Messaging Platform [T-PL:MP] [CRM-887]

DANE PRODUKTU

Start komercyjny	2003-01-01
Koniec komercyjny	unknown
Partnerzy	Avantis
Product Manager	Wojciech Koszut
Analityk Biznesowy	Wojciech Koszut
System Analyst	Wojciech Koszut
Architekt	Tomasz Knyziak
SLA	n/d
Testowy adres	http://november:8080/jmx-console/HtmlAdaptor
Produkcyjny adres	https://thunder:8443/jmx-console/HtmlAdaptor
Aplikacje wspomagajce produkt	

CEL BIZNESOWY

Implementacja i wdroenie na produkcji klastra MP 3.0 sprzegnietego z MP 2.1 na którym bdzie dziaaa conajmniej jedna usuga.

OPIS BIZNESOWY

Zwikszenie niezawodnoci i dostpnoci kluczowego systemu Avantis.

DOKUMENTACJA PRODUKTU

Wymagania niefunkcjonalne

- poziom dostpnoci (procentowe ujcie czasu, w którym serwisy mog wymienia wiadomoci pomidzy sob) 99,95%
- moliwo pracy na wielu wzach TAK
- czas uruchomienia pojedynczego wza max. 180s przy pustej kolejce
- stopie transakcyjnoci - ATOMIC

- liczba serwisów obsugiwanych równolegle - 10000

- rednia dobowa przepustowo interfejsów wejciowych (SRV -> CORE, msg/h) 360000
- szczytowa przepustowo interfejsów wejciowych (msg/s) 400
- maksymalny czas odpowiedzi interfejsów wejciowych (ms) 1000
- maksymalna rónica czasu odpowiedzi interfejsów wejciowych pomidzy najwolniejszym a najszybszym serwisem (%, idealnie 0) 10%

- rednia dobowa przepustowo interfejsów wyjciowych (SRV <- CORE, msg/h przy pomijalnym czasie obsugi w serwisie) 360000
- szczytowa przepustowo interfejsów wyjciowych (msg/s) 400
- maksymalny czas odpowiedzi interfejsów wyjciowych (ms) 5000
- maksymalna rónica czasu odpowiedzi interfejsów wejciowych pomidzy najwolniejszym a najszybszym serwisem (%, idealnie 0) 50%

MP 2.1

DoD

Requirement/Task

- writing code
- mark task with Jira plugin for Idea

Sprint

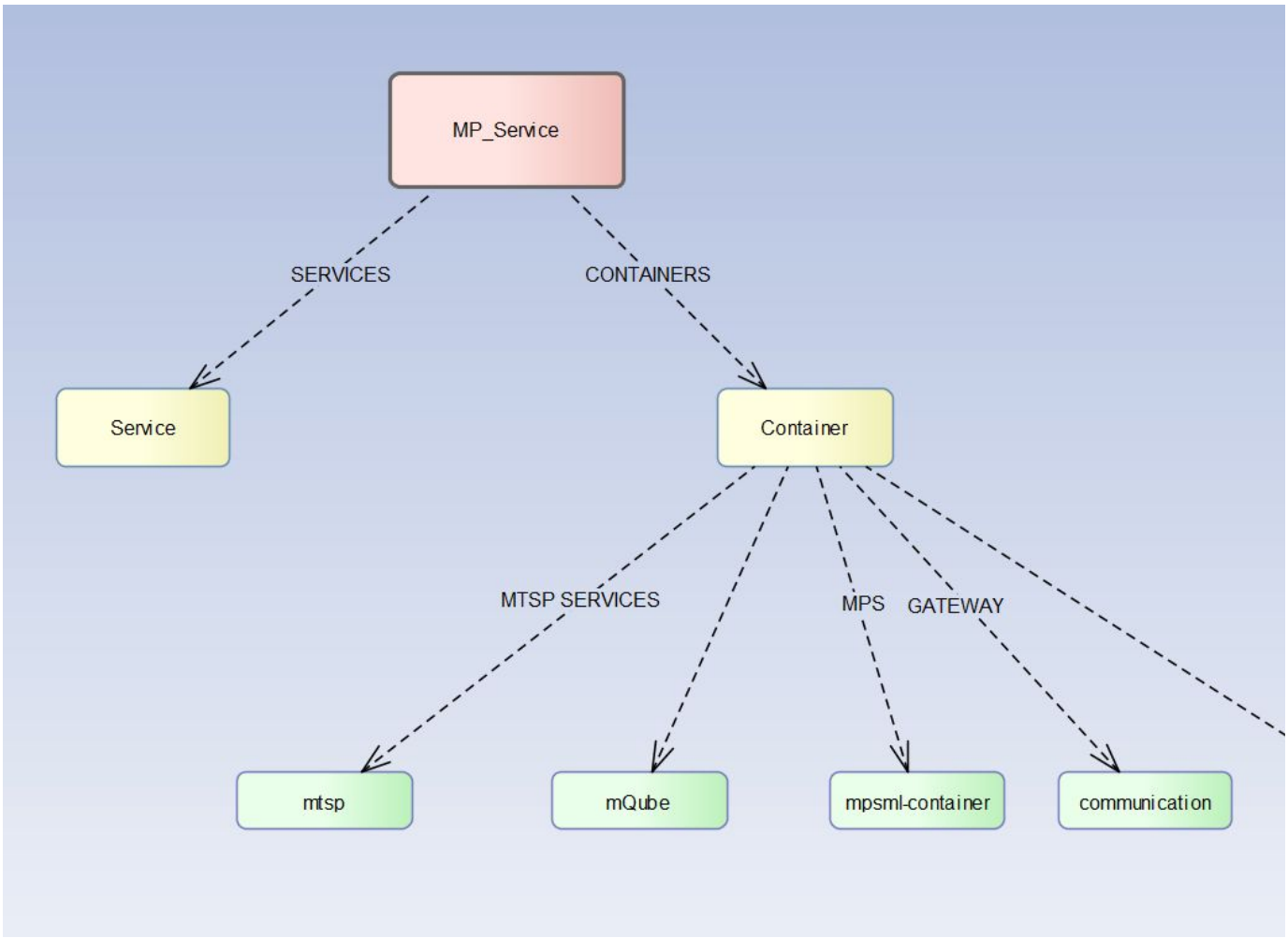
- pass meta project information
- pass static code analysis
- pass unit tests requirements

Release

Wymagania niefunkcjonalne

Podzia usug MP

- Obrazujcy podzia na rodzaje mp usug. Na pojedyncze i wpite do kontenerów, oraz na rodzaje kontenerów.



Usugi komunikacyjne

- W skad usug komunikacyjnych mona zaliczy bramki sms/mms dla operatorów oraz usugi do komunikacji z partnerami.
- Usugi komunikacyjne znajduj si w: ***/usr/local/mpservices/comm***
- Logi usug komunikacyjnych znajduj si w: ***/usr/local/logs/mpservices/comm***
- Szczegóowy opis bramek komunikacyjnych mona odnale: https://wiki.avantis.pl/bin/view/Utrzymanie/BramkiKomunikacyjneAgf

Usugi biznesowe i technologiczne

- W skad usug biznesowych mona zaliczy szereg aplikacji speniajcych rónorodne funkcje. Midzy innymi serwisy MT/konkursy/usugi serwujce content, usugi wystawiajce aplikacjom WWW metody poprzez RMI.
- Usugi znajduj si w: ***/usr/local/mpservices/services***
- Logi usug biznesowych znajduj si w: ***/usr/local/logs/mpservices/services***

Usugi systemowe

- W skad usug systemowych mona zaliczy usugi monitorujce takie jak arhplus, mpnotifier oraz inne usugi systemowe niezbdne do dziaania pozostaych usug(np. rmiregister)
- Usugi systemowe znajduj si w: ***/usr/local/mpservices/system***
- Logi usug biznesowych znajduj si w: ***/usr/local/logs/mpservices/system***

Konfiguracja usug mp

Opis moliwej konfiguracji znajdujcej si w bazie.

- MPSERVICE

```
SERVICE_ID - unikalny identyfikator usugi
NAME – nazwa usugi
ACTIVATED – status aktywnoci usugi.
DEPLOY_TIMESTAMP - czas dodania usugi
CONFIGBEAN_ID - klucz obcy do tabeli
BU - N/A
PROJECT_MANAGER - Imi i nazwisko osoby z bisnesu odpowiedzialnej za usug.
IT_DEVELOPER - Imi i nazwisko osoby, która stworzya dany serwis. W przypadku MPSów - imi i nazwisko
osoby, która wdroya dany serwis.
TASK_ID - Numer taska na wdroenie serwisu.
ACTIVE_FROM - Data rozpoczcia dziaania serisu.
ACTIVE_TO - Data zakoczenia dziaania serwisu.
FULL_BU - N/A
DOT_TASK - Numer taska tworzonego serwisu w aplikacji DotProjekt?.
SVN - Link do repozytorium usugi w Subversion (SVNie).
DEV_EMAIL - Adres mailowy developera
```

- MPSERVICE_CONFIGURATIONS w której znajduj si szczegóowe informacje na temat aktualnej konfiguracji serwisu.

```
CONFIG_AUTO_ID- klucz gówny czcy tabel configuration z tabel mpservice.
LINEAR_CONFIG - konfiguracja serwisu w formie pliku w formacie XML.
OWNER_SERVICE_ID - id serwisu, którego dotyczy konfiguracja.
DEPLOY_TIME - czas dodania konfiguracji uslugi.
SERVICE_TYPE - typ serwisu (1 - usuga komunikacyjna 2 - usuga biznesowa)
MAX_REDELIVERY_NUM - ilo retransmisji pojedynczych wiadomoci w przypadku bdu (0 –
nieskoczono)
MAX_MESSAGES - ilo równolegych wtków którymi s dostarczane wiadomoci do usugi.
SERVICE_DESC - sowny opis co dana usuga robi.
OWNER_SERVICE_NAME - nazwa usugi której dotyczy konfiguracja.
START_DATE - Data rozpoczcia dziaania serwisu.
STOP_DATE - Data zakoczenia dziaania serwisu.
PRODUCTION_STATUS - status ( 10 - usuga produkcyjna, 2 - usuga nieprodukcyjna)
```

- GATEWAY posiadaj dodatkow konfiguracj zawart w polu LINEAR_CONFIG w MPSERVICE_CONFIGURATIONS
- Dotyczy tylko bramek PLUS oraz ORANGE, ich konfiguracja róni si w zalenoci od operatora i LA

```
<property name="la" value="xxx" />
<property name="smsc_port" value="xxx" />
<property name="smsc_host" value="xxx" />
<property name="login" value="xxx" />
<property name="password" value="xxx" />
<property name="alive_time" value="xxx" />
<property name="alive_timeout" value="xxx" />
<property name="redelivery_time" value="xxx"
/>
<property name="window_size" value="xxx"/>
```

Dodawanie nowej MP Usugi

- Rezerwacja MPID w MP Core
- Wdroenie MPS

Dokumentacja deweloperska MP 2.1

MP 2.1 (Messaging Platform 2.1) jest autorskim rozwizaniem typu MoM, rozwijanym - najpierw w Emisji pod nazw EMF, póniej w Avantis / DV ju od 2002 roku. Standard, w którym rozwizanie jest stworzone naleaoby nazwa JEE5 (cho nie obyo si bez gwatów na specyfikacji ;) - a kluczowymi elementami JEE5, które wypadaoby zna, eby porusza si w miar sprawnie po kodzie s specyfikacje: EJB 3.0, JPA 1.0 i JMS 1.1.

EJB 3.0 peni rol modelu komponentowego - to dziki beanom sesyjnym (w wikszoci bezstanowym - SLSB) mamy powstrzykiwane zalenoci pomidzy elementami systemu, jak równie opublikowane na wiat fasady (API dostpne dla serwisów MP spoza maszyny wirtualnej MP).

JPA 1.0 peni rol interfejsu do baz danych - za jego porednictwem odczytujemy konfiguracje, routingi i inne informacje z baz danych. O baz danych oparty jest równie mechanizm wiadomoci opónionych - si rzeczy równie opiera si on o JPA.

JMS 1.1 jest mechanizmem dostpu do kolejek. Kolejki s bodaj najwaniejszym elementem MP - istot dziaania platformy jest bowiem umieszczanie odpowiednich wiadomoci w odpowiednich kolejkach - i umoliwianie rozmaitym komponentom zjadanie wiadomoci z tych kolejek Kolejki - w zdecydowanej wikszoci - s nieograniczone i trwae, czyli e ich stan jest po cichu zapisywany w jeszcze jednej bazie danych.

Jasiek - gówny autor - wymyli co takiego, jak architektur OSOQ - One Service - One Queue, co oznacza, e kadej usudze MP jest dedykowana osobna kolejka. Pomys ten zrodzi si jako konsekwencja innej niecodziennej rzeczy w architekturze - aplikacja JEE wpycha wiadomoci do zdalnych interfejsów konsumentów (normalnie, konsumenci powinni czy si do kolejek udostpnianych przez JEE i zasysa z nich wiadomoci) - co oznaczao, e przy jednej kolejce serwisowej i jakim rozdzielaczu, wolne dziaanie jednej usugi spowolniao dziaanie wszystkich pozostaych. Có, takie zaoenia.

Cao jest aktualnie wdroona na serwerze aplikacyjnym JBossAS w wersji 4.2.3 i nie ma planów uruchamiania go na niczym innym.

Do skompilowania i uruchomienia MPCore wymaga Javy w wersji co najmniej 1.7 (switch ze Stringiem, try-with-resources).

Core vs. serwisy

Serwisy MP to takie programiki w Javie, które mog podczy si do core'a, wysya za jego porednictwem wiadomoci do innych serwisów - jak równie odbiera wiadomoci dla nich przeznaczone. Brzmi prosto - bo i w sumie skomplikowane by nie musi.

Ale jest :)

Serwis MP - co jest w miar naturalne - moe (przez RMI) wywoywa metody wystawiane na zdalnych interfejsach beanów sesyjnych core'a. To oznacza, e jeeli gdziekolwiek w kodzie core'a znajdziemy interfejs oznaczony adnotacj @Remote - jego metody s dostpne dla serwisów.

To, co jest nienaturalne to to, e usuga - rejestrujc si - przekazuje core'owi referencj na implementowany przez siebie interfejs MPService - dajc w ten sposób core'owi moliwo wykonywania operacji na samym sobie. Taka konstrukcja, cho w oczywisty sposób dziaa, jest niezgodna ze standardem JEE i stanowi gówn, jeeli nie jedyn przeszkod w klastrowaniu MP.

Tym niemniej, yjemy z tym co mamy - zatem czas na cykl ycia usugi. Bierze w tym udzia bean o wdzicznej nazwie MPServiceRegistryBean, bdcy singletonem (tu uwaga - standard EJB wprowadzi singletony dopiero w wersji 3.1, zatem w JBossie 4.2.3 nie s one dostpne - podpieramy si specyficzn dla JBossa protez zwan POJO Service - dziaa tak samo). Kluczem jest metoda register - to j serwis wywouje na samym pocztku podczania do MP - i przekazuje tam adres (ID serwisu) i referencj na implementowany przez siebie interfejs MPService. Metoda register podejmuje prób wyrejestrowania starej kopii serwisu - co, jeeli pada ona na skutek np. zabicia z dziewitki - w oczywisty sposób si nie powiedzie, wic ju na tym etapie mamy element, który moe si zaci :)

Zakadajc, e udao zarejestrowa - MP tworzy po stwojej stronie co takiego jak MPServiceProxy - obiekt bdcy owijk na interfejs serwisu - to za jego porednictwem core bdzie upycha w serwisie wiadomoci (wywoujc metod deliver), odpytywa o samopoczucie… Serwis nastpnie pobiera swoj konfiguracj (przechowywan po stronie MP, metoda getServiceConfiguration), konfiguruje sam siebie i jest gotów do startu.

Start na dobr spraw odbywa si lokalnie (medody init() i start() na AbstractMPService). Do MP jest przekazywany tylko rezultat przejcia w inny stan.

Wejcie wiadomoci

Punktem wejcia wiadomoci do MP jest pojedynczy zdalny interfejs - MPFacadeRemote. Jest on wystawiony przez bezstanowy bean pl.avantis.mp.core.router.MPFacadeBean i udostpniony przez JNDI - dziki temu kady serwis MP, znajc adres, pod którym pracuje MP Core, moe pobra referencj do tego interfejsu i wywoywa na nim metody po RMI. Interfejs jest prosty jak konstrukcja cepa, zawiera raptem cztery metody:

- Long sendMessage(MPMessage)
- LinkedList<Long> sendMessages(LinkedList<MPMessage>)
- void correctAccounting(ServiceAddress, Long, MPAcknowledgeStatus)
- boolean sendJsonSerializedEdwEvent(String)

Jak si atwo domyli, do wysyania wiadomoci su pierwsze dwie - przy czym druga z nich umoliwia przeprowadzenie tej operacji hurtem (nie odpowiem na pytanie, dlaczego wymuszamy uywanie LinkedList zamiast dowolnej listy, odpowiedzi na nie nie zna sam ksie ciemnoci). Wany jest fakt (zdanie prawdziwe od kwietnia 2014) - obydwie operacje s w peni transakcyjne, co oznacza, e jeeli nie powiedzie si jakikolwiek ich krok (p. niej), zostan w caoci wycofane. Innymi sowy, jeeli jedna wiadomo w paczce bdzie za, do MP nie wejdzie adna ;).

Co dokadnie dzieje si w momencie wejcia wiadomoci do systemu

Wiele rzeczy :)

Na skutek nieszczliwego splotu wypadków (znanego równie jako Przemona Ch Jaka Do Zastosowania Wszystkich Istniejcych Mechanizmów JEE5) pierwsza rzecz, która si dzieje, to nadanie wiadomoci identyfikatora… przez Interceptor. Dlaczego tak, nikt nie zgadnie - wane jest to, e wanie tam - id jest pobierany ze specjalnego beana (opisz go niej) i przylepiany do wiadomoci.

Kolejna rzecz - ju w samej metodzie send, poza interceptorem - to otwarcie sesji JMS, dziki której bdzie mona umieci wiadomoci w kolejkach (ta sama sesja moe suy do umieszczenia wielu wiadomoci w wielu kolejkach). Nastpnie, kada z wiadomoci z przesanej listy (a pojedyncze send de facto po spodem wywouje send z list majc jeden element ;) jest routowana - cilej, wyznaczana jest priorytetyzowana lista usug, do których wiadomo moe trafi.

Routing dziaa wg. nastpujcego algorytmu:

- Jeeli w wiadomoci by okrelony serwis docelowy, na list trafia tylko on i cze.
- Jeeli nie by, MP bierze wszystkie tablice routingu przywizane do serwisuródowego, sprawdza warunki (payloady, large accounty, treci…) i ukada wg. priorytetów. Kiedy by jeszcze taki pomys, e jak znajdzie si wicej, ni jeden routing o tym samym priorytecie, wiadomo bdzie klonowana i wysyana w kopiach równolegle. W praktyce ten mechanizm nigdy nie jest wykorzystywany.

Jeeli powstaa lista jest pusta, leci NoRoutingRules - jeeli nie jest pusta, wiadomo trafi do pierwszego serwisu na licie - ale zanim to nastpi, sprawdzany jest jeszcze jeden warunek: czy czas wysania wiadomoci jest w przyszoci - czy pusty albo w przeszoci. W pierwszym przypadku, wiadomo trafia do kolejki wiadomoci opónionych. W drugim - do kolejki serwisowej.

Nastpnie - jeeli z jaki tam warunków wynika, e wiadomo powinna by zaccountowana, jej kopia jest umieszczana w kolejce accountingowej. Podobnie, jeeli z jaki tam warunków wynika, e w momencie wysania wiadomoci powinien by zrzucony generyczny RBR, jest wrzucany do kolejki biznesowej.

I to w zasadzie koczy sam proces po stronie MP. Wtki poboczne, które si w midzyczasie pojawiy, opisz w podpunktach.

Identyfikatory wiadomoci

Identyfikatory wiadomoci s pobierane ze singletona MessageIdentifierService - teoretycznie umoliwia on zapisywanie biecej wartoci licznika albo w bazie, albo w pliku - praktycznie jest to baza danych (mechanizm TableGenerator z JPA). Tym niemniej, eby nie siga do bazy za kadym razem, pewna pula (konkretnie - 1000) identyfikatorów jest rezerwowana i rozdawana z AtomicLonga - tak wic zmiany w bazie s dokonywane de facto co 1000 pobra ID.

Warunki accountowania wiadomoci

Nie wszystkie wiadomoci wpadajce do MP s accountowane - w duej mierze decyzja o tym, czy dana wiadomo podlega temu procesowi zaley od przenoszonego przez ni payloadu (np. domylnie ObjectPayloady nie s accoutowane) oraz tzw. roli wiadomoci:

- Jeeli rola wiadomoci to INQUIRY, INQUIRY_RESPONSE (uywana do pingowania, czym odpowie serwis, eby mona byo poda informacj o kosztach SkyCashowi) lub TEST/TEST_RESPONSE (uywane przez mechanizmy monitorujce), wiadomo nie jest accountowana.
- W przypadku innych ról, sprawdza sie payload. Payloady uytkowe (SMSy, MMSy) s zawsze accountowane. Accountowanie ObjectPayloadów zaley od ustawionej na nich flagi “supportAccounting” (domylnie - false).
- Acki s accountowane specjalnie, natomiast z punktu widzenia MP - s po prostu na pa wrzucane do kolejki accountingowej.

Sam accounting to zewntrzny proces - podcza si do kolejki accountingowej z drugiej strony i wysysa wiadomoci z kolejki ACC. MA zdefiniowane struktury JPA dla wikszoci wiadomoci i payloadów i stara si transakcyjnie zapisa wiadomoci w bazie. Jeeli nie uda si to podczas 36 prób (przy normalnej pracy - ok. 3 godzin

Wyjcie wiadomoci

Rozmow o wyjciu wiadomoci porzucilimy na etapie rejestracji serwisu w MP i powstajcym obiekcie MPServiceProxy. Jedn ze skadowych tego obiektu jest kolekcja obiektów MPDispatchExecutor, której wielko odpowiada skonfigurowanej liczbie wpadajcych równolegle do serwisu wiadomoci. MPDispatchExecutor jest de facto listenerem JMSowym na kolejce serwisowej (uwaga - nie s to beany MDB, poniewa te maj na sztywno okrelane kolejki, na których nasuchuj), którego zadaniem jest zjedzenie wiadomoci z kolejki i próba upchnicia jej w zdalnej (wystawianej przez MP-Serwis) metodzie ‘deliver’. To, ile wymaga to zamieszania i apania wyjtków (wszake wiadomo nieupchnita w serwisie nie moe zgin), to ju osobny rozdzia. Bez wtpienia, operacja ta nie jest transakcyjna i tylko od skrupulatnoci pokole programistów zaley rzetelno retransmisji wiadomoci wychodzcych do serwisu.

Metoda “deliver” moe si po stronie mp-serwisu zakoczy rozmaicie - serwis moe si wywali (rzuci RuntimeException albo MPServiceFailedException), moe si zawiesi (wtedy muka, bo wtek w MP si zacina), moe rzuci specjalnym MPDelayedDeliveryException - wtedy wiadomo teoretycznie powinna wyldowa w opónionych. Nad poprawnoci tego procesu czuwa skomplikowana banda ifów, try’ów i catchy

Wdroenie MP 2.1 od goego metalu

Przygotowanie baz danych

1. Zakadamy nowy schemat w MySQLu: mptimers. Dodaj dwóch uytkowników: mptimers i mpjms. Pierwszemu daj pene prawa do schematu mptimers, drugiemu pene prawa do schematu mpjms
2. Dodajemy pole DELIVERY_TIME typu TIMESTAMP(6) w tabeli ACCOUNTING_RECORD w schemacie mp (do zapisywania planowanego czasu dostarczenia wiadomoci)
3. Dodajemy pola IS_SDA_REQUEST i IS_UDA_REQUEST - obydwa typu NUMBER(1,0) - w tabeli SMS_RECORD
4. W schemacie MP dodajemy tabel MID_STORE, z kluczem gównym w polu MID_STORE_ID (NUMBER 19,0) i, dodatkowo, polem CURRENT_VALUE (NUMBER 19,0).

Przygotowanie MP Core

1. Root zakada katalogi /usr/local/mpcore2.0 i /usr/local/logs/mpcore2.0 i zmienia im waciciela i grup na mpadmin.
2. Wypakowuj dystrybucyjnego JBossa 4.2.3 (wersja dla JDK6) do /usr/local/mpcore2.0. Od tej pory bdzie to moja cieka bazowa.
3. W pliku ./bin/run.conf, ustawiam JAVA_HOME na ciek znalezion w punkcie 2, a JAVA_OPTS zmieniam tak, eby wyglday tak jak te: JAVA_OPTS="-server -Xmx8096m -Xms8096m -XX:+UseParallelGC -Dsun.rmi.transport.connectionTimeout=300000 -Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=3600000 -Djboss.server.log.dir=/usr/local/logs/mpcore2.0"
4. Wywalamy katalogi ./server/all i ./server/minimal
5. Do katalogu ./server/default wycigam binarki MP z lokalizacji: https://svn.avantis.pl/repos/dev/avantis/release/messaging-platform/trunk/

KONFIGURACJA PLATFORMY MP

Klonowanie wiadomoci z kolejki monitoringQueue do monitoringQueueClone

Na potrzeby serwisu mp-config-pump powstaa funkcja kopiowania wiadomoci trafiajcych do kolejki monitoringQueue do monitoringQueueClone.

Wspomniany serwis przenosi (midzy innymi) wiadomoci z kolejki MP monitoringQueueClone na coreApp do monitoringQueue na thunder. Poniewa kolejka monitoringQueueClone nie jest potrzebna na platformie thunder, domylnie funkcja klonowania jest wyczona. Aby j wczy (obecnie na platfiermie coreApp) naley w pliku /usr/local/mpcore/server/default/conf/mpconfig.xml doda wpis:

```
<cloningMonitoringQueue>true</cloningMonitoringQueue>
```

USUGI I PROJEKTY

[DOC] Restart MP

[DOC] Tworzenie nowych mp-service

[DOC] Zmiana routingu MP na potrzeby dodania nowego operatora

Avantis Cache [T-PL:MP->ACA] [CRM-864]

Gateways [T-PL:MP->GW]

- [DOC] Bramki Komunikacyjne AGF - dokumentacja techniczna
  - [DOC] Bramki Sowackie
  - [DOC] Lista Bramek Zagranicznych
- [DOC] Bramki USSD
- [DOC] Virgopass - agregator komunikacji
- Dimoco-gw
  - [MPID-101236] dimoco-gateway
- Dimoco PSMS Gateway
- PauProxy - Dokumentacja techniczna

MP 3.0

MP Console [T-PL:MP->CON]

MP Core [T-PL:MP->MPCORE]

- [DOC] Kolejki Mp
- [DOC] Messaging Platform Monitoring
- [DOC] mp-config-pump
- [DOC] MpCore Service - kolejki JMS - instrukcje techniczne
- [DOC] MP Secondary Platform
- [DOC] Procedura upgrade'u MP

MT Subscription Gateways [T-PL:MP->MTGW]

- Dimoco MT Gateway
  - [MPID-101241] dimoco-mt-gateway
- Dimoco MT Gateway - tech
- Dokumentacja bramki Agmo MT Czechy
  - [MPID-101321] agmo-mt-gateway-Czechy
- MediaSat MT Gateway
- Telemedia Romania MT Gateway
  - [MPID-101237] telemedia-ro-mt-gateway

Payloads

- NotificationPayload

Historia zmian

User	Edits	Last Update	Watches
Wojciech WK. Koszut	16	1619 days ago	0
Tomasz TM. Matoszka	11	1856 days ago	0

Piotr PS. Skonieczny	6	2087 days ago	0
Unknown User (testerek)	2	2224 days ago	0
Kamil KW. Wojewoda	1	2231 days ago	0
Unknown User (tknyziak)	1	2151 days ago	0
Unknown User (tmajewski)	1	1626 days ago	0
ConfluenceApiAccess	0	1633 days ago	0