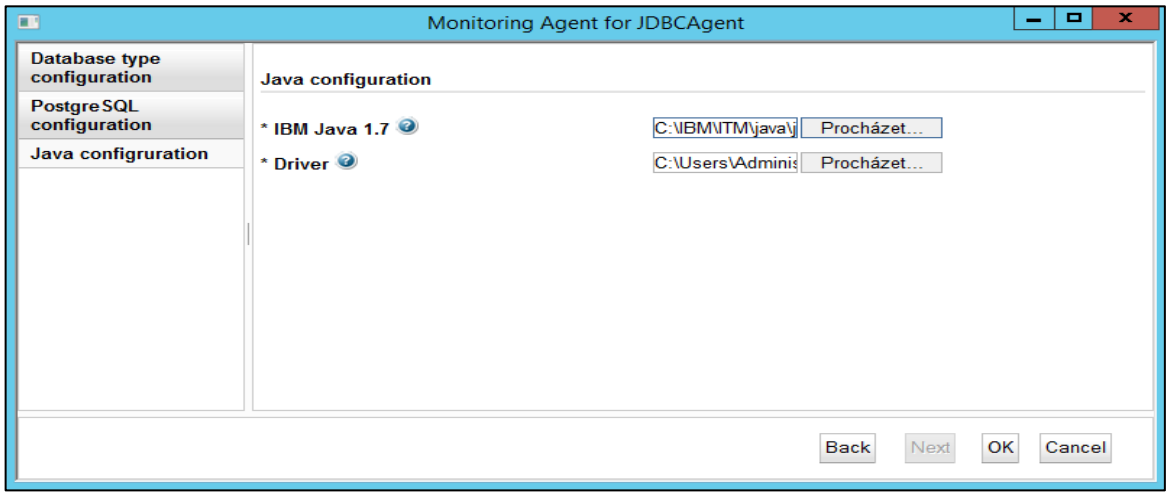
Obrázek 27: Výběr databáze v průvodci – vlastní zpracování

Uživatel je dotázán na název hostitele databáze, port, název databáze, uživatelské jméno, heslo a možnost zvolit ignorování existujících řádků.



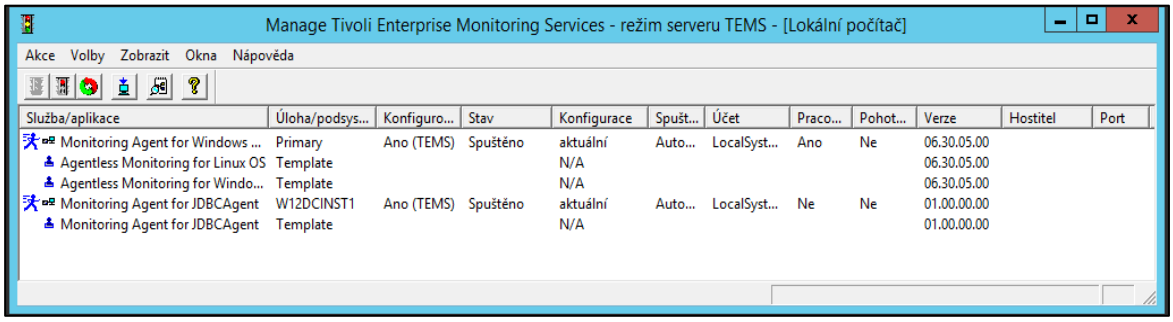
Obrázek 28: Nastavení databáze PostgreSQL – vlastní zpracování

Stiskem tlačítka Next je uživatel přesměrován na obrazovku se specifikací běhového prostředí IBM Java 1.7 a JDBC ovladače pro příslušnou databázi.



Obrázek 29: Nastavení běhového prostředí IBM Java 1.7 - vlastní zpracování

Potvrzením je konfigurace ukončena a lze přejít ke spuštění. V tabulce se nachází nově vytvořená položka s konfigurací. Pravým tlačítkem je zobrazena nabídka s příkazem Spustit. Po spuštění se objeví v levé části řádku modrá běžící postava.



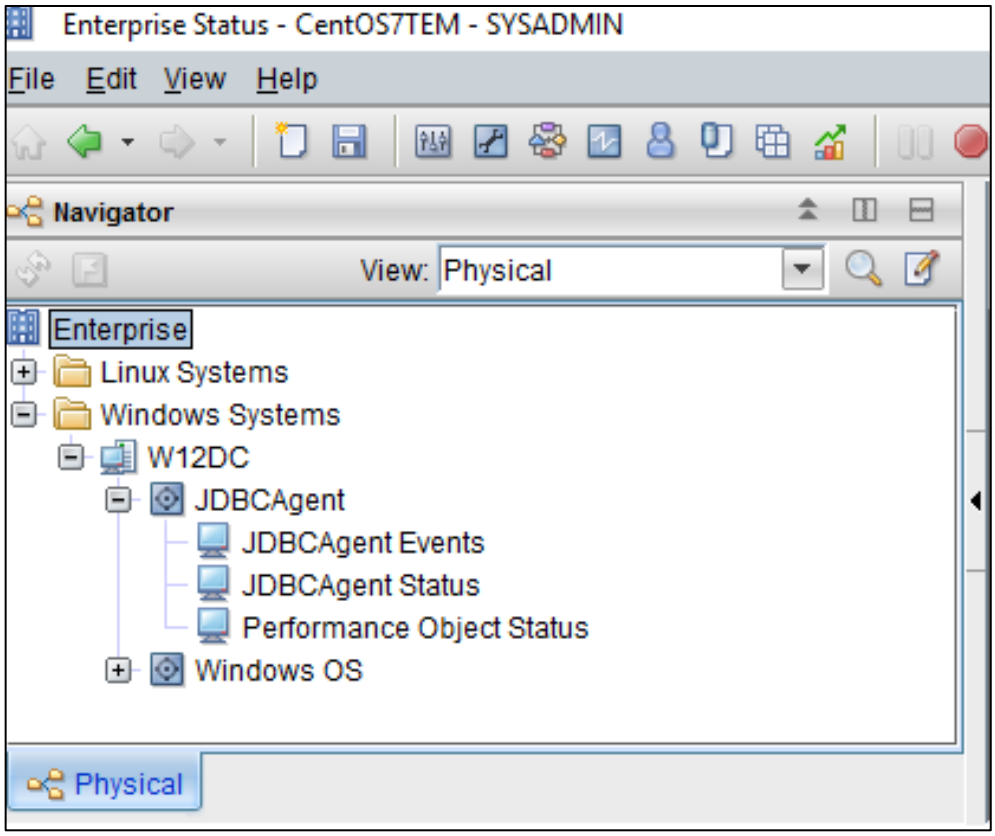
Obrázek 30: Běžící JDBCAgent - vlastní zpracování

4.6 Vytvoření situací v Tivoli Enterprise Portal

Pro spuštění Tivoli Enterprise Portal je vhodné použít IBM Java verze 1.7. Portál lze spustit pomocí prohlížeče Internet Explorer nebo pomocí příkazu:

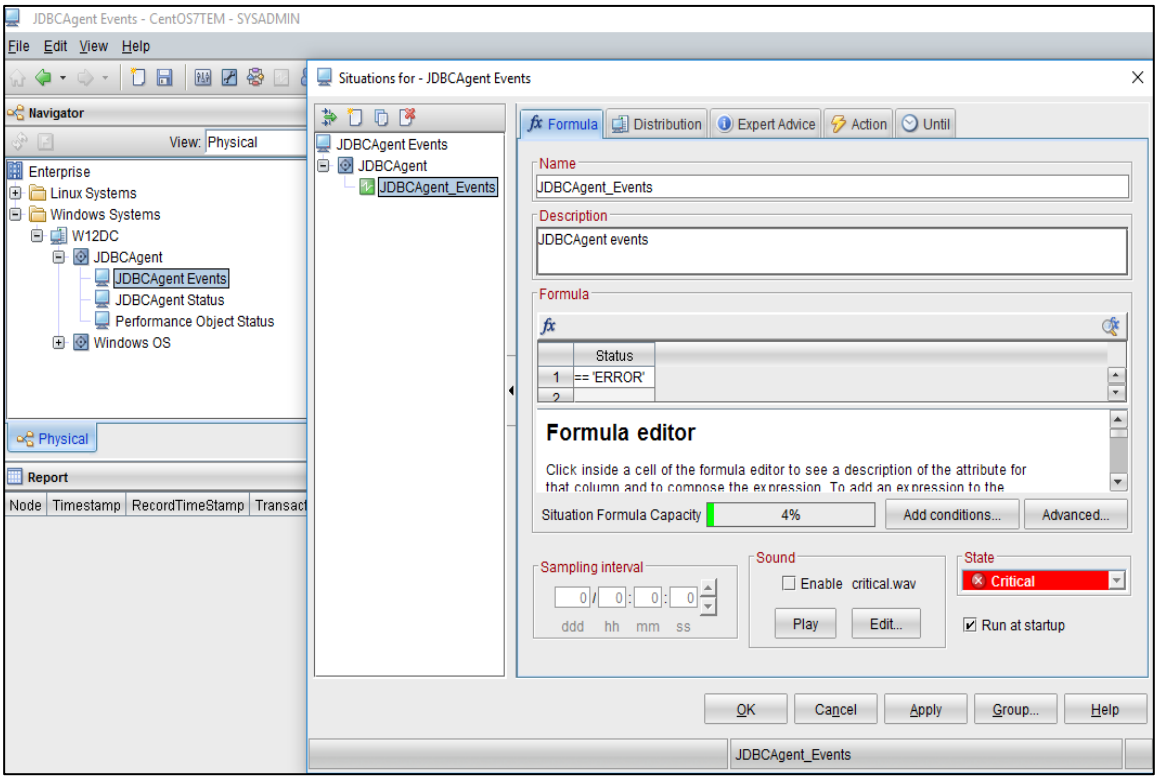
javaws.exe http://192.168.3.2:15200/tep.jnlp

V portálu je mezi monitorovanými systémy platformy Windows zobrazen server, na kterém je nainstalován IBM Windows OS Agent a vytvořený JDBCAgent. Zde jsou patrné položky JDBCAgent Events a JDBCAgent Status. Každý agent má standardně položku Performance Object Status, kde jsou zobrazeny statistiky běhu agenta.



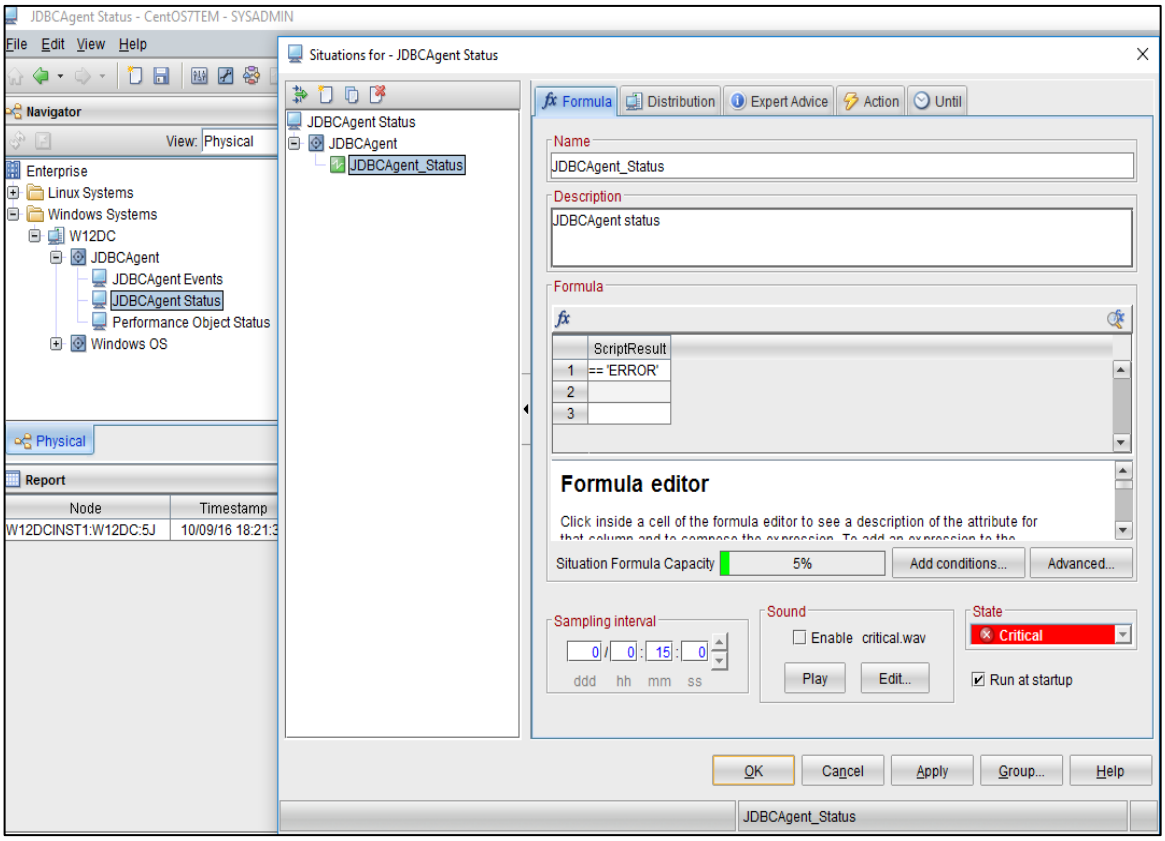
Obrázek 31:Tivoli Enterprise Portal Navigátor – vlastní zpracování

Označením položky JDBCAgent Events a stiskem pravého tlačítka myši je otevřena nabídka, kde je vybrána položka Situations. Otevře si dialogové okno pro správu situací nad vybranou skupinou atributů. Tlačítkem bílého listu je vytvořena nová situace. Situační podmínka je následující – pokud atribut Status je roven řetězci ERROR, je generována kritická událost. Pro spuštění situace při každém staru agentu je nutné zaškrtnout pole Run at startup. Jelikož bylo ve vývojovém prostředí IBM Agent Builder nastaveno, že mají být pouze kontrolovány nové řádky v log souboru, nelze nastavit vzorkovací interval. Při nastání podmínky se vytvoří ryzí událost, která musím být ručně uzavřena uživatelem.



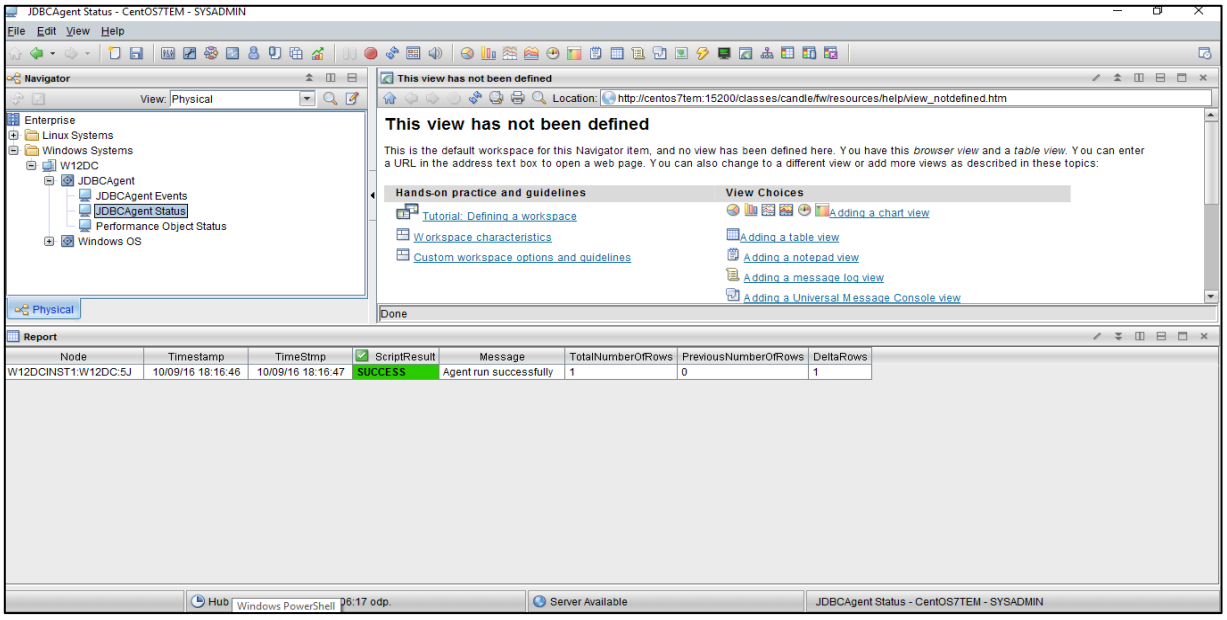
Obrázek 32: Vytvoření situační podmínky JDBCAgent_Events – vlastní zpracování

Nastavení situační podmínky nad skupinou atributů JDBCAgent Status probíhá stejným způsobem jako výše. Pokud je hodnota ScriptResult rovna řetězci ERROR, je generována vzorkovaná kritická událost. Rozdíl nastává při konfiguraci vzorkovacího intervalu. Interval udává, kdy se má daná situační podmínka ověřit. Při každém ověřování jsou načteny aktuální hodnoty pro danou skupinu atributů. Vzorkovací interval je nastaven na 15 minut.



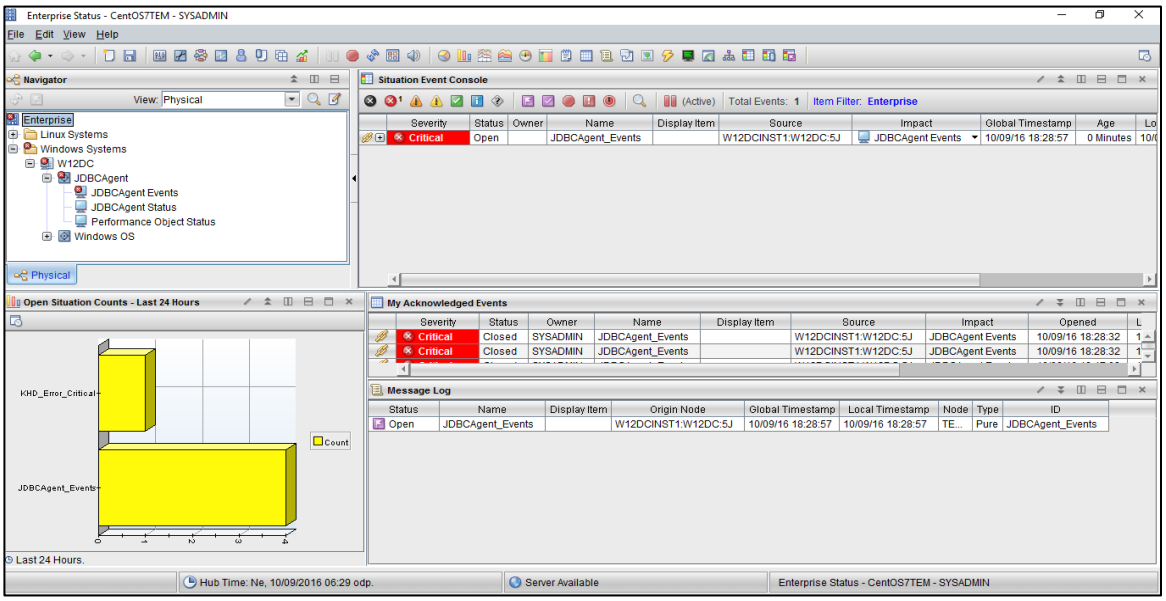
Obrázek 33: Vytvoření situační podmínky JDBCAgent_Status – vlastní zpracování

Zvolením požadované skupiny atributů je možné si prohlédnout aktuální hodnoty. Při časté aktualizaci pohledu jsou zobrazené hodnoty vzaty z interní keše. Důvodem je nižší vytížení monitorovacího serveru a infrastruktury.



Obrázek 34: Zobrazení hodnot skupiny atributů JDBCAgent Status – vlastní zpracování

Otestování agenta a nastavených situací může být provedeno zapsáním dat, která splňují situační podmínku, do databáze. Monitorovací systém aktualizuje jednotlivé hodnoty atributů a vyhodnotí situační podmínky. Pokud došlo ke shodě, je vytvořena událost, která je zobrazena v tabulce Situation Event Console. V navigátoru je vidět cesta, odkud se událost šířila. Položka Enterprise vždy ukazuje ikonu související s nejvyšší závažností, která nastala v navigačním podstromě. Ryzí události je nutné uzavírat. Při řešení je tzv. potvrzena. Potvrzením dává uživatel najevo, že se bude zabývat danou událostí. Uživatelské jméno je následně zobrazeno v poli Owner. Při spolupráci více uživatelů je jasné, kdo jakou situaci řeší. Po vyřešení musí uživatel situaci uzavřít.



Obrázek 35: Zobrazení ryzí události v Situation Event Console – vlastní zpracování

5 Výsledky a diskuse

Hlavním cílem diplomové práce bylo přiblížení problematiky monitorování systémů a aplikací v podnikovém prostředí. Nejprve byl charakterizován protokol SNMP a postupný vývoj směrem k verzi třetí. Pro identifikaci příslušných vlastností koncových systémů je nutné využít MIB soubory obsahující informace o hierarchii a vlastnostech objektů, které lze získat z koncových systémů pomocí protokolu SNMP. Soubory MIB využívají pro popis jednotlivých objektů zápis SMI verze prvé a druhé. Nejbezpečnější verzí SNMP je třetí verze, která nabízí šifrovaný přenos informací. Před nasazením zařízení v podniku je doporučeno změnit výchozí přístupové údaje.

V návaznosti na kapitolu o SNMP byl charakterizován přístup monitorování systémů s agenty a bez agentů. Hlavním rozdílem je nutnost instalace agenta na vzdálený systém. Sledování systému bez agentů je zajištěno pomocí funkcionality vzdáleného systému, z kterého jsou data získávána pomocí SNMP, WMI nebo CIM rozhraní. Monitorování s agenty je založeno na nutnosti instalace agenta na monitorovaný systém.

Nedílnou součástí monitorování je zpracování informací z logovacích systémů. Shrnuta byla základní doporučení pro tvorbu chybových zpráv, které vedou k úspěšnému řešení problematických stavů. Na platformě Microsoft Windows byl představen základní logovací systémem Windows Event Log. Dále byl charakterizován standard Syslog, za kterým stojí uskupení IETF.

Následující kapitoly byly věnovány nástrojům a doporučením při tvorbě agentů. Velké množství agentů přijímá na vstupu řetězcové hodnoty, které dále zpracovává. Ze zmíněného důvodu byly představeny regulární výrazy, které usnadňují zpracování řetězců. Ukázky využití regulárních výrazů byly provedeny v jazycích Perl a Java. Doporučena byla integrovaná vývojová prostředí a další podpůrné nástroje pro vývoj a ladění výsledných programů a skriptů v jazycích Perl a Java.

Vývoj agentů určených pro systém IBM Tivoli Monitoring probíhá v prostředí IBM Tivoli Agent Builder. Agent Builder je založen na vývojovém prostředí Eclipse. Představeny byly možnosti tvorby základních agentů, které nabízí dané prostředí.

V závěru teoretické části byl představen systém IBM Tivoli Monitoring a jednotlivé komponenty.

Praktická část navázala na teoretické poznatky. Pro ukázku byl vytvořen agent, který monitoroval nové řádky v databázi PostgreSQL. Agent byl založen na šabloně, která využívá výstupu z programu. Program agenta byl vytvořen v jazyce Java, zabalen do instalačního balíku a nainstalován. Proces vytvoření agenta byl charakterizován v úvodní části. Nejprve bylo analyzováno zadání a následně vytvořen výkonný program. Výsledný agent byl nakonfigurován ve vývojovém prostředí IBM Agent Builder a zabalen do instalačního balíku.

Ruční úprava vytvořených skriptů byla ukázána před samotnou instalací agenta. Nejprve byla nainstalována podpora agenta na komponenty Tivoli Enterprise Monitoring Server a na Tivoli Enterprise Portal Server. Dále byla provedena instalace agenta na cílový systém, vytvořena výchozí konfigurace a na závěr byl agent spuštěn.

Závěrečná část se zabývala zobrazením nasbíraných dat v IBM Tivoli Enterprise Portal a nastavením situací. V portálu byla ověřena správnost instalace agenta a vytvořeny situace, které generují požadované události. Ověření správné funkce agenta bylo provedeno přidáním požadované řádky do sledované tabulky. Při následujícím spuštění situace došlo k vytvoření události.

6 Závěr

Monitorování systémů a aplikací v podnikovém prostředí je nutnou činností, která vede k dlouhodobé stabilitě a prosperitě podniku. Vhodným řešením je integrované prostředí, kde jsou agregovány všechny události a výstupy jednotlivých agentů. Společnost IBM je jedním z předních výrobců software a produkt IBM Tivoli Monitoring se řadí mezi špičku na poli monitorování. V práci je charakterizováno prostředí IBM Tivoli Monitoring a ukázán postup vývoje a integrace agenta. Podpora ze strany IBM je na dobré úrovni a poskytování nástrojů pro vývoj agentů je samozřejmostí.

Hlavního cíle práce bylo dosaženo. Problematika monitorování systémů v podnikovém prostředí je komplexní činnost a jednotlivé dílčí cíle nastínily okruhy vědomostí, které je vhodné znát.

Práce obsahuje doporučení, které nástroje je dobré využít pro vývoj agentů a zjednodušit samotný proces tvorby agenta.

Pro další výzkum je možné srovnat monitorovací nástroje jiných dodavatelů a ukázat tvorbu a integraci agenta.

7 Citovaná literatura

1. **Mauro, Douglas R.** *Esential SNMP Second Edition.* Sebastopol : O'Reilly Media, Inc., 2005. 978-0-596-00840-6.

2. **Bigelow, Stephen J.** *Mistrovství v počítačových sítích.* Brno : Computer press, 2004. 80-251-0178-9.

3. **Rose, M. a McCloghrie, K.** Structure and Identification of Management Information for TCP/IP-based Internets. *IETF Tools.* [Online] [Citace: 17. 8 2016.] https://tools.ietf.org/html/rfc1155.

4. **Rose, M. a McCloghrie, K. .** Concise MIB Definitions. *IETF Tools.* [Online] [Citace: 17. 8 2016.] https://tools.ietf.org/html/rfc1212.

5. **IBM.** IBM Tivoli Monitoring Version 6.3 Installation and Setup Guide. *IBM Knowledge Center.* [Online] [Citace: 22. 8 2016.] https://www.ibm.com/support/knowledgecenter/SS3JRN_7.2.1/com.ibm.itm.doc_6.3/itm63_install.pdf.

6. **Microsoft.** Common Information Model. *Microsotf Developer Network.* [Online] [Citace: 4. 8 2016.] https://msdn.microsoft.com/en-us/library/aa389234(v=vs.85).aspx.

7. **Microsoft**. Windows Management Instrumentation. *Microsoft Developer Network.* [Online] [Citace: 2. 8 2016.] https://msdn.microsoft.com/en-us/library/aa394582(v=vs.85).aspx.

8. **Microsoft**. Logging Guidelines. *Windows Dev Center.* [Online] [Citace: 15. 8 2016.]https://msdn.microsoft.com/en-us/library/windows/desktop/aa363667(v=vs.85).aspx.

9. **Splunk.** Logging best practices. *splunk>dev.* [Online] [Citace: 16. 8 2016.] http://dev.splunk.com/view/logging-best-practices/SP-CAAADP6.

10. **Microsoft.** Event Logs. *Microsoft Technet.* [Online] [Citace: 16. 8 2016.] https://technet.microsoft.com/en-us/library/cc722404(v=ws.11).aspx.

11. **R. Gerhards a G. Adiscon.** The Syslog Protocol. *tools.ietf.org.* [Online] [Citace: 8. 12 2016.] https://tools.ietf.org/html/rfc5424.

12. **linux.die.net.** syslog(3) - Linux man page. *linux.die.net.* [Online] [Citace: 8. 17 2016.] http://linux.die.net/man/3/syslog.

13. **AARON, LESKIW.** Understanding Syslog: Servers, Messages & Security. *NetworkManagement Software.* [Online] [Citace: 8. 7 2016.] http://www.networkmanagementsoftware.com/what-is-syslog/.

14. **IBM.** IBM. IBM Tivoli Agent Builder Version 6.3.1 User's Guide. *IBM developerWorks.* [Online] [Citace: 6. 12 2016.] http://www.ibm.com/support/knowledgecenter/SSTFXA 6.3.0/com.ibm.itm.doc 6.3/agentbuilder63.

15. **Friedl, Jeffrey E. F.** *Mastering Regular Expressions Third Edition.* Sebastopol : O'Reilly Media, Inc., 2006. 0-596-52812-4.

16. **Dařena, František.** *Myslíme v jazyku Perl.* místo neznámé : Grada, 2005. 978-80-247-6390-3.

17. **Schildt, Herbert.** *Mistrovství - Java.* Brno : Computer Press, 2014. 978-80-251-4145-8.

18. **IBM.** IBM Tivoli Monitoring Version 6.3 Administrator's Guide. *IBM Knowledge Centre.* [Online] [Citace: 16. 4 2016.] http://www.ibm.com/support/knowledgecenter/SSTFXA_6.3.0/com.ibm.itm.doc_6.3/itm63_admin.pdf.

19. **IBM**. Úvodní stránka produktu Tivoli Monitoring. *IBM Knowledge Center.* [Online] [Citace: 16. 4 2016.] http://www.ibm.com/support/knowledgecenter/SSTFXA/welcome.