7.5

Mobilní systém zpráv a M2M

Poznámka

Než začnete používat tyto informace a produkt, který podporují, přečtěte si informace, které uvádí “Poznámky” na stránce 189.

Toto vydání se vztahuje k verzi 7, vydání 5 produktu IBM® WebSphere MQ a ke všem následujícím vydáním a modifikacím, dokud nebude v nových vydáních uvedeno jinak.

Obsah

Úvod do produktu MQTT

Informace o odesílání zpráv mezi mobilními aplikacemi pomocí přenosu telemetrie MQ (MQTT). Protokol je určen pro použití v bezdrátových sítích a sítích s nízkou šířkou pásma. Mobilní aplikace používající produkt MQTT odesílá a přijímá zprávy voláním knihovny MQTT . Zprávy se vyměňují prostřednictvím serveru systému zpráv MQTT . Klient a server MQTT zvládají složité aspekty spolehlivého doručování zpráv pro mobilní aplikaci a udržují nízké náklady na správu sítě.

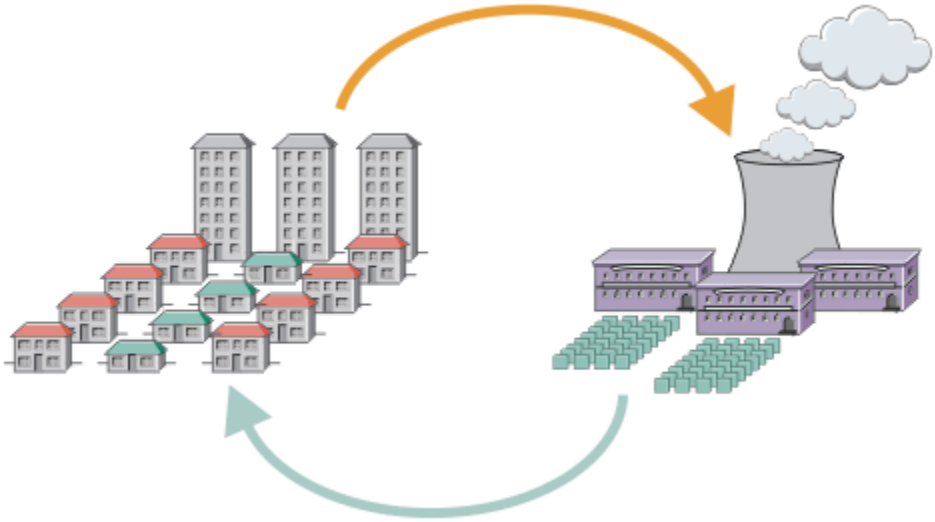
Aplikace MQTT běží na mobilních zařízeních, jako jsou chytré telefony a tablety. Produkt MQTT se také používá pro telemetrii pro příjem dat ze senzorů a pro jejich vzdálené řízení. Pro mobilní zařízení a senzory nabízí společnost MQTT vysoce škálovatelný protokol publikování/odběru se zajištěným doručením. Chcete-li odesílat a přijímat zprávy MQTT , přidejte do své aplikace knihovnu klienta MQTT .

Knihovna klienta MQTT je malá. Knihovna se chová jako poštovní schránka, odesílá a přijímá zprávy s jinými aplikacemi MQTT , které jsou připojeny k serveru MQTT . Posíláním zpráv namísto připojení k serveru, který čeká na odezvu, šetří aplikace MQTT životnost baterie. Knihovna odesílá zprávy na jiná zařízení prostřednictvím serveru MQTT , na kterém běží protokol MQTT version 3.1 . Můžete odesílat zprávy do specifického klienta nebo používat systém zpráv publikování/odběru pro připojení mnoha zařízení.

Knihovny klienta MQTT připojují aplikace pro mobilní zařízení a senzory k serveru MQTT pomocí protokolu MQTT .

IBM MessageSight a IBM WebSphere MQ jsou MQTT servery. Mohou propojit velké objemy klientských aplikací MQTT a mohou vzájemně propojit sítě MQTT a IBM WebSphere MQ . Viz “Začínáme se servery MQTT” na stránce 135. Produkty IBM WebSphere MQ a IBM MessageSight mohou tvořit most mezi externími webovými aplikacemi, které jsou spuštěny na mobilních zařízeních a senzorech, a dalšími typy aplikací pro publikování/odběr a zasílání zpráv, které jsou spuštěny v rámci podniku. Most usnadňuje sestavení "inteligentních řešení" , která zahrnují mobilní zařízení a senzory.

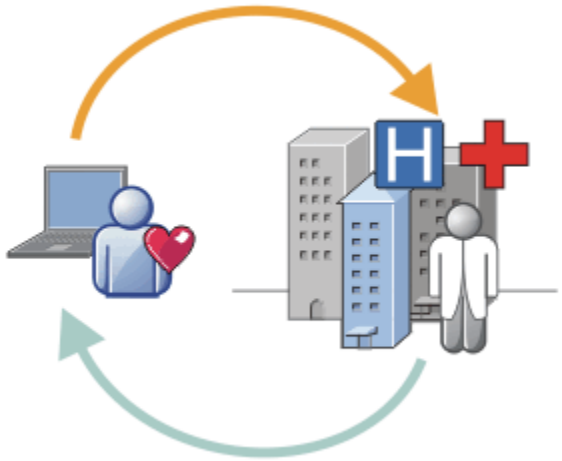
Chytrá řešení zpřístupní množství informací dostupných na internetu aplikacím běžícím na mobilních a senzorových zařízeních. Dva příklady chytrých aplikací, které jsou založeny na telemetrii, jsou chytrá elektřina a chytré zdravotní služby.



- Zpráva MQTT , která obsahuje data o využití energie odeslaná poskytovateli služeb.
- Aplikace telemetrie odesílá řídicí příkazy, které jsou založeny na analýze dat o využití energie.
- Další informace viz Scénář telemetrie: Monitorování a řízení domácí energie.

Obrázek 1. Inteligentní měření elektřiny

Obrázek 2. Inteligentní monitorování stavu



- Aplikace telemetrie odešle vaše zdravotní údaje do nemocnice a lékaře.
- Výstrahy nebo zpětnou vazbu zpráv produktu MQTT lze odeslat na základě analýzy vašich zdravotních údajů.
- Další informace viz Scénář telemetrie: Monitorování domácího pacienta.

Produkt MQTT můžete zabudovat do malých zařízení napsáním vlastní aplikace pro produkt MQTT protocol. Aby vám to pomohlo, produkt IBM poskytuje knihovny klienta, které podporují aplikace, které běží přes produkt MQTT. Viz téma “Začínáme s klienty MQTT” na stránce 10. Produkt IBM poskytuje knihovny klienta pro aplikace iOS a pro Android aplikace V 7.5.0.1 a JavaScript klienta prohlížeče pro webové aplikace agnostické platformy. V 7.5.0.1 JavaScript Stránky klienta se připojují k IBM MessageSight a IBM WebSphere MQ pomocí protokolu MQTT přes WebSockets. Produkt IBM také poskytuje MQTT ukázkové aplikace pro C a Java na systémech Linux® a Windows.

Knihovny C a Java jsou spuštěny na platformách iOS, Android, Windowsa UNIX and Linux . Zdrojový kód jazyka C pro knihovnu klienta MQTT můžete přenést na jiné platformy. Knihovny klienta MQTT pro C a Java

jsou k dispozici s licencí typu open source z projektu Eclipse Paho . Viz Eclipse Paho. Specifikace MQTT protocol je otevřená a je k dispozici v adresáři MQTT.org.

MQTT protocol

Produkt MQTT protocol je lehký v tom smyslu, že klienti jsou malí a efektivně využívají šířku pásma sítě. Protokol MQTT podporuje zaručené doručení a přenos typu fire-and-forget. V protokolu je doručení zprávy odděleno od aplikace. Rozsah oddělení v aplikaci závisí na způsobu zápisu klienta MQTT a serveru MQTT . Oddělené doručení uvolní aplikaci z libovolného připojení k serveru a z čekání na zprávy. Model interakce je podobný e-mailu, ale je optimalizován pro programování aplikací.

Protokol MQTT V3.1 je publikován; viz Specifikace protokolu MQTT V3.1 . Specifikace identifikuje řadu charakteristických rysů protokolu:

- Jedná se o protokol publikování/odběru.

  Kromě toho, že poskytuje distribuci zpráv typu jeden-na-mnoho, publikuje/odebírají aplikace s oddělením. Obě funkce jsou užitečné v aplikacích, které mají mnoho klientů.
- Není nijak závislá na obsahu zprávy.
- Běží přes protokol TCP/IP, který poskytuje základní síťovou konektivitu.
- Má tři kvality služby pro doručování zpráv:

  "Nejvýše jednou"
  Zprávy jsou doručovány v souladu s nejlepším úsilím základní sítě Internet Protocol . Může dojít ke ztrátě zprávy.

  Tuto kvalitu služby můžete využít například při komunikaci s údaji o okolním senzoru. Nezáleží na tom, zda je individuální čtení ztraceno, pokud je další publikováno brzy poté.

  "Nejméně jednou"
  Zprávy jsou zaručeny, že dorazí, ale může dojít k duplikátům.

  "Přesně jednou"
  Zprávy jsou zaručeny, že dorazí přesně jednou.

  Tuto kvalitu služeb použijte například s fakturačními systémy. Duplicitní nebo ztracené zprávy mohou vést k nepříjemnosti nebo k zavedení nesprávných poplatků.
- Je ekonomický ve způsobu, jakým spravuje tok zpráv v síti. Například záhlaví s pevnou délkou je dlouhé pouze 2 bajty a výměny protokolů jsou minimalizovány, aby se snížil provoz na síti.
- Má funkci "Poslední vůle a zákon" , která oznamuje odběratelům nestandardní odpojení klienta od serveru MQTT . Viz téma “Datum poslední vůle a potvrzení” na stránce 128.

Produkt MQTT version 3.1 je podporován produkty IBM IBM WebSphere MQ a IBM MessageSight. Produkt MQTT je implementován přes protokol TCP/IP. Pro sítě jiné než TCP/IP je k dispozici jiná verze protokolu MQTT-S. Viz Specifikace MQTT-S version 1.2.

MQTT komunity

Produkt IBM spouští produkt KomunitaIBM Developer Systém zpráv pro MQTT vývojáře, kteří píší aplikace pro IBM MessageSight a IBM WebSphere MQ.

Produkt MQTT.org je vhodným místem k tomu, abyste se dozvěděli více o implementacích a rozšířeních protokolu MQTT a diskutovali o nich.

MQTT je projekt typu open source Eclipse , pod položkou Eclipse Technology Project. Komunita Paho vyvíjí klienty a servery typu open source. Viz Eclipse Paho.

Úvod do produktu MQTT

Informace o odesílání zpráv mezi mobilními aplikacemi pomocí přenosu telemetrie MQ (MQTT). Protokol je určen pro použití v bezdrátových sítích a sítích s nízkou šířkou pásma. Mobilní aplikace používající produkt MQTT odesílá a přijímá zprávy voláním knihovny MQTT . Zprávy se vyměňují prostřednictvím

serveru systému zpráv MQTT . Klient a server MQTT zvládají složité aspekty spolehlivého doručování zpráv pro mobilní aplikaci a udržují nízké náklady na správu sítě.

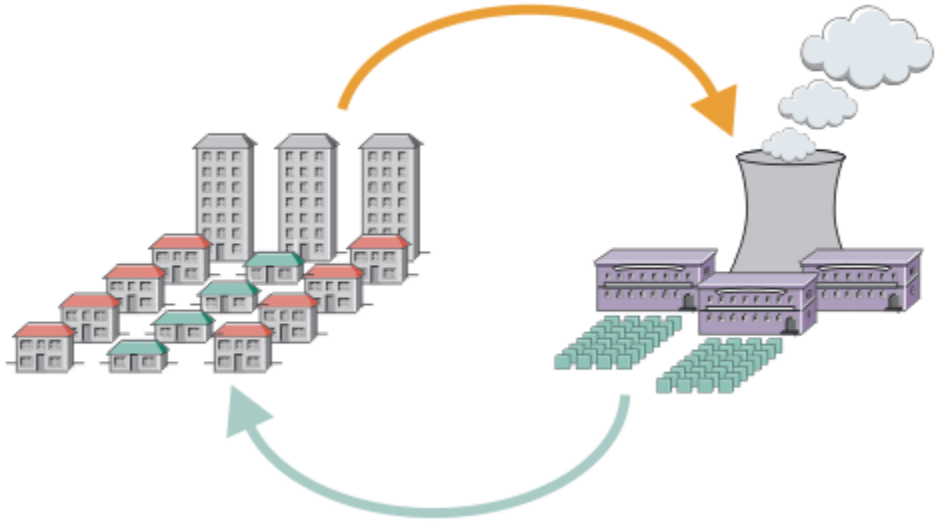
Aplikace MQTT běží na mobilních zařízeních, jako jsou chytré telefony a tablety. Produkt MQTT se také používá pro telemetrii pro příjem dat ze senzorů a pro jejich vzdálené řízení. Pro mobilní zařízení a senzory nabízí společnost MQTT vysoce škálovatelný protokol publikování/odběru se zajištěným doručením. Chcete-li odesílat a přijímat zprávy MQTT , přidejte do své aplikace knihovnu klienta MQTT .

Knihovna klienta MQTT je malá. Knihovna se chová jako poštovní schránka, odesílá a přijímá zprávy s jinými aplikacemi MQTT , které jsou připojeny k serveru MQTT . Posíláním zpráv namísto připojení k serveru, který čeká na odezvu, šetří aplikace MQTT životnost baterie. Knihovna odesílá zprávy na jiná zařízení prostřednictvím serveru MQTT , na kterém běží protokol MQTT version 3.1 . Můžete odesílat zprávy do specifického klienta nebo používat systém zpráv publikování/odběru pro připojení mnoha zařízení.

Knihovny klienta MQTT připojují aplikace pro mobilní zařízení a senzory k serveru MQTT pomocí protokolu MQTT .

IBM MessageSight a IBM WebSphere MQ jsou MQTT servery. Mohou propojit velké objemy klientských aplikací MQTT a mohou vzájemně propojit sítě MQTT a IBM WebSphere MQ . Viz “Začínáme se servery MQTT” na stránce 135. Produkty IBM WebSphere MQ a IBM MessageSight mohou tvořit most mezi externími webovými aplikacemi, které jsou spuštěny na mobilních zařízeních a senzorech, a dalšími typy aplikací pro publikování/odběr a zasílání zpráv, které jsou spuštěny v rámci podniku. Most usnadňuje sestavení "inteligentních řešení" , která zahrnují mobilní zařízení a senzory.

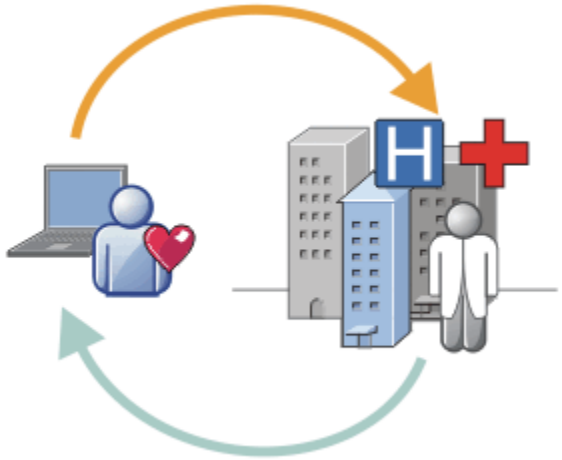
Chytrá řešení zpřístupní množství informací dostupných na internetu aplikacím běžícím na mobilních a senzorových zařízeních. Dva příklady chytrých aplikací, které jsou založeny na telemetrii, jsou chytrá elektřina a chytré zdravotní služby.



- Zpráva MQTT , která obsahuje data o využití energie odeslaná poskytovateli služeb.
- Aplikace telemetrie odesílá řídicí příkazy, které jsou založeny na analýze dat o využití energie.
- Další informace viz Scénář telemetrie: Monitorování a řízení domácí energie.

Obrázek 3. Inteligentní měření elektřiny

Obrázek 4. Inteligentní monitorování stavu



- Aplikace telemetrie odešle vaše zdravotní údaje do nemocnice a lékaře.
- Výstrahy nebo zpětnou vazbu zpráv produktu MQTT lze odeslat na základě analýzy vašich zdravotních údajů.
- Další informace viz Scénář telemetrie: Monitorování domácího pacienta.

Produkt MQTT můžete zabudovat do malých zařízení napsáním vlastní aplikace pro produkt MQTT protocol. Aby vám to pomohlo, produkt IBM poskytuje knihovny klienta, které podporují aplikace, které běží přes produkt MQTT. Viz téma “Začínáme s klienty MQTT” na stránce 10. Produkt IBM poskytuje knihovny klienta pro aplikace iOS a pro Android aplikace V 7.5.0.1 a JavaScript klienta prohlížeče pro webové aplikace agnostické platformy. V 7.5.0.1 JavaScript Stránky klienta se připojují k IBM MessageSight a IBM WebSphere MQ pomocí protokolu MQTT přes WebSockets. Produkt IBM také poskytuje MQTT ukázkové aplikace pro C a Java na systémech Linux a Windows.

Knihovny C a Java jsou spuštěny na platformách iOS, Android, Windowsa UNIX and Linux . Zdrojový kód jazyka C pro knihovnu klienta MQTT můžete přenést na jiné platformy. Knihovny klienta MQTT pro C a Java jsou k dispozici s licencí typu open source z projektu Eclipse Paho . Viz Eclipse Paho. Specifikace MQTT protocol je otevřená a je k dispozici v adresáři MQTT.org.

MQTT protocol

Produkt MQTT protocol je lehký v tom smyslu, že klienti jsou malí a efektivně využívají šířku pásma sítě. Protokol MQTT podporuje zaručené doručení a přenos typu fire-and-forget. V protokolu je doručení zprávy odděleno od aplikace. Rozsah oddělení v aplikaci závisí na způsobu zápisu klienta MQTT a serveru MQTT . Oddělené doručení uvolní aplikaci z libovolného připojení k serveru a z čekání na zprávy. Model interakce je podobný e-mailu, ale je optimalizován pro programování aplikací.

Protokol MQTT V3.1 je publikován; viz Specifikace protokolu MQTT V3.1 . Specifikace identifikuje řadu charakteristických rysů protokolu:

- Jedná se o protokol publikování/odběru.

  Kromě toho, že poskytuje distribuci zpráv typu jeden-na-mnoho, publikuje/odebírají aplikace s oddělením. Obě funkce jsou užitečné v aplikacích, které mají mnoho klientů.
- Není nijak závislá na obsahu zprávy.
- Běží přes protokol TCP/IP, který poskytuje základní síťovou konektivitu.
- Má tři kvality služby pro doručování zpráv:

  "Nejvýše jednou"
  Zprávy jsou doručovány v souladu s nejlepším úsilím základní sítě Internet Protocol . Může dojít ke ztrátě zprávy.

  Tuto kvalitu služby můžete využít například při komunikaci s údaji o okolním senzoru. Nezáleží na tom, zda je individuální čtení ztraceno, pokud je další publikováno brzy poté.

"Nejméně jednou"
Zprávy jsou zaručeny, že dorazí, ale může dojít k duplikátům.

"Přesně jednou"
Zprávy jsou zaručeny, že dorazí přesně jednou.

Tuto kvalitu služeb použijte například s fakturačními systémy. Duplicitní nebo ztracené zprávy mohou vést k nepříjemnosti nebo k zavedení nesprávných poplatků.

- Je ekonomický ve způsobu, jakým spravuje tok zpráv v síti. Například záhlaví s pevnou délkou je dlouhé pouze 2 bajty a výměny protokolů jsou minimalizovány, aby se snížil provoz na síti.
- Má funkci "Poslední vůle a zákon" , která oznamuje odběratelům nestandardní odpojení klienta od serveru MQTT . Viz téma “Datum poslední vůle a potvrzení” na stránce 128.

Produkt MQTT version 3.1 je podporován produkty IBM IBM WebSphere MQ a IBM MessageSight. Produkt MQTT je implementován přes protokol TCP/IP. Pro sítě jiné než TCP/IP je k dispozici jiná verze protokolu MQTT-S. Viz Specifikace MQTT-S version 1.2.

MQTT komunity

Produkt IBM spouští produkt KomunitaIBM Developer Systém zpráv pro MQTT vývojáře, kteří píší aplikace pro IBM MessageSight a IBM WebSphere MQ.

Produkt MQTT.org je vhodným místem k tomu, abyste se dozvěděli více o implementacích a rozšířeních protokolu MQTT a diskutovali o nich.

MQTT je projekt typu open source Eclipse , pod položkou Eclipse Technology Project. Komunita Paho vyvíjí klienty a servery typu open source. Viz Eclipse Paho.

Začínáme s klienty MQTT

Můžete začít vyvíjet mobilní aplikaci nebo aplikaci machine-to-machine (M2M) sestavením a spuštěním ukázkové aplikace klienta MQTT , která používá knihovnu klienta MQTT . Ukázkové aplikace a přidružené knihovny klienta jsou k dispozici v adresáři Mobile Messaging and M2M Client Pack na adrese IBM. Existují verze aplikací a klientských knihoven napsané v adresáři Java, v adresáři JavaScripta v adresáři C. Tyto aplikace můžete spouštět na většině platforem a zařízení, včetně zařízení a produktů Android z produktu Apple.

Než začnete

Chcete-li vytvořit a spustit aplikaci, potřebujete určité zkušenosti s vytvářením aplikací pro cílové zařízení nebo platformu a používaný programovací jazyk. Malá zkušenost je obvykle dost, aby se vzorová aplikace nahoru a běží na zvoleném zařízení nebo platformě.

Pokud používáte podnikový server MQTT , jako např. IBM WebSphere MQ nebo IBM MessageSight, můžete si vyměňovat informace z ukázkové aplikace s existujícími podnikovými aplikacemi.

Informace o této úloze

Vaše cíle jsou následující:

1. Zvolte server MQTT , ke kterému se můžete připojit k aplikaci klienta.
2. Stáhněte si Mobile Messaging and M2M Client Pack.
3. Sestavte pro cílové zařízení nebo platformu ukázkové aplikace z balíku klienta.
4. Ověřte, že se ukázky chovají podle očekávání, tak, že je připojíte k serveru MQTT .

V důsledku sestavení a testování ukázkových aplikací pro vaše zařízení nebo platformu vytvoříte pracovní vývojové prostředí, které pak můžete použít k sestavení vlastních klientských aplikací.

Soubor Mobile Messaging and M2M Client Pack obsahuje sadu SDK MQTT . Tato sada SDK vám poskytuje následující prostředky:

- Ukázkové MQTT klientské aplikace napsané v adresáři Java, v adresáři JavaScripta v adresáři C.
- Knihovny klienta MQTT , které podporují tyto klientské aplikace, a umožňují jim spouštět je na většině platforem a zařízení.

Sada SDK také obsahuje zdrojový kód pro Klient MQTT pro C. Tento zdrojový kód můžete upravit tak, aby sestavoval knihovny klienta MQTT pro C pro jiné platformy. Nápovědu k tomuto tématu naleznete v části “Sestavení klienta MQTT pro knihovny C” na stránce 30. Zdrojový kód pro produkt Klient MQTT pro C je také k dispozici s licencí typu open source od společnosti Eclipse Paho.

Procedura

Následující články vás provedou kroky specifickými pro platformu pro sestavení a spuštění ukázkové aplikace MQTT na stolním počítači nebo na mobilním zařízení pro produkt Android nebo z produktu Apple:

- “Začínáme s klientem produktu MQTT pro produkt Java” na stránce 11
- “Začínáme s produktem Klient MQTT pro produkt Java on Android” na stránce 17
- V 7.5.0.1 “Začínáme s produktem Klient systému zpráv MQTT pro produkt JavaScript” na stránce 23
- “Začínáme s klientem MQTT pro jazyk C” na stránce 25
- “Začínáme s klientem MQTT pro C na systému iOS” na stránce 47

Jak pokračovat dále

Chcete-li vyvinout novou aplikaci MQTT , musíte mít nebo získat následující dovednosti:

- Programování v jazyce, který je vyžadován pro zařízení nebo platformu.
- Programování pro cílové zařízení nebo platformu.
- Návrh aplikací publikování/odběru.
- Návrh programů pro programovací model MQTT .
- Návrh programů pro spuštění na zvoleném mobilním zařízení.
- Použití SSL a JAAS k zabezpečení programů.

Pro připojení klienta MQTT k jinému zařízení nebo aplikaci nepotřebujete žádné znalosti programování v síti, protože MQTT je systém pro zasílání zpráv a systém front. Knihovny klienta MQTT spravují síťová připojení pro vaši aplikaci.

Chcete-li integrovat klienta MQTT s existujícími podnikovými aplikacemi, máte dvě možnosti. Témata MQTT publikování/odběru můžete sdílet například s aplikací IBM WebSphere MQ nebo JMS , nebo můžete napsat svůj vlastní integrační adaptér jako jiného klienta MQTT .

Zdroje informací, které je třeba dnes konzultovat, jsou:

- Vyvíjení aplikací pro WebSphere MQ Telemetry
- MQTT.org
- Eclipse Paho

Související pojmy

“Začínáme se servery MQTT” na stránce 135

Začínáme s klientem produktu MQTT pro produkt Java

s ukázkovými aplikacemi produktu Java pro ukázkové aplikace produktu MQTT s použitím produktu buď IBM MessageSight , nebo IBM WebSphere MQ jako server MQTT. Ukázkové aplikace používají knihovnu klienta ze sady nástrojů pro vývoj softwaru MQTT (SDK) od společnosti IBM. Ukázková aplikace produktu `SampleAsyncCallBack` je modelem pro zápis aplikací produktu MQTT pro operační systém Android a dalších operačních systémů založených na události.

Než začnete

- Aplikaci Klient MQTT pro produkt Java můžete spustit na libovolné platformě s JSE 1.5 nebo vyšším, které je "Java Kompatibilní". Viz téma Systémové požadavky pro IBM Mobile Messaging and M2M Client Pack.
- Je-li mezi vaším klientem a serverem brána firewall, zkontrolujte, zda neblokuje provoz produktu MQTT .

Informace o této úloze

Účelem této úlohy je zkontrolovat, zda můžete sestavit a spustit klienta MQTT pro ukázkovou aplikaci Java , připojit jej k produktu IBM WebSphere MQ nebo IBM MessageSight jako server MQTT version 3 a vyměňovat si zprávy.

Pomocí této úlohy spusťte ukázkovou aplikaci z pracovní plochy produktu Eclipse nebo z příkazového řádku. Kroky v tomto příkladu jsou určeny pro Windows. S malými úpravami můžete spustit ukázkovou aplikaci na libovolné platformě, která podporuje produkt JSE 1.5 nebo vyšší.

Aplikace můžete spouštět na stejném serveru jako produkt IBM WebSphere MQ, kde je pro vás nakonfigurováno prostředí pro spouštění aplikací, které se připojují k produktu IBM WebSphere MQ . Postupujte podle úlohy “Konfigurace služby MQTT z příkazového řádku” na stránce 139 , chcete-li instalovat a konfigurovat IBM WebSphere MQ s volbou IBM WebSphere MQ Telemetry na Windows nebo Linux. Když je prostředí nainstalováno a nakonfigurováno, spusťte ukázkovou aplikaci, `MQTTV3Sample`, abyste ověřili instalaci.

Postup

1. Zvolte server MQTT , ke kterému se můžete připojit k aplikaci klienta.

   Server musí podporovat protokol MQTT version 3.1 . Všechny servery MQTT z IBM to dělají, včetně IBM WebSphere MQ a IBM MessageSight. Viz “Začínáme se servery MQTT” na stránce 135.
2. Volitelné: Nakonfigurujte server MQTT .
   - V systému IBM WebSphere MQje třeba při nastavení správce front a konfiguraci služby telemetrie (MQXR) dokončit jednu či více následujících úloh:
     - “Konfigurace služby MQTT z příkazového řádku” na stránce 139
     - “Konfigurace služby MQTT s produktem IBM WebSphere MQ Explorer” na stránce 141
   - Na jiných serverech se poraďte s dokumentací k serveru. Pro modul Really Small Message Brokernejsou vyžadovány žádné kroky konfigurace. Viz Really Small Message Broker.
3. Stáhněte si produkt Mobile Messaging and M2M Client Pack a nainstalujte sadu SDK produktu MQTT .

   K dispozici není žádný instalační program, stačí jen rozbalit stažený soubor.

   a. Stáhněte si Mobile Messaging and M2M Client Pack.

   b. Vytvořte složku, do které budete instalovat sadu SDK.

      Je možné, že budete chtít pojmenovat složku `MQTT`. Cesta k této službě je zde označována jako *`sdkroot`*.

   c. Rozbalte komprimovaný obsah souboru Mobile Messaging and M2M Client Pack do *`sdkroot`*. Rozšíření vytváří adresářový strom, který začíná v *`sdkroot`*\SDK.
4. Nainstalujte vývojovou sadu produktu Java (JDK) verze 6 nebo novější.

   Protože vyvíjíte aplikaci Java for Android, musí produkt JDK pocházet z produktu Oracle. Sadu JDK lze získat z Soubory ke stažení Java SE.
5. Kompilujte a spusťte jeden nebo více z klienta produktu MQTT pro ukázkové aplikace produktu Java :
   - “Kompilace a spuštění ukázkových programů Paho z příkazového řádku” na stránce 13
   - “Kompilujte a spusťte všechny ukázkové aplikace Java klienta produktu MQTT z produktu Eclipse” na stránce 15
   - “Začínáme s produktem Klient MQTT pro produkt Java on Android” na stránce 17

The following MQTT client sample Java apps are included in the SDK:

MQTTV3Sample
Ukázka je také obsažena v produktu IBM WebSphere MQ a odkazuje na balík produktu `com.ibm.micro.client.mqttv3.jar` .

Sample
`Sample` se nachází v balíku Paho a odkazuje na balík `org.eclipse.paho.client.mqttv3` . Podobá se příkazu `MQTTV3Sample`; čeká, dokud nebude dokončena každá akce MQTT .

SampleAsyncWait
`SampleAsyncWait` je v balíku `org.eclipse.paho.client.mqttv3`. Používá asynchronní rozhraní MQTT API; čeká na jiný podproces, dokud není akce dokončena. Hlavní podproces může provádět jinou práci, dokud se nesynchronizuje s podprocesem, který čeká na dokončení akce MQTT.

SampleAsyncCallBack
`SampleAsyncCallBack` je v balíku `org.eclipse.paho.client.mqttv3`. Volá asynchronní rozhraní MQTT API. Asynchronní rozhraní API nečeká, až MQTT dokončí zpracování volání, vrátí se do aplikace. Aplikace pokračuje s ostatními úlohami, poté čeká na další událost, aby mohla být zpracována. MQTT odešle oznámení o události zpět do aplikace po dokončení zpracování. Rozhraní MQTT řízené událostmi je vhodné pro programovací model služeb a aktivit Android a dalších operačních systémů řízených událostmi.

Jako příklad se podívejte, jak ukázka `mqttExerciser` integruje MQTT do Android pomocí programovacího modulu služeb a aktivit.

mqttExerciser
`mqttExerciser` je ukázkový program pro Android. Protože je sestaven a spuštěn jinak, je popsán samostatně. Viz “Začínáme s produktem Klient MQTT pro produkt Java on Android” na stránce 17.

Výsledky

Kompilovali jste a spustili ukázkové aplikace produktu MQTT Java , které jsou připojeny k produktu IBM WebSphere MQ nebo IBM MessageSight jako server MQTT .

Jak pokračovat dále

Prostudujte si referenční informace Javadoc , viz krok “3” na stránce 16 z “Kompilujte a spusťte všechny ukázkové aplikace Java klienta produktu MQTT z produktu Eclipse” na stránce 15. Případně otevřete soubory HTML dokumentace Javadoc , které se nacházejí v adresáři SDK`\clients\java\doc\javadoc` v produktu Mobile Messaging and M2M Client Pack.

Kompilace a spuštění ukázkových programů Paho z příkazového řádku

Kompilujte a spusťte ukázkovou aplikaci Paho `Sample.java` z příkazového řádku. Ukázka je uvedena v sadě SDK produktu MQTT . Ukázka je sestavena pomocí knihoven klienta produktu MQTT Paho v balíku produktu `org.eclipse.paho.client.mqttv3` . Demonstruje vydavatele a odběratele produktu MQTT . Dva další ukázky Paho ve stejném adresáři mohou být sestaveny a spouštěny stejným způsobem. Liší se tím, že asynchronně volají knihovnu MQTT .

Než začnete

Kroky “1” na stránce 12 až “4” na stránce 12 v hlavní úloze provedete tak, že nakonfigurujete server MQTT a stáhnete Mobile Messaging and M2M Client Pack.

Informace o této úloze

Kompilujte a spusťte `Sample.java` z podadresáře ukázek klienta SDK SDK`\clients\java\samples`. Kód Java se nachází v adresáři SDK`\clients\java\samples\org\eclipse\paho\sample\mqttv3app`.

Postup

1. Vytvořte skript v adresáři ukázek klienta, abyste kompilovali a spustili `Sample` na zvolené platformě.

   Následující skript zkompiluje a spustí ukázku v systému Windows.

```
@echo on
setlocal
set classpath=
set JAVADIR=C:\Program Files\IBM\Java70\bin
"%JAVADIR%\javac"
-cp ..\org.eclipse.paho.client.mqttv3.jar .\org\eclipse\paho\sample\mqttv3app\Sample.java
start "Sample Subscriber" "%JAVADIR%\java" -cp  .;..\org.eclipse.paho.client.mqttv3.jar
org.eclipse.paho.sample.mqttv3app.Sample -a subscribe -b localhost -p 1883
@rem Sleep for 2 seconds
ping -n 2 127.0.0.1 > NUL 2>&1
"%JAVADIR%\java" -cp  .;..\org.eclipse.paho.client.mqttv3.jar
org.eclipse.paho.sample.mqttv3app.Sample -b localhost -p 1883
pause
endlocal
```

   Obrázek 5. Kompilace a spuštění produktu `Sample.java`

2. Spusťte skript.

   Výsledky:

```
Connected to tcp://localhost:1883 with client ID SampleJavaV3_subscribe
Subscribing to topic "Sample/#" qos 2
Press <Enter> to exit
Time:   2012-11-09 17:23:22.718  Topic: Sample/Java/v3  Message:        Message
from blocking MQTTv3 Java client sample  QoS:   2
```

   Obrázek 6. `MQTTV3Sample` Odběratel

```
Connecting to tcp://localhost:1883 with client ID SampleJavaV3_publish
Connected
Publishing at: 2012-11-09 17:22:07.734 to topic "Sample/Java/v3" qos 2
Disconnected
```

   Obrázek 7. `MQTTV3Sample` Vydavatel

   Pokud ponecháte spuštěnou aplikaci odběratele, nemůžete znovu spustit stejného odběratele. Identifikátor klienta nového odběratele je stejný jako u starého odběratele. Ve stejnou dobu nemůžete spustit dva klienty MQTT se stejným identifikátorem klienta. Můžete nastavit volbu `-i` k nastavení identifikátoru klienta tak, abyste mohli současně spouštět různé odběratele.

   Spustíte-li stejného klienta znovu, můžete využít volbu `-c` ke spuštění a spuštění klienta s parametrem `cleansession` nastaveným na hodnotu false. S touto volbou můžete prozkoumávat chování přerušených relací klienta.

3. Ukončete odběratele stisknutím klávesy Enter nebo zavřením okna.

Jak pokračovat dále

Vytvořte skripty pro kompilaci a spuštění dalších ukázek v podadresáři `SDK\clients\java\samples\org\eclipse\paho\sample\mqttv3app` . Zkopírujte skript v produktu Obrázek 5 na stránce 14 a nahraďte položku `Sample` řetězcem `SampleAsyncWait` nebo `SampleAsyncCallBack`. Další vzorky jsou asynchronní verze synchronního programu `Sample` .

`SampleAsyncWait`
`SampleAsyncWait` je v balíku `org.eclipse.paho.client.mqttv3`. Používá asynchronní rozhraní MQTT API; čeká na jiný podproces, dokud není akce dokončena. Hlavní podproces může provádět jinou práci, dokud se nesynchronizuje s podprocesem, který čeká na dokončení akce MQTT.

`SampleAsyncCallBack`
`SampleAsyncCallBack` je v balíku `org.eclipse.paho.client.mqttv3`. Volá asynchronní rozhraní MQTT API. Asynchronní rozhraní API nečeká, až MQTT dokončí zpracování volání, vrátí se do aplikace. Aplikace pokračuje s ostatními úlohami, poté čeká na další událost, aby mohla být

zpracována. MQTT odešle oznámení o události zpět do aplikace po dokončení zpracování. Rozhraní MQTT řízené událostmi je vhodné pro programovací model služeb a aktivit Android a dalších operačních systémů řízených událostmi.

Jako příklad se podívejte, jak ukázka `mqttExerciser` integruje MQTT do Android pomocí programovacího modulu služeb a aktivit.

Asynchronní ukázky demonstrují způsob snížení množství času, který aplikační bloky produktu MQTT během čekání na klienta MQTT čekají. Je důležité odstranit blokování hovorů v hlavním vláknu pro zvýšení responsivity a bateriového života v mobilním prostředí.

Ukázky demonstrují dva vzory pro volání asynchronních rozhraní v klientovi MQTT .

1. Produkt `SampleAsyncWait` není blokovat, zatímco MQTT čeká na síťové interakce.
2. Produkt `SampleAsyncCallBack` neblokuje čekání na dokončení žádných akcí klienta produktu MQTT . To druhé je nezbytné, když stránka JavaScript volá klienta MQTT z prohlížeče. Stránky JavaScript nesmí blokovat. Odezvy na akce musí být odeslány zpět do hlavního podprocesu prohlížeče, který volá obslužnou rutinu událostí produktu MQTT , kterou napíšete, aby zpracovali oznámení.

Kompilujte a spusťte všechny ukázkové aplikace Java klienta produktu MQTT z produktu Eclipse

Kompilujte a spusťte ukázkové aplikace Java klienta MQTT , které jsou v Mobile Messaging and M2M Client Pack. Vykazují vydavatele a odběratele produktu MQTT .

Informace o této úloze

Kompilujte a spusťte ukázky MQTT Java , `Samplea` `MQTTV3Sample` v Eclipse. `Sample` je v podadresáři klientů *sdkroot*`\SDK\clients\java\samples\org\eclipse\paho\sample\mqttv3app` SDK a `MQTTV3Sample.java` je v *sdkroot*`\SDK\clients\java\samples`.

Postup

1. Stáhněte Eclipse IDE for Java Developers.
2. Vytvořte projekt Java s názvem `MQTT Samples` v produktu Eclipse.
   a) **Soubor** > **Nový** > **Projekt Java** a zadejte `MQTT Samples`. Klepněte na tlačítko **Další**.

      Zkontrolujte, zda je prostředí JRE správné nebo pozdější verze.JSE musí být na verzi 1.5 nebo pozdější.

   b) V okně **Nastavení prostředí Java** klepněte na volbu **Propojit další zdrojové složky**.

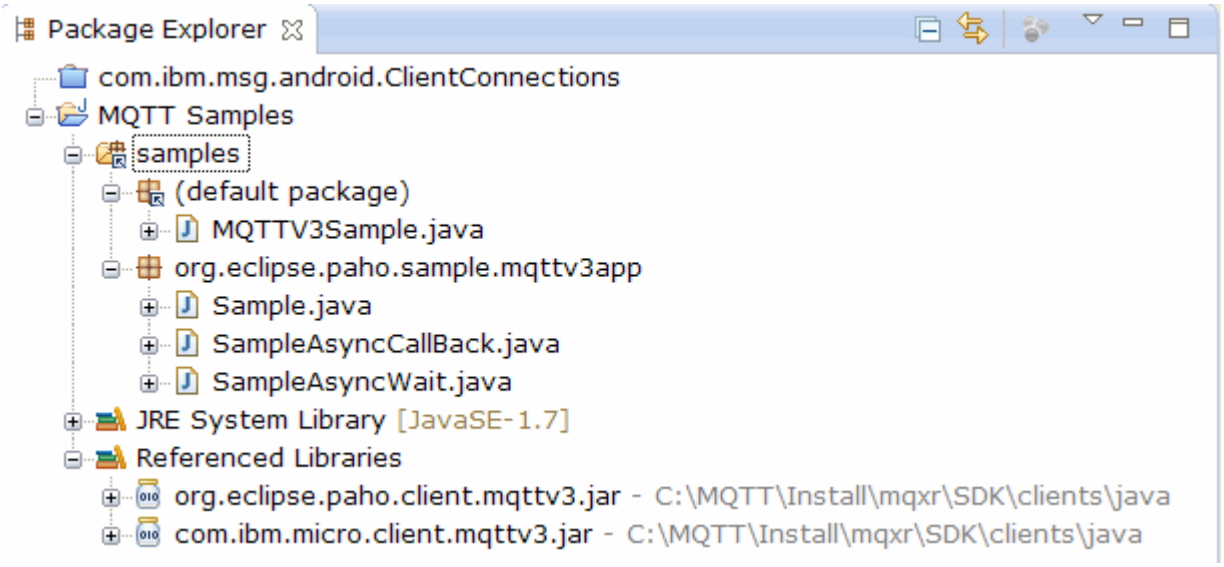
   c) Přejděte do adresáře, do kterého jste nainstalovali složku sady SDK produktu MQTT Java . Vyberte složku *sdkroot*`\SDK\clients\java\samples` a klepněte na tlačítko **OK** > **Další** > **Dokončit**.

   d) V okně **Nastavení prostředí Java** klepněte na volbu **Knihovny** > **Přidat externí soubory JAR** .

   e) Přejděte do adresáře, do kterého jste nainstalovali složku sady SDK produktu MQTT Java . Vyhledejte složku *sdkroot*`\SDK\clients\java` a vyberte soubory `org.eclipse.paho.client.mqttv3.jar` a `com.ibm.micro.client.mqttv3.jar` ; klepněte na volbu **Otevřít** > **Dokončit**.

      Ukázka `MQTTV3Sample.java` odkazuje na `com.ibm.micro.client.mqttv3.jar`a ukázky v adresářovém stromu paho se odkazují na `org.eclipse.paho.client.mqttv3.jar`. Produkt `com.ibm.micro.client.mqttv3.jar` je zachován, takže existující aplikace produktu MQTT budou pokračovat v sestavení a spuštění beze změn.

   Projekt produktu `MQTT Samples` je sestaven s varováními, ale bez chyb.



Obrázek 8. Klient produktu MQTT pro projekt Java

3. Volitelné: Nainstalujte klienta MQTT Javadoc.

   Při instalaci klienta MQTT Javadoc , editor Java popisuje třídy MQTT v bublinové nápovědě.

   a) Otevřete **Průzkumník balíků** > **Odkazované knihovny** ve svém projektu Java. Klepněte pravým tlačítkem myši na položku `org.eclipse.paho.client.mqttv3.jar` **> Vlastnosti**.

   b) V navigátoru Vlastnosti klepněte na volbu **Umístění dokumentace Javadoc**.

   c) Klepněte na volbu **Adresa URL dokumentace Javadoc** > **Procházet** na stránce **Umístění dokumentace Javadoc** a vyhledejte složku `SDK\clients\java\doc\javadoc` **> OK**.

   d) Klepněte na volbu **Ověřit** > **OK** .

   Zobrazí se výzva k otevření prohlížeče a zobrazení dokumentace.

   e) Zopakujte tuto proceduru pro soubor `com.ibm.micro.client.mqttv3.jar` .

4. Vytvořte konfiguraci běhového prostředí vydavatele a odběratele, abyste mohli spustit aplikaci `mqttv3app.Sample` .

   a) Klepněte pravým tlačítkem myši na třídu **Ukázka** , klepněte na volbu **Spustit jako** > **Spustit konfigurace**.

   b) Klepněte pravým tlačítkem myši na **Aplikace Java** > **Nová** a zadejte název `SampleSubscriber`.

   c) Klepněte na kartu argumentů a zadejte argumenty programu následované parametrem **Apply**.

   ```
   -a subscribe -b localhost -p 1883
   ```

   d) Opakováním posledního kroku vytvořte konfiguraci produktu `SamplePublisher` vynecháním parametru `-a subscribe` .

5. Spusťte odběratele produktu `mqttv3app.Sample` následovaného vydavatelem.

   a) Klepněte na nabídku **Spustit** > **Spustit konfigurace** .

   b) Klepněte na volbu **SampleSubscriber** > **Spustit**.

   Otevřete pohled **Konzola** . Odběratel čeká na publikování.

   ```
   Connected to tcp://localhost:1883 with client ID SampleJavaV3_subscribe
   Subscribing to topic "Sample/#" qos 2
   Press <Enter> to exit
   ```

   c) Klepněte na volbu **SamplePublisher** > **Spustit**.

Otevřete pohled **Konzola** . Zobrazí se publikování, které je vytvořeno vydavatelem:

```
Connecting to tcp://localhost:1883 with client ID SampleJavaV3_publish
Connected
Publishing at: 2012-11-09 14:09:29.859 to topic "Sample/Java/v3" qos 2
Disconnected
```

d) Přepnout zobrazení konzoly na konzolu odběratele.

Ikona konzol přepínačů je  .

Odběratel obdržel publikování:

```
...
Time: 2012-11-09 14:09:30.593  Topic: Sample/Java/v3  Message: Message from blocking
MQTTv3 Java client sample  QoS: 2
```

6. Volitelné: Vytvořte konfiguraci běhového prostředí vydavatele a odběratele pro produkt `MQTTV3Sample`.
   a) Klepněte pravým tlačítkem myši na třídu **MQTTV3Sample** , klepněte na nabídku **Spustit jako** > **Spustit konfigurace**.
   b) Klepněte pravým tlačítkem myši na **Aplikace Java** > **Nová** a zadejte název `MQTTV3SampleSubscriber`.
   c) Klepněte na kartu argumentů a zadejte argumenty programu následované parametrem **Apply**.

```
-a subscribe -b localhost -p 1883
```

   d) Opakováním posledního kroku vytvořte konfiguraci produktu `MQTTV3SamplePublisher` vynecháním parametru `-a subscribe` .
7. Volitelné: Spusťte odběratele produktu `MQTTV3Sample` následovaného vydavatelem.
   a) Klepněte na nabídku **Spustit** > **Spustit konfigurace** .
   b) Klepněte na volbu **MQTTV3SampleSubscriber** > **Spustit**.

   Otevřete pohled **Konzola** . Odběratel čeká na publikování.

```
Connected to tcp://localhost:1883
Subscribing to topic "MQTTV3Sample/#" qos 2
Press <Enter> to exit
```

   c) Klepněte na volbu **MQTTV3SamplePublisher** > **Spustit**.

   Otevřete pohled **Konzola** . Můžete si prohlédnout publikaci, která je vytvořena vydavatelem.

```
Connected to tcp://localhost:1883
Publishing to topic "MQTTV3Sample/Java/v3" qos 2
Disconnected
```

   d) Přepnout zobrazení konzoly na konzolu odběratele.

Ikona konzol přepínačů je  .

Odběratel obdržel publikování:

```
MQTTV3Sample/Java/v3
Message: Message from MQTTv3 Java client
QoS:  2
```

Začínáme s produktem Klient MQTT pro produkt Java on Android

Můžete nainstalovat produkt Ukázková aplikace klienta MQTT Java pro produkt Android , který bude vyměňovat zprávy se serverem MQTT . Aplikace používá knihovnu klienta ze sady MQTT SDK z produktu IBM. Můžete buď sestavit aplikaci sami, nebo stáhnout předem sestavenou ukázkovou aplikaci.

Než začnete

- Podporované a referenční platformy klienta MQTT viz Systémové požadavky pro IBM Mobile Messaging and M2M Client Pack.
- Je-li mezi vaším klientem a serverem brána firewall, zkontrolujte, zda neblokuje provoz produktu MQTT .
- Ukázková aplikace klienta MQTT funguje na systému Ice Cream Sandwich (Android 4.0) a vyšší. Tato verze produktu Android také poskytuje ostřejší rozlišení displeje na tabletech.

Informace o této úloze

Ukázková aplikace klienta MQTT Java pro produkt Android se nazývá "`mqttExerciser`". Tato aplikace používá knihovnu klienta ze sady SDK MQTT a vyměňuje zprávy se serverem MQTT .

Ukázkovou aplikaci můžete buď sami sestavit a poté ji exportovat z Eclipse jako `mqttExerciser.apk`, nebo můžete použít předem sestavenou ukázkovou aplikaci, která je k dispozici jako soubor `mqttExerciser.apk` ve složce *sdkroot*`\SDK\clients\android\samples\apks` produktu Mobile Messaging and M2M Client Pack. Pokud se rozhodnete sestavit aplikaci sami, vývojové prostředí, které sestavíte, je přizpůsobeno tak, aby zahrnovalo mobilní zasílání zpráv do aplikací for Android. To by vám mělo pomoci, když začnete začleňovat mobilní zasílání zpráv do svých vlastních aplikací.

Postup

1. Zvolte server MQTT , ke kterému se můžete připojit k aplikaci klienta.

   Server musí podporovat protokol MQTT version 3.1 . Všechny servery MQTT z IBM to dělají, včetně IBM WebSphere MQ a IBM MessageSight. Viz “Začínáme se servery MQTT” na stránce 135.
2. Získejte správné nástroje.

   Nainstalujte vývojovou sadu produktu Java (JDK) verze 6 nebo novější. Protože vyvíjíte aplikaci Java for Android, musí produkt JDK pocházet z produktu Oracle. Sadu JDK lze získat z Soubory ke stažení Java SE.

   Také potřebujete vývojové prostředí Eclipse . Musí být Eclipse 3.6.2 (Helios) nebo vyšší. Eclipse musí mít úroveň kompilátoru Java alespoň 6, aby se shodovala s vaší sadou JDK. To vše můžete získat z Eclipse Foundation.

   Nakonec potřebujete sadu SDK Android . Můžete to získat z Získat sadu Android SDK.
3. Stáhněte si produkt Mobile Messaging and M2M Client Pack a nainstalujte sadu SDK produktu MQTT .

   K dispozici není žádný instalační program, stačí jen rozbalit stažený soubor.

   a. Stáhněte si Mobile Messaging and M2M Client Pack.

   b. Vytvořte složku, do které budete instalovat sadu SDK.

      Je možné, že budete chtít pojmenovat složku `MQTT`. Cesta k této službě je zde označována jako *sdkroot*.

   c. Rozbalte komprimovaný obsah souboru Mobile Messaging and M2M Client Pack do *sdkroot*. Rozšíření vytváří adresářový strom, který začíná v *sdkroot*`\SDK`.
4. Volitelné: Sestavte ukázkovou aplikaci mqttExerciser for Android.

   Nakonfigurujte nástroje Eclipse a Android a naimportujte a sestavte projekt `mqttExerciser` ze sady MQTT SDK.

   Poznámka: Pokud to nyní nechcete provést, můžete použít předem sestavenou ukázkovou aplikaci, která je k dispozici jako soubor `mqttExerciser.apk` ve složce *sdkroot*`\SDK\clients\android\samples\apks` sady MQTT SDK.

   a) Spusťte vývojové prostředí Eclipse s prostředím JRE ze sady JDK.

   ```
   eclipse -vm "JRE path"
   ```

   b) Vyberte a nainstalujte sadu balíků a platforem ze sady SDK Android .

Seznam platforem a balíků, které společnost Google doporučuje, naleznete v tématu Přidávání platforem a balíků .

Poznámka: Platforma SDK musí být Android API úrovně 16 nebo novější. S dřívějšími úrovněmi rozhraní API nelze projekt úspěšně zkompilovat.

c) Přidejte modul plug-in Android Vývojové nástroje (ADT) do souboru Eclipse.

d) Naimportujte ukázkový projekt aplikace `mqttExerciser` do adresáře Eclipsea opravte chyby.

i) Naimportujte ukázkový projekt aplikace ze sady MQTT SDK v cestě *sdkroot*`\SDK\clients\android\samples\mqttExerciser`.

Pohled **Problémy** vypisuje mnoho chyb sestavení. Chyby sestavení vyřešíte v několika dalších krocích.

ii) Zkopírujte knihovnu `org.eclipse.paho.client.mqttv3.jar` do složky **libs** v projektu Android . Windows Například v systému Windowsse nachází ve složce *sdkroot*`\SDK\clients\java` . Zobrazí se okno **Operace souboru** . Přijměte výběr volby **Kopírovat soubory** a poté klepněte na tlačítko **OK**.

iii) Klepněte pravým tlačítkem myši na složku projektu `com.ibm.msg.android`; klepněte na volbu **Nástroje systému Android ...** > **Přidat knihovnu podpory ...**. Přečtěte si a přijměte licenční podmínky a poté klepněte na tlačítko **Instalovat**.

iv) Klepněte pravým tlačítkem myši na složku projektu `com.ibm.msg.android`; klepněte na volbu **Nástroje systému Android ...** > **Vlastnosti projektu opravy**.

v) Pokud pracovní prostor stále obsahuje asi 84 chyb, což odkazuje na přepsání metody supertřídy, úroveň shody kompilátoru je pravděpodobně nastavena na 1.5 nebo nižší. Android SDK verze 16 očekává, že úroveň shody kompilátoru nebude vyšší než 1.5. Chcete-li opravit zbývající chyby, postupujte takto:

a) Zkontrolujte a (v případě potřeby) aktualizujte sadu SDK Android a příslušné moduly plug-in Eclipse na Android SDK verze 17.

b) Klepněte pravým tlačítkem myši na složku projektu **com.ibm.msg.android** a vyberte volbu **Vlastnosti** > **Kompilátor Java**. Zkontrolujte úroveň shody kompilátoru, nastavte ji na alespoň 1.6, pak znovu sestavte pracovní prostor.

Projekt se sestaví s několika varováními a bez chyb.

5. Nainstalujte a spusťte soubor Ukázková aplikace klienta MQTT Java na zařízení Android .

Viz `developer.android.com` stránka Spuštění aplikace.

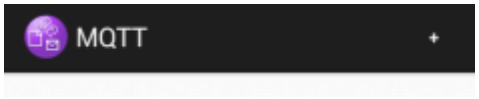
Pokud jste aplikaci sestavili sami jako projekt Eclipse , můžete spustit aplikaci z Eclipse.

Máte-li soubor balíku aplikace (APK) `mqttExerciser.apk`, můžete jej nainstalovat mimo platformu Eclipse pomocí příkazu instalace Android Debug Bridge (ADB). Tento příkaz vezme umístění souboru APK jako argument. Pokud používáte předem sestavenou ukázkovou aplikaci, umístění je *sdkroot*`\SDK\clients\android\samples\apks\mqttExerciser.apk`.

6. Pomocí ukázkové aplikace mqttExerciser for Android se můžete připojit, přihlásit k odběru a publikovat v tématu.

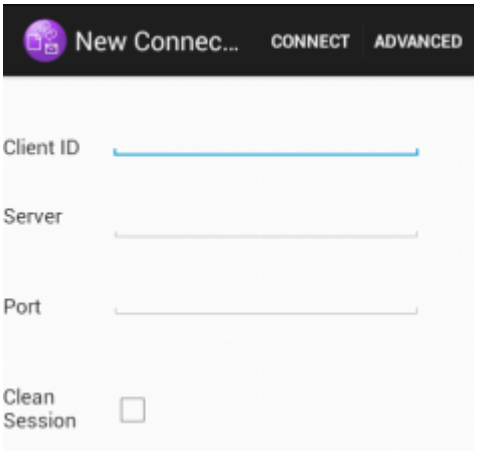
a) Otevřete produkt Ukázková aplikace klienta MQTT Java pro produkt Android.

Toto okno je otevřeno ve vašem zařízení Android :



b) Připojte se k serveru MQTT .

i) Klepnutím na symbol **+** otevřete nové připojení MQTT .



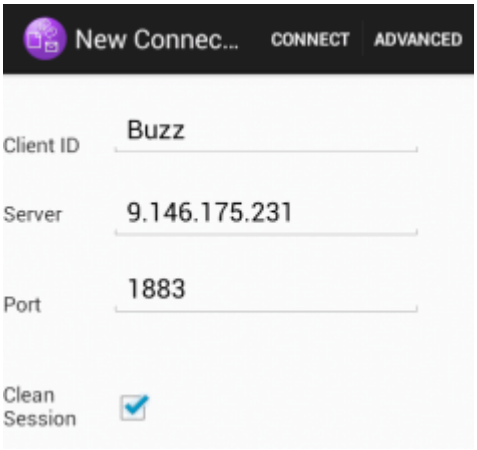
ii) Do pole **ID klienta** zadejte libovolný jedinečný identifikátor. Buďte trpěliví, úhozy mohou být pomalé.

iii) Zadejte do pole **Server** adresu IP vašeho serveru MQTT .

Jedná se o server, který jste vybrali v prvním hlavním kroku. Adresa IP nesmí být `127.0.0.1`

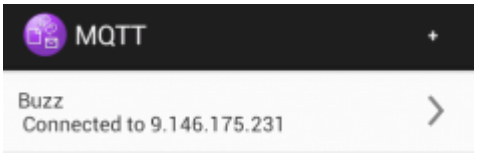
iv) Zadejte číslo portu připojení MQTT .

Výchozí číslo portu pro normální MQTT připojení je 1883.



v) Klepněte na tlačítko **Připojit**.

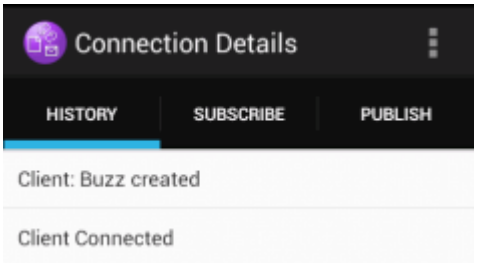
Je-li připojení úspěšné, zobrazí se zpráva "Connecting" následované tímto oknem:



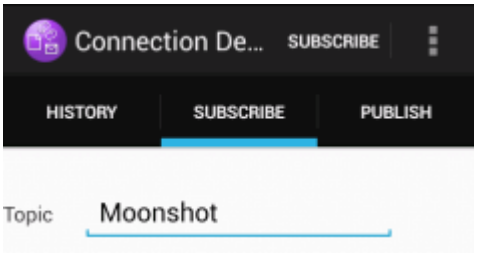
c) Přihlaste se k odběru tématu.

i) Klepněte na zprávu **Připojeno** .

Otevře se okno **Podrobnosti připojení** s vypsanou historií:



ii) Klepněte na kartu **Odebírat** a zadejte řetězec tématu.

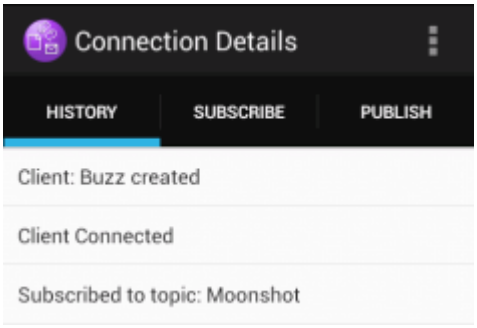


iii) Klepněte na akci **Odebírat** .

Na krátkou dobu se zobrazí zpráva "Odebírané" .

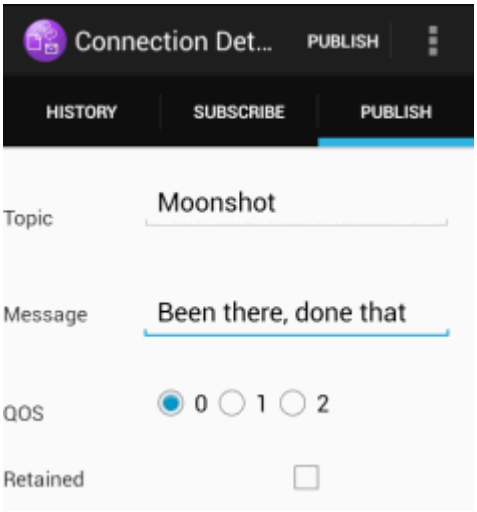
iv) Klepněte na kartu **Historie** .

Historie nyní zahrnuje odběr:



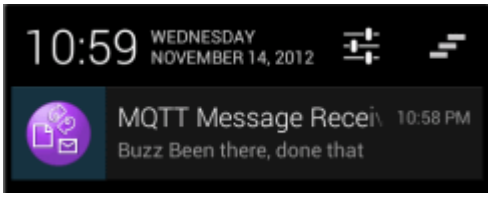
d) Nyní publikujte do stejného tématu.

i) Klepněte na kartu **Publikovat** a zadejte stejný řetězec tématu jako pro přihlášení k odběru. Zadejte zprávu.

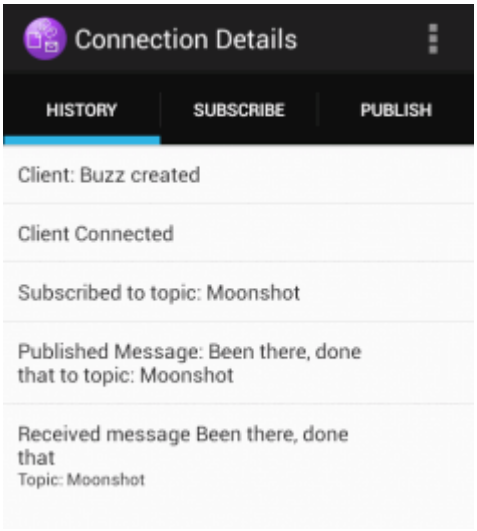


ii) Klepněte na akci **Publikovat** .

Krátce se zobrazí dvě zprávy, "Publikované" následované zprávou "Odebírané". Publikování se zobrazí ve stavové oblasti (stažením oddělovacího pruhu dolů otevřete stavové okno).



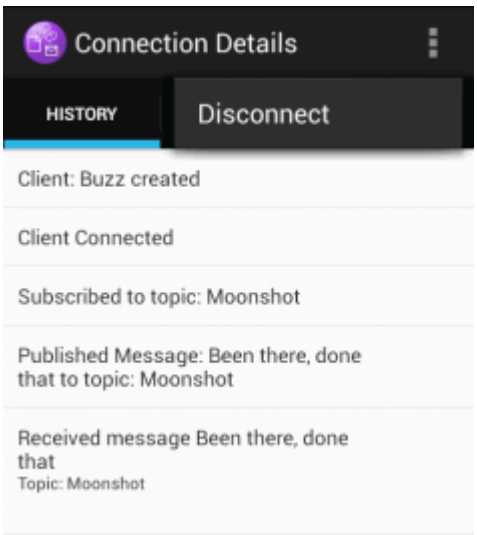
iii) Klepnutím na kartu **Historie** zobrazíte úplnou historii.



e) Odpojte instanci klienta.

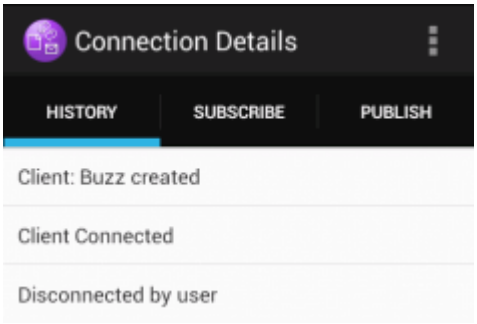
i) Klepněte na ikonu nabídky na řádku s akcemi.

Ukázková aplikace klienta MQTT Java pro produkt Android přidá tlačítko **Odpojit** do okna MQTT **Podrobnosti připojení** .

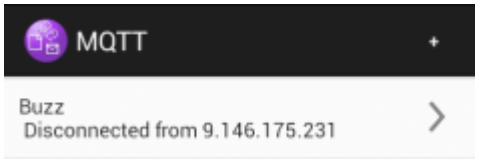


ii) Klepněte na volbu **Odpojit**.

Připojený stav se změní na odpojený:



f) Klepnutím na tlačítko **Zpět** se vrátíte na seznam relací Ukázková aplikace klienta MQTT Java .



- Klepnutím na znaménko plus spusťte novou relaci Ukázková aplikace klienta MQTT Java .
- Chcete-li odpojeného klienta znovu připojit, klepněte na něj.
- Klepnutím na tlačítko **Zpět** se vrátíte na příruční panel.

g) Klepnutím na tlačítko úlohy zobrazíte seznam spuštěných aplikací. Vyhledejte soubor Ukázková aplikace klienta MQTT Java. Přejeďte prstem po ikoně mimo obrazovku a zavřete ji.

Jak pokračovat dále

Pokud jste ukázkovou aplikaci sestavili sami, jste připraveni začít vyvíjet vlastní aplikace Android , které volají knihovny MQTT pro výměnu zpráv. Aplikace Android můžete modelovat na třídách v produktu `mqttExerciser`. Chcete-li studovat ukázku, vygenerujte Javadoc pro třídy v `com.ibm.msg.android` a `com.ibm.msg.android.service` v projektu `mqttExerciser` .

Související informace

Správa projektů z prostředí Eclipse pomocí ADT

Začínáme s produktem Klient systému zpráv MQTT pro produkt JavaScript

Můžete začít pracovat se serverem Klient systému zpráv MQTT pro produkt JavaScript zobrazením ukázkové domovské stránky klienta systému zpráv a procházením prostředků, na které se odkazuje. Chcete-li zobrazit tuto domovskou stránku, nakonfigurujte server MQTT tak, aby přijímal připojení ze serveru Ukázkové stránky klienta systému zpráv MQTT JavaScript, pak napíšete adresu URL, kterou jste nakonfigurovali na serveru do webového prohlížeče. Produkt Klient systému zpráv MQTT pro produkt JavaScript se automaticky spustí na vašem zařízení a zobrazí se ukázková domovská stránka klienta systému zpráv. Tato stránka obsahuje odkazy na obslužné programy, dokumentaci k programovacímu rozhraní, výukový program a další užitečné informace.

Než začnete

Pro rozšířené použití nebo pro použití v produkci budete chtít aktualizovat ukázkovou domovskou stránku klienta systému zpráv nebo ji odebrat. Vezměte prosím na vědomí, že uživatelská rozhraní, která jsou výsledkem kódu ukázky, nejsou oprávněná, aby byla kompatibilní se standardy usnadnění přístupu nebo požadavky na usnadnění přístupu.

Chcete-li podporovat produkt Klient systému zpráv MQTT pro produkt JavaScript, musíte mít server MQTT . Tento server musí podporovat protokol MQTT V3.1 nad WebSockets. IBM MessageSight, a IBM WebSphere MQ Version 7.5.0, Fix Pack 1 a novějších verzí, podpora MQTT protocol přes WebSockets. Viz “Začínáme se servery MQTT” na stránce 135. Chcete-li instalovat produkt IBM WebSphere MQ pro bezplatné 90denní vyhodnocení, prohlédněte si téma “instalaceIBM WebSphere MQ” na stránce 137.

Objekt WebSocket protocol byl nedávno vytvořen. Je-li mezi vaším klientem a serverem brána firewall, zkontrolujte, zda neblokuje provoz WebSockets. Podobně, pokud váš prohlížeč ještě nepodporuje WebSocket protocol[1]nebudete moci používat obslužný program klienta nebo výukové programy, které jsou k dispozici na ukázkové domovské stránce klienta systému zpráv. Tabulka Tabulka 1 na stránce 24 obsahuje seznam prohlížečů, jejichž nejnovější verze byly testovány a jsou zobrazeny pro práci s klientem systému zpráv.

[1] Konkrétně, pokud nepodporuje standard RFC 6455 (WebSocket).

Tabulka 1. Podporované prohlížeče pro použití s produktem Klient systému zpráv MQTT pro produkt JavaScript			
Android	**iOS**	**Linux**	**Windows**
Firefox for Android 19.0 a novější Chrome for Android 25.0 a novější	Safari 6.0 a novější Chrome 14.0 a novější	Firefox 6.0 a novější Chrome 14.0 a novější	Firefox 6.0 a novější Chrome 14.0 a vyšší

Informace o této úloze

Většina kroků v této úloze je konfigurovat server MQTT . Vše, co je nezbytné pro přístup ke klientovi systému zpráv produktu JavaScript , je spustit prohlížeč, který podporuje WebSocket protocol.

V systému IBM WebSphere MQpostupujte podle kroků pro povolení produktu IBM WebSphere MQ Telemetry vytvořením ukázkových kanálů. Připojte se k vzorovém výchozímu kanálu produktu MQTT WebSockets na portu 1883. Adresa URL ukázkové domovské stránky klienta systému zpráv je `http://hostname:1883` na IBM WebSphere MQ.

V produktu IBM MessageSightnainstalujte a nastavte zařízení, nakonfigurujte rozbočovač systému zpráv tak, aby přijímal připojení, a vytvořte koncový bod MQTT WebSockets .

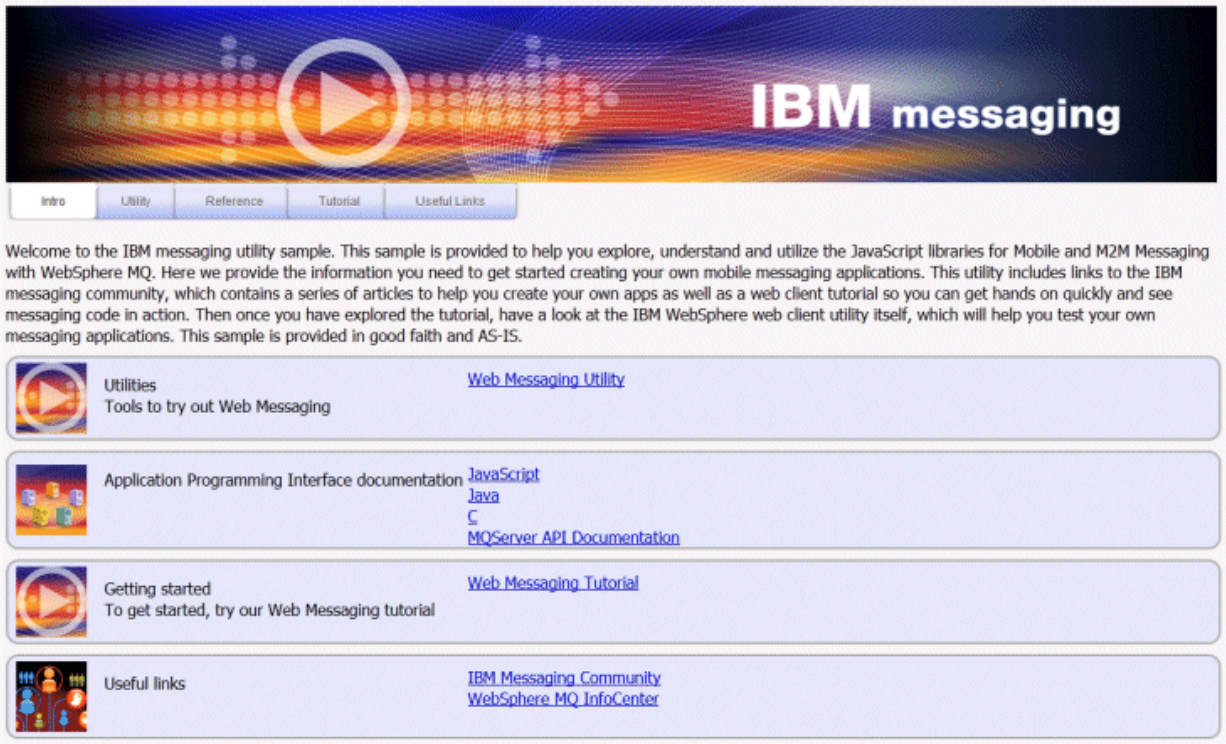
Postup

1. Stáhněte soubor Mobile Messaging and M2M Client Packa zvolte server MQTT , ke kterému se můžete připojit k aplikaci klienta.

   Viz “Začínáme s klienty MQTT” na stránce 10.
2. Konfigurujte svůj server MQTT tak, aby přijímal připojení ze vzorových stránek HTML Klient systému zpráv MQTT pro produkt JavaScript .
   - V systému IBM WebSphere MQ:
     - Pokud již máte správce front IBM WebSphere MQ , který je nakonfigurovaný pro produkt MQTT, změňte protokol v definici kanálu tak, aby podporoval protokol MQTT i HTTP. Viz **ALTER CHANNEL**.
     - Chcete-li vytvořit správce front produktu IBM WebSphere MQ a konfigurovat ukázkový koncový bod produktu MQTT WebSockets , proveďte jednu z následujících úloh:
       - “Konfigurace služby MQTT z příkazového řádku” na stránce 139
       - “Konfigurace služby MQTT s produktem IBM WebSphere MQ Explorer” na stránce 141
3. Ve svém zařízení otevřete webový prohlížeč.
4. Zadejte adresu URL ukázkové domovské stránky klienta systému zpráv.
   - V systému IBM WebSphere MQse jedná o `http://hostname:1883`
   - V systému IBM MessageSightse jedná o `http://hostname:port`

     kde *hostname* je název DNS nebo adresa IP adaptéru Ethernet, který jste nakonfigurovali na zařízení IBM MessageSight jako koncový bod, ke kterému se má klient připojit, a *port* je číslo portu TCP/IP, které jste přiřadili koncovému bodu pro klienta.

   Zobrazí se ukázková domovská stránka klienta systému zpráv.



Obrázek 9. Ukázková domovská stránka produktu Klient systému zpráv MQTT pro produkt JavaScript

Výsledky

Nakonfigurovali jste kanál MQTT pro produkt WebSockets.

Na ukázkové domovské stránce produktu Klient systému zpráv MQTT pro produkt JavaScriptklepněte na volbu **Obslužný program webového systému zpráv** a vyzkoušejte různé funkce v rozhraní API klienta systému zpráv. Můžete se například připojit ke správci front, přihlásit se k odběru zpráv a poté publikovat některé zprávy. Můžete také klepnout na volbu **Výukový program webového systému zpráv** a naučit se, jak vytvořit webovou stránku, která volá klienta systému zpráv produktu MQTT pro rozhraní API produktu JavaScript .

Související pojmy

“Aplikace Klient systému zpráv MQTT pro produkt JavaScript a webové aplikace” na stránce 116
“Jak naprogramovat aplikace systému zpráv v produktu JavaScript” na stránce 120

Související úlohy

“Připojení Klient systému zpráv MQTT pro produkt JavaScript přes SSL a WebSockets” na stránce 77
Webovou aplikaci bezpečně připojte k produktu IBM WebSphere MQ pomocí ukázkových HTML stránek Klient systému zpráv MQTT pro produkt JavaScript s SSL a WebSocket protocol.

Začínáme s klientem MQTT pro jazyk C

Vstaňte a běží s ukázkovým klientem MQTT pro C na libovolné platformě, na které můžete kompilovat zdroj C. Ověřte, že můžete spustit ukázkového klienta MQTT pro prostředí C s produktem buď IBM MessageSight , nebo IBM WebSphere MQ jako server MQTT.

Než začnete

- Je-li mezi vaším klientem a serverem brána firewall, zkontrolujte, zda neblokuje provoz produktu MQTT .

- Podporovaný a referenční MQTT klient pro platformy C. Viz Systémové požadavky pro IBM Mobile Messaging and M2M Client Pack.

Informace o této úloze

Postupujte podle této úlohy ke kompilaci a spuštění ukázkového klienta MQTT pro komponentu C na systému Windows z příkazového řádku nebo z produktu Microsoft Visual Studio 2010. Microsoft Visual Studio 2010 se také používá ke kompilaci klienta v příkladu z příkazového řádku. Upravte skripty příkazového řádku pro kompilaci a spuštění ukázky na jiných platformách.

Postup

1. Zvolte server MQTT , ke kterému se můžete připojit k aplikaci klienta.

   Server musí podporovat protokol MQTT version 3.1 . Všechny servery MQTT z IBM to dělají, včetně IBM WebSphere MQ a IBM MessageSight. Viz “Začínáme se servery MQTT” na stránce 135.
2. Nainstalujte vývojové prostředí C na platformě, na které sestavujete.

   Soubory Makefile v příkladech v tomto tématu se zaměřují na následující nástroje:

   - 

Pro iOS, v Apple Mac s OS X 10.8.2 s vývojovými nástroji iOS z Xcode.
   - 

Pro Linux, gcc verze 4.4.6 z Red Hat® Enterprise Linux verze 6.2.

     Minimální podporovaná úroveň knihovny `glibc` C je 2.12a jádro Linux je 2.6.32.
   - 

Pro Microsoft Windows, Visual Studio verze 10.0.
3. Stáhněte si produkt Mobile Messaging and M2M Client Pack a nainstalujte sadu SDK produktu MQTT .

   K dispozici není žádný instalační program, stačí jen rozbalit stažený soubor.

   a. Stáhněte si Mobile Messaging and M2M Client Pack.

   b. Vytvořte složku, do které budete instalovat sadu SDK.

      Je možné, že budete chtít pojmenovat složku `MQTT`. Cesta k této službě je zde označována jako *`sdkroot`*.

   c. Rozbalte komprimovaný obsah souboru Mobile Messaging and M2M Client Pack do *`sdkroot`*. Rozšíření vytváří adresářový strom, který začíná v *`sdkroot`*`\SDK`.
4. Volitelné: Postupujte podle kroků uvedených v tématu “Sestavení klienta MQTT pro knihovny C” na stránce 30.

   Tento krok proveďte pouze v případě, že Mobile Messaging and M2M Client Pack nezahrnuje knihovnu klienta C pro vaši cílovou platformu.
5. Kompilujte a spusťte ukázkovou aplikaci C klienta MQTT , `MQTTV3Sample.c`.

   - Z příkazového řádku postupujte podle kroků uvedených v tématu “Kompilace a spuštění ukázkové aplikace C klienta MQTT z příkazového řádku” na stránce 26.
   - V prostředí IDE postupujte podle kroků v části “Kompilace a spuštění ukázkové aplikace C klienta MQTT z produktu Microsoft Visual Studio” na stránce 27.

Kompilace a spuštění ukázkové aplikace C klienta MQTT z příkazového řádku

Kompilujte a spusťte vzorovou aplikaci C klienta MQTT z příkazového řádku. Ukázka je uvedena v sadě SDK produktu MQTT . Demonstruje vydavatele a odběratele produktu MQTT .

Než začnete

Nainstalujte vývojové prostředí C; například produkt Microsoft Visual Studio 2010, jak je použit v příkladu.

Informace o této úloze

Kompilujte a spusťte ukázku jazyka C, MQTTV3Sample, v podadresáři klientů SDK, *sdkroot*\SDK\clients\c\samples.

Postup

Vytvořte skript v adresáři ukázek klienta, abyste kompilovali a spustili Sample na zvolené platformě.

Následující skript zkompiluje a spustí ukázku na 32bitové platformě Windows , sestavené pomocí produktu Microsoft Visual Studio 2010. Spusťte skript z podadresáře *sdkroot*\SDK\clients\c\samples .

```
@echo off
setlocal
call "C:\Program Files\Microsoft Visual Studio 10.0\VC\vcvarsall.bat" x86
cl  /nologo /D "WIN32" /I "..\include" "MQTTV3Sample.c" /link /
nologo ..\windows_ia32\mqttv3c.lib
set path=%path%;..\windows_ia32;
start "MQTT Subscriber" MQTTV3Sample -a subscribe -b localhost -p 1883
@rem Sleep for 2 seconds
ping -n 2 127.0.0.1 > NUL 2>&1
MQTTV3Sample -b localhost -p 1883
pause
endlocal
```

Výsledky

Vydavatel a předplatitel zapisují výstup do svých příkazových oken:

```
Setting environment for using Microsoft Visual Studio 2008 x86 tools.
MQTTV3Sample.c
Connected to tcp://localhost:1883
Publishing to topic "MQTTV3Sample/C/v3" qos 2
Disconnected
Press any key to continue . . .
```

Obrázek 10. Výstup z vydavatele

```
Connected to tcp://localhost:1883
Subscribing to topic "MQTTV3Sample/#" qos 2
Topic:          MQTTV3Sample/C/v3
Message:        Message from MQTTv3 C client
QoS:            2
```

Obrázek 11. Výstup z odběratele

Kompilace a spuštění ukázkové aplikace C klienta MQTT z produktu Microsoft Visual Studio

Kompilujte a spusťte ukázkovou aplikaci C klienta MQTT z produktu Microsoft Visual Studio. Ukázka je v Mobile Messaging and M2M Client Pack. Demonstruje vydavatele a odběratele produktu MQTT .

Než začnete

V příkladu je použit produkt Microsoft Visual Studio 2010. V produktu Windows a jiných platformách můžete použít jiná vývojová prostředí C; například Eclipse IDE for C/C++ Developers.

Informace o této úloze

Kompilujte a spusťte ukázku jazyka C, MQTTV3Sample s produktem Microsoft Visual Studio 2010. MQTTV3Sample.c je v podadresáři klientů SDK *sdkroot*\SDK\clients\c\samples.

Postup

1. Spusťte produkt Microsoft Visual Studio.
2. Vytvořit nový projekt z existujícího kódu.
   a) Klepněte na volbu **Soubor** > **Nový** > **Projekt z existujícího kódu**.
   b) Vyberte typ projektu **Visual C++** jako typ projektu, který má být vytvořen.
   c) Klepněte na tlačítko **Další**.
3. Uveďte parametry v okně **Umístění projektu a zdrojové soubory** .
   a) Klepněte na tlačítko **Procházet** a vyhledejte adresář *sdkroot*\SDK\clients\c\samples .
   b) Pojmenujte projekt MQTTV3Sample.
   c) Klepněte na tlačítko **Další**.
4. Vyberte volbu **Projekt aplikace konzoly** v seznamu **Typ projektu** . Klepněte na tlačítko **Dokončit**.
5. Konfigurovat pouze konfiguraci ladění.

   Ve výchozím nastavení Microsoft Visual Studio vytvoří konfiguraci vydání i ladění. Ve výukovém programu nakonfigurujete konfiguraci ladění. Chcete-li potlačit chyby sestavení, zrušte zaškrtnutí volby **Sestavit** pro konfiguraci vydání.

   a) Klepněte na volbu **Projekt** > **MQTTV3Sample Vlastnosti** > **Configuration Manager**. Vyberte volbu **Vydání** jako **Konfigurace aktivního řešení** a zrušte zaškrtnutí volby **Sestavit**.
   b) Vyberte volbu **Ladit** jako **Konfigurace aktivního řešení** > **Zavřít**.

      Zkontrolujte, že upravujete konfiguraci ladění ve všech následujících krocích.
6. Upravte nastavení **C/C++** v **MQTTV3Sample Stránky vlastností**.
   a) V okně **MQTTV3Sample Stránky vlastností** otevřete nabídku **Vlastnosti konfigurace** > **C/C++** > **Obecné**.
   b) V seznamu obecných vlastností klepněte na volbu **Další adresáře Include**a přidejte cestu k adresáři produktu *sdkroot*\SDK\clients\c\include a klepněte na tlačítko **Použít**.
7. Upravte nastavení **Linker**
   a) Otevřete **Vlastnosti konfigurace** > **Linker** > **Obecné**.
   b) V seznamu obecných vlastností klepněte na volbu **Další adresáře knihovny**a přidejte cestu k adresáři produktu *sdkroot*\SDK\clients\c\windows_ia32 .
   c) V seznamu vlastností Linker klepněte na **Příkazový řádek**. Zadejte mqttv3c.lib do oblasti zadávání dat **Další volby** a klepněte na tlačítko **Použít**.
8. Odeberte zdrojový soubor MQTTV3SSample.c z projektu.
   a) Otevřete složku **MQTTV3sample** > **Zdrojové soubory** v okně **Průzkumník řešení** .
   b) Klepněte pravým tlačítkem myši na **MQTTV3SSample.c** > **Vyloučit z projektu**
9. Sestavte projekt produktu MQTTV3Sample .
   a) Klepněte pravým tlačítkem myši na projekt **MQTTV3sample** v Průzkumníku řešení a klepněte na volbu **Sestavit**.

      Sestavení se dokončí bez chyb.
10. Přidejte dva nové projekty ke spuštění produktu **MQTTV3Sample** jako instance běhového prostředí odběratele a vydavatele.

    Projekty Vydavatel a Odběratel mají obsahovat příkazy ke spuštění produktu **MQTTV3Sample**. Nejsou sestaveny a neobsahují žádný kód.

    a) V **Průzkumníku řešení**klepněte pravým tlačítkem myši na položku **Řešení ` MQTTV3Sample `** > **Přidat** > **Nový projekt**.
    b) Zadejte Subscriber do pole **Název** . Ponechte vybranou volbu **Win32 Console Application** . Klepněte na tlačítko **OK**.

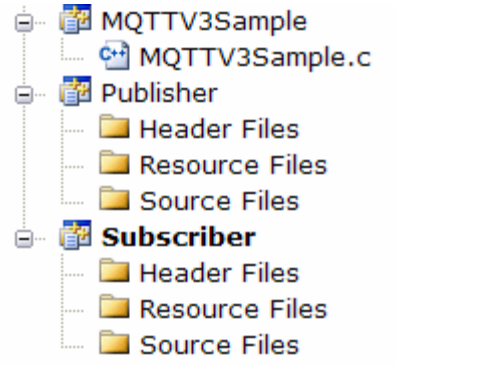
       Spustí se **Průvodce aplikacíWin32** .

c) V **Průvodci aplikacíWin32**klepněte na tlačítko **Další**. Zaškrtněte políčko **Prázdný projekt** > **Dokončit**

d) Opakováním těchto kroků přidejte projekt vydavatele.

e) Klepněte pravým tlačítkem myši na projekt odběratele a klepněte na volbu **Nastavit jako projekt StartUp** .

Okno **Průzkumník řešení** se zobrazí v produktu Obrázek 12 na stránce 29.



Obrázek 12. Řešení MQTTV3Sample

11. Konfigurujte stránky vlastností odběratele.

a) Klepněte pravým tlačítkem myši na položku **Odběratel** v Průzkumníku řešení **Vlastnosti** > **Vlastnosti konfigurace** > **Vlastnosti** > **Ladění** .

Zkontrolujte, že název okna je **Stránky vlastností odběratele**.

b) Klepněte na volbu **Prostředí**. Zadejte `path=%path%;sdkroot\SDK\clients\c\windows_ia32` a klepněte na **Použít**.

Změňte *sdkroot* tak, aby vyhovoval vašemu prostředí.

c) Klepněte na volbu **Příkaz**a nahraďte hodnotu `$(TargetPath)` cestou k modulu `MQTTV3Sample`

Například: *sdkroot*`\SDK\clients\c\samples\debug\MQTTV3Sample`

Změňte *sdkroot* tak, aby vyhovoval vašemu prostředí.

d) Klepněte na volbu **Argumenty příkazu** a zadejte `-a subscribe -b localhost -p 1883` a klepněte na tlačítko **Použít**.

12. Konfigurujte stránky vlastností vydavatele.

Tip: Projekt stránek vlastností můžete změnit klepnutím na projekty v okně **Průzkumník řešení** .

a) Zopakujte kroky odběratele pro vydavatele.

Argument příkazu je `-b localhost -p 1883`

13. Zastavte proces sestavení, který sestavuje projekty vydavatele a odběratele.

a) Na stránkách vlastností kteréhokoli z projektů klepněte na volbu **Configuration Manager** a zrušte výběr volby **Sestavit** pro vydavatele a odběratele v konfiguracích Vydání a Ladění. Klepněte na **Zavřít**.

14. Spusťte ukázku.

a) Klepnutím na tlačítko **F5** spusťte odběratele.

b) Klepněte pravým tlačítkem myši na položku **Vydavatel** v Průzkumníku řešení, **Ladit** > **Spustit novou instanci** .

Výsledky

Výstup vydavatele a odběratele do příkazového okna. Produkt Visual Studio zavře okno vydavatele. Podívejte se na okno odběratele, které je zobrazeno na následujícím obrázku, a poté zavřete okno odběratele.

```
Connected to tcp://localhost:1883
Subscribing to topic "MQTTV3Sample/#" qos 2
Topic:          MQTTV3Sample/C/v3
Message:        Message from MQTTv3 C client
QoS:            2
```

Obrázek 13. Výstup z odběratele

Jak pokračovat dále

Sestavit a spustit asynchronního vydavatele a odběratele. Příklady jsou `MQTTV3ASample.c` a `MQTTV3ASSample.c` v *sdkroot*`\SDK\clients\c\samples`.

Sestavení klienta MQTT pro knihovny C

Chcete-li sestavit klienta produktu MQTT pro knihovny jazyka C, postupujte podle následujících kroků. Téma obsahuje přepínače kompilace a linkování pro řadu platforem a příklady sestavení knihoven na systémech iOS a Windows.

Než začnete

1. Sestavte knihovnu klienta jazyka C pouze v případě potřeby. Propojte předpřipravené knihovny klienta v sadě SDK (Software Development Kit) SDK v podadresáři `SDK\clients\c` , pokud se jedna shoduje s cílovou platformou.
2. Nakonfigurujte server MQTT k testování knihovny, kterou sestavujete s produktem Ukázková aplikace C klienta MQTT. Viz “Začínáme se servery MQTT” na stránce 135. Ověřte konfiguraci serveru spuštěním jedné z ukázkových aplikací klienta MQTT .
3. Sestavujete-li zabezpečenou verzi knihovny jazyka C, která podporuje (Secure Sockets Layer) SSL, musíte také sestavit knihovnu OpenSSL . Viz “Sestavení balíku produktu OpenSSL” na stránce 44.

   Důležité: Stažení a redistribuce balíku produktu OpenSSL podléhá přísnému režimu importu a exportu a podmínkám licencování typu open source. Všimněte si omezení a varování před tím, než se rozhodnete, zda stáhnout balík.

Informace o této úloze

Chcete-li stáhnout a sestavit knihovnu produktu OpenSSL , postupujte podle pokynů v části “Sestavení balíku produktu OpenSSL” na stránce 44 . Chcete-li sestavit zabezpečenou verzi klienta MQTT pro knihovnu C, musíte sestavit produkt OpenSSL . Nevyžadujete, aby produkt OpenSSL vytvářeli nezabezpečenou verzi knihovny MQTT . Tento postup obsahuje příklady sestavení knihovny pro produkty iOS a Windows.

Sestavte klienta produktu MQTT pro knihovnu C stažením nástrojů vývojové knihovny C a sadou nástrojů pro vývoj softwaru (SDK) produktu MQTT na vaši platformu sestavení. Napište makefile a sestavte knihovnu pro cílovou platformu, která obsahuje volby, které jsou dokumentovány v příručce “Volby sestavení produktuMQTT pro různé platformy” na stránce 31. Kroky specifické pro platformu pro sestavení a spuštění souboru Makefile jsou uvedeny zde:

- iOS “Sestavování knihoven klienta MQTT pro C na serveru Apple Mac pro použití se zařízeními iOS” na stránce 33
- Windows “Sestavování knihoven produktu MQTT v systému Windows” na stránce 39

Postup

1. Nainstalujte vývojové prostředí C na platformě, na které sestavujete.

   Soubory Makefile v příkladech v tomto tématu se zaměřují na následující nástroje:

   - iOS Pro iOS, v Apple Mac s OS X 10.8.2 s vývojovými nástroji iOS z Xcode.

- 

Pro Linux, gcc verze 4.4.6 z Red Hat Enterprise Linux verze 6.2.

  Minimální podporovaná úroveň knihovny `glibc` C je 2.12a jádro Linux je 2.6.32.
- 

Pro Microsoft Windows, Visual Studio verze 10.0.

2. Stáhněte si produkt Mobile Messaging and M2M Client Pack a nainstalujte sadu SDK produktu MQTT .

   K dispozici není žádný instalační program, stačí jen rozbalit stažený soubor.

   a. Stáhněte si Mobile Messaging and M2M Client Pack.

   b. Vytvořte složku, do které budete instalovat sadu SDK.

      Je možné, že budete chtít pojmenovat složku `MQTT`. Cesta k této službě je zde označována jako *`sdkroot`*.

   c. Rozbalte komprimovaný obsah souboru Mobile Messaging and M2M Client Pack do *`sdkroot`*. Rozšíření vytváří adresářový strom, který začíná v *`sdkroot`*\SDK.

3. Rozbalte zdrojový kód pro klienta MQTT pro knihovny C.

   Komprimovaný soubor zdrojového kódu je *`sdkroot`*`\SDK\clients\c\source.zip`.

4. Volitelné: Sestavte OpenSSL.

   Viz “Sestavení balíku produktu OpenSSL” na stránce 44.

5. Sestavte klienta MQTT pro knihovny C.

   Příkazy a volby pro sestavení knihoven jsou uvedeny v seznamu “Volby sestavení produktuMQTT pro různé platformy” na stránce 31.

   Postupujte podle kroků uvedených v následujících příkladech a napište soubor Makefile pro sestavení klienta MQTT pro knihovny C pro vaši cílovou platformu.

   - “Sestavování knihoven klienta MQTT pro C na serveru Apple Mac pro použití se zařízeními iOS” na stránce 33
   - “Sestavování knihoven produktu MQTT v systému Windows” na stránce 39

Volby sestavení produktuMQTT pro různé platformy

V následující tabulce jsou uvedeny volby kompilátoru a sestavení pro sestavení klienta MQTT pro knihovny C na různých platformách.

Tabulka 2. Volby sestavení produktuMQTT pro různé platformy

Platforma	Compiler	Compiler Options	Linker Options	Extra Options
AIX	gcc	-fPIC -Os -Wall -DREVERSED -I *MQTTCLIENT_DIR*	-Wl,-G	
Linux s390x			-shared -Wl,-soname, libmqttv3c.so	-m 64
Linux x86-64		-fPIC -Os -Wall -I *MQTTCLIENT_DIR*		
Linux x86-32				-m 32
Linux ARM (glibc)	arm-linux-gcc			
Linux ARM (uclibc)	arm-unknown-linux -uclibcgnueabi-gcc			
Windows 32 bitů	cl	/D "WIN32" /D "_UNICODE" /D "UNICODE" /D "_CRT_SECURE_NO_WARNINGS" / nologo /c /O2 /W3 /Fd /MD /TC	/nologo /machine:x86 /manifest kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib ws2_32.lib /implib:mqttv3c.lib) /pdb:mqttv3c.pdb) /map:mqttv3c.map)	
iOS ARMv7	gcc -arch armv7	-DUSE_NAMED_SEMAPHORES -DNOSIGPIPE -DOPENSSL -Os -Wall -fomit-frame-pointer -isysroot /Applications/Xcode.app/\ Contents/Developer/Platforms/\ iPhoneOS.platform/Developer/SDKs/\ iPhoneOS6.0.sdk	-L/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS6.0.sdk/usr/lib/system	
iOS ARMv7s	gcc -arch armv7s			

Tabulka 2. Volby sestavení produktuMQTT pro různé platformy (pokračování)

Platforma	Compiler	Compiler Options	Linker Options	Extra Options
iOS ARMi386	gcc -arch i386	-DUSE_NAMED_SEMAPHORES -DNOSIGPIPE -DOPENSSL -Os -Wall -fomit-frame-pointer -isysroot / Applications/ Xcode.app/\ Contents/Developer/ Platforms/\ iPhoneSimulator.platform/ Developer/SDKs/ iPhoneSimulator6.0.sdk	-L/Applications/Xcode.app/ Contents/Developer/ Platforms/ iPhoneSimulator.platform/ Developer/SDKs/ iPhoneSimulator6.0.sdk/usr /lib/system	

iOS *Sestavování knihoven klienta MQTT pro C na serveru Apple Mac pro použití se zařízeními iOS*

Chcete-li vytvořit soubor Makefile pro sestavení knihoven klienta MQTT pro C na serveru Apple Macpro další použití se zařízeními iOS , postupujte podle následujících kroků.

Než začnete

1. Nainstalujte nástroje pro sestavení, vyvinout a spustit soubor Makefile v produktu Apple Mac s operačním systémem OS X 10.8.2nebo novějším.
2. Nainstalujte nástroje příkazového řádku pro Xcode, které zahrnují program **make** . Stáhněte nástroje příkazového řádku z Xcode.

Informace o této úloze

Vytvořte soubor Makefile, který sestaví knihovny klienta MQTT pro systém C pro produkt iPhone nebo iPad se spuštěným procesorem ARMv7 nebo ARMv7s , a simulátorem produktu iPhone , který je spuštěn na 64bitovém procesoru i386-64 . Viz Seznam zařízení iOS.

Tip: “Výpis Makefile MQTTios.mak” na stránce 37 vypíše úplný soubor Makefile.

1. Zkopírujte a vložte výpis do souboru.
2. Převeďte úvodní znak každého řádku, který následuje za cílem na kartě; viz krok “8” na stránce 35.
3. Spusťte jej s příkazem uvedeným v kroku “9” na stránce 37 procedury.

Postup

1. Stáhněte a nainstalujte vývojové nástroje produktu iOS .
   a. Přihlaste se s ID uživatele, který má oprávnění k administraci.
   b. Zkontrolujte Apple Mac ve verzi 10.8.2 nebo novější.

c. Přejděte na webový server Xcode , chcete-li stáhnout produkt Xcode z obchodu s aplikacemi produktu Mac .

d. Nainstalujte produkt Xcode, prostředí příkazového řádku a simulátor.

Pokud má aplikace Mac App Store více verzí simulátoru, vyberte verzi, která je kompatibilní s úrovní iOS , kterou cílíte pro vaši aplikaci.

2. Vytvořit soubor Makefile `MQTTios.mak`

Přidejte prolog:

```
# Výstup sestavení se vytváří v aktuálním adresáři.
# MQTTCLIENT_DIR musí odkazovat na základní adresář obsahující zdrojový kód klienta MQTT.
# Výchozí hodnota MQTTCLIENT_DIR je aktuální adresář.
# Výchozí parametr TOOL_DIR je /Applications/Xcode.app/Contents/Developer/Platforms
# Výchozí hodnota OPENSSL_DIR je sdkroot/openssl, relativní k adresáři sdkroot/sdk/clients/c/
mqttv3c/src
# OPENSSL_DIR musí odkazovat na základní adresář obsahující sestavení OpenSSL .
# Příklad: make -f MQTTios.mak MQTTCLIENT_DIR=sdkroot/sdk/clients/c/mqttv3c/src all
```

3. Nastavte umístění zdrojového kódu produktu MQTT .

Spusťte soubor Makefile ve stejném adresáři jako zdrojové soubory produktu MQTT , nebo nastavte parametr příkazového řádku `MQTTCLIENT_DIR` :

```
make -f makefile MQTTCLIENT_DIR=sdkroot/SDK/clients/c/mqttv3c/src
```

Přidejte následující řádky do souboru makefile:

```
ifndef MQTTCLIENT_DIR
  MQTTCLIENT_DIR = ${CURDIR}
endif
VPATH = ${MQTTCLIENT_DIR}
```

Příklad nastaví produkt VPATH na adresář, kde produkt **make** hledá zdrojové soubory, které nejsou explicitně identifikovány; například všechny soubory záhlaví, které jsou nezbytné v sestavení.

4. Volitelné: Nastavte umístění knihoven produktu OpenSSL .

Tento krok je nezbytný pro sestavení verzí SSL klienta MQTT pro knihovny C.

Nastavte výchozí cestu ke knihovnám OpenSSL do stejného adresáře tak, jak jste rozbalili sadu SDK produktu MQTT . Jinak nastavte `OPENSSL_DIR` jako parametr příkazového řádku.

```
ifndef OPENSSL_DIR
  OPENSSL_DIR = ${MQTTCLIENT_DIR}/../../../../../openssl-1.0.1c
endif
```

Tip: *OpenSSL* je adresář OpenSSL , který obsahuje všechny podadresáře OpenSSL . Možná budete muset přesunout adresářový strom z místa, kde jste jej rozšířili, protože obsahuje nepotřebné prázdné nadřízené adresáře.

5. Nastavte adresáře vývojových nástrojů.

Pokud jste instalovali produkt Xcode do jiného umístění, nastavte parametr `TOOL_DIR` na příkazový řádek.

```
ifndef TOOL_DIR
   TOOL_DIR = /Applications/Xcode.app/Contents/Developer/Platforms endif
IPHONE_SDK = iPhoneOS.platform/Developer/SDKs/iPhoneOS6.0.sdk
IPHONESIM_SDK = iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator6.0.sdk
SDK_ARM = ${TOOL_DIR}/${IPHONE_SDK}
SDK_i386 = ${TOOL_DIR}/${IPHONESIM_SDK}
```

6. Vyberte všechny zdrojové soubory, které jsou nezbytné pro sestavení každé knihovny produktu MQTT . Také nastavte název a umístění knihovny MQTT pro sestavení.

Chcete-li zobrazit seznam všech zdrojových souborů produktu MQTT , přidejte následující řádek do souboru Makefile:

```
ALL_SOURCE_FILES = ${wildcard ${MQTTCLIENT_DIR}/*.c}
```

Zdrojové soubory závisí na tom, zda sestavujete synchronní nebo asynchronní knihovnu a zda knihovna zahrnuje SSL, či nikoli.

Přidejte jeden nebo více těchto řádků, které jsou závislé na cílech, které se mají sestavit. Sdílené knihovny se vytvářejí v adresáři `darwin_x86_64` .

- Synchronní, nezabezpečené:

```
MQTTLIB = mqttv3c
SOURCE_FILES = ${filter-out ${MQTTCLIENT_DIR}/MQTTAsync.c ${MQTTCLIENT_DIR}/SSLSocket.c,
${ALL_SOURCE_FILES}}
MQTTLIB_DARWIN = darwin_x86_64/lib${MQTTLIB} .a
```

- Synchronní, zabezpečené:

```
MQTTLIB_S = mqttv3cs
SOURCE_FILES_S = ${filter-out ${MQTTCLIENT_DIR}/MQTTAsync.c, ${ALL_SOURCE_FILES}}
MQTTLIB_DARWIN_S = darwin_x86_64/lib${MQTTLIB_S} .a
```

- Asynchronní, nezabezpečené:

```
MQTTLIB_A = mqttv3a
SOURCE_FILES_A = ${filter-out ${MQTTCLIENT_DIR}/MQTTClient.c ${MQTTCLIENT_DIR}/
SSLSocket.c, ${ALL_SOURCE_FILES}}
MQTTLIB_DARWIN_A = darwin_x86_64/lib${MQTTLIB_A} .a
```

- Asynchronní zabezpečení:

```
MQTTLIB_AS = mqttv3as
SOURCE_FILES_AS = ${filter-out ${MQTTCLIENT_DIR}/MQTTClient.c, ${ALL_SOURCE_FILES}}
MQTTLIB_DARWIN_AS = darwin_x86_64/lib${MQTTLIB_AS} .a
```

7. Definujte kompilátor a volby kompilátoru.

   Podívejte se na volby pro různé platformy, které jsou zobrazeny ve volbě Volby sestavení MQTT pro různé platformy.

   a) Nastavte Gnu projekt C a C++ (**gcc**) jako kompilátor.

      Vyberte tři křížové kompilátory, chcete-li sestavit knihovnu pro různá zařízení a simulátor produktu iPhone :

```
CC = iPhoneOS.platform/Developer/usr/bin/gcc
CC_armv7 = ${TOOL_DIR}/${CC} -arch armv7
CC_armv7s = ${TOOL_DIR}/${CC} -arch armv7s
CC_i386 = ${TOOL_DIR}/${CC} -arch i386
```

   b) Přidejte volby kompilátoru.

```
CCFLAGS = -Os -Wall -fomit-frame-pointer
```

   c) Přidejte cesty Include.

```
CCFLAGS_SO_ARM = ${CCFLAGS} -isyroot ${SDK_ARM} -I${OPENSSL_DIR}/include -L$
{SDK_ARM}/usr/lib/system
CCFLAGS_SO_i386 = ${CCFLAGS} -isyssroot ${SDK_i386} -I${OPENSSL_DIR}/include -L$
{SDK_i386}/usr/lib/system
```

8. Definujte cíle sestavení.

   Tip: Každý následující řádek, který definuje implementaci cíle, musí začínat znakem tabulátoru.

   a) Definujte cíl **all** .

      Cíl "all" sestaví všechny knihovny.

```
all: ${MQTTLIB_DARWIN} ${MQTTLIB_DARWIN_A} ${MQTTLIB_DARWIN_AS} ${MQTTLIB_DARWIN_S}
```

Vypsáním prvního seznamu se jedná o výchozí cíl.

a) Sestavte synchronní, nezabezpečenou knihovnu, `libmqttv3c.a`.

```
${MQTTLIB_DARWIN}: ${SOURCE_FILES}
    -mkdir darwin_x86_64
    ${CC_armv7} ${CCFLAGS_SO_ARM} -c ${SOURCE_FILES} -DUSE_NAMED_SEMAPHORES -DNOSIGPIPE
    libtool -static -syslibroot ${SDK_ARM} -o $@.armv7 *.o
    rm *.o
    ${CC_armv7s} ${CCFLAGS_SO_ARM} -c ${SOURCE_FILES} -DUSE_NAMED_SEMAPHORES -DNOSIGPIPE
    libtool -static -sysliroot ${SDK_ARM} -o $@.armv7s *.o
    rm *.o
    ${CC_i386} ${CCFLAGS_SO_i386} -c ${SOURCE_FILES} -DUSE_NAMED_SEMAPHORES -DNOSIGPIPE
    libtool -static -sysliroot ${SDK_i386} -o $@.i386 *.o
    rm *.o
    ligo -create $@.armv7 $@.armv7s $@.i386 -output $@
```

Příkaz `rm *.o` odstraní všechny soubory objektů, které jsou vytvořeny pro každou knihovnu. `lipo` zřetězí všechny tři knihovny do jednoho souboru.

b) Sestavte asynchronní, nezabezpečenou knihovnu, `libmqttv3a.a`.

```
${MQTTLIB_DARWIN_A}: ${SOURCE_FILES_A}
    -mkdir darwin_x86_64
    ${CC_armv7} ${CCFLAGS_SO_ARM} -c ${SOURCE_FILES_A} -DMQTT_ASYNC -DUSE_NAMED_SEMAPHOES
-DNOSIGPIPE
    libtool -static -syslibroot ${SDK_ARM} -o $@.armv7 *.o
    rm *.o
    ${CC_armv7s} ${CCFLAGS_SO_ARM} -c ${SOURCE_FILES_A} -DMQTT_ASYNC
-DUSE_NAMED_SEMAPHOES -DNOSIGPIPE
    libtool -static -sysliroot ${SDK_ARM} -o $@.armv7s *.o
    rm *.o
    ${CC_i386} ${CCFLAGS_SO_i386} -c ${SOURCE_FILES_A} -DMQTT_ASYNC -DUSE_NAMED_SEMAPHOES
-DNOSIGPIPE
    libtool -static -sysliroot ${SDK_i386} -o $@.i386 *.o
    rm *.o
    ligo -create $@.armv7 $@.armv7s $@.i386 -output $@
```

Příkaz `rm *.o` odstraní všechny soubory objektů, které jsou vytvořeny pro každou knihovnu. `lipo` zřetězí všechny tři knihovny do jednoho souboru.

c) Sestavte synchronní, zabezpečenou knihovnu, `libmqttv3cs.a`.

```
${MQTTLIB_DARWIN_S}: ${SOURCE_FILES_S}
    -mkdir darwin_x86_64
    ${CC_armv7} ${CCFLAGS_SO_ARM} -c ${SOURCE_FILES_S} -DOPENSSL -DUSE_NAMED_SEMAPHOES
-DNOSIGPIPE
    libtool -static -syslibroot ${SDK_ARM} -L${OPENSSL_DIR}/armv7 -lssl -lcrypto -o
$@.armv7 *.o
    rm *.o
    ${CC_armv7s} ${CCFLAGS_SO_ARM} -c ${SOURCE_FILES_S} -DOPENSSL -DUSE_NAMED_SEMAPHOES
-DNOSIGPIPE
    libtool -static -syslibroot ${SDK_ARM} -L${OPENSSL_DIR}/armv7s -lssl -lcrypto -o
$@.armv7s *.o
    rm *.o
    ${CC_i386} ${CCFLAGS_SO_i386} -c ${SOURCE_FILES_S} -DOPENSSL -DUSE_NAMED_SEMAPHOES
-DNOSIGPIPE
    libtool -static -syslibroot ${SDK_ARM} -L${OPENSSL_DIR}/i386 -lssl -lcrypto -o
$@.i386 *.o
    rm *.o
    ligo -create $@.armv7 $@.armv7s $@.i386 -output $@
```

Příkaz `rm *.o` odstraní všechny soubory objektů, které jsou vytvořeny pro každou knihovnu. `lipo` zřetězí všechny tři knihovny do jednoho souboru.

d) Sestavte asynchronní, zabezpečenou knihovnu, `libmqttv3as.a`.

```
${MQTTLIB_DARWIN_AS}: ${SOURCE_FILES_AS}
    -mkdir darwin_x86_64
    ${CC_armv7} ${CCFLAGS_SO_ARM} -c ${SOURCE_FILES_AS} -DMQTT_ASYNC -DOPENSSL
-DUSE_NAMED_SEMAPHORES -DNOSIGPIPE
    libtool -static -syslibroot ${SDK_ARM} -L${OPENSSL_DIR}/armv7 -lssl -lcrypto -o
$@.armv7 *.o
    rm *.o
```

```
    ${CC_armv7s} ${CCFLAGS_SO_ARM} -c ${SOURCE_FILES_AS} -DMQTT_ASYNC -DOPENSSL
-DUSE_NAMED_SEMAPHORES -DNOSIGPIPE
    libtool -static -syslibroot ${SDK_ARM} -L${OPENSSL_DIR}/armv7s -lssl -lcrypto -o
$@.armv7s *.o
    rm *.o
    ${CC_i386} ${CCFLAGS_SO_i386} -c ${SOURCE_FILES_AS} -DMQTT_ASYNC -DOPENSSL
-DUSE_NAMED_SEMAPHORES -DNOSIGPIPE
    libtool -static -syslibroot ${SDK_ARM} -L${OPENSSL_DIR}/i386 -lssl -lcrypto -o
$@.i386 *.o
    rm *.o
    ligo -create $@.armv7 $@.armv7s $@.i386 -output $@
```

Příkaz `rm *.o` odstraní všechny soubory objektů, které jsou vytvořeny pro každou knihovnu. `lipo` zřetězí všechny tři knihovny do jednoho souboru.

e) Definujte cíl **`clean`** .

Cíl "clean" odebere všechny soubory a adresáře, které jsou generovány souborem Makefile

```
FOTY: čisté
čisté:
    -rm -f *.obj
    -rm -f -r darwin_x86_64
```

9. Spusťte soubor Makefile.

```
make -f  MQTTCLIENT_DIR=sdkroot/sdk/clients/c/mqttv3c/src
```

Výsledky

V adresáři *sdkroot/sdk/clients/c/mqttv3c/src*/darwin_x86_64 jsou vytvořeny následující soubory.

```
libmqttv3c.a.armv7
libmqttv3c.a.armv7s
libmqttv3c.a.i386
libmqttv3c.a
libmqttv3a.a.armv7
libmqttv3a.a.armv7s
libmqttv3a.a.i386
libmqttv3a.a
libmqttv3cs.a.armv7
libmqttv3cs.a.armv7s
libmqttv3cs.a.i386
libmqttv3cs.a
libmqttv3as.a.armv7
libmqttv3as.a.armv7s
libmqttv3as.a.i386
libmqttv3as.a
```

Výpis Makefile `MQTTios.mak`

```
# Výstup sestavení se vytváří v aktuálním adresáři.
# MQTTCLIENT_DIR musí odkazovat na základní adresář obsahující zdrojový kód klienta MQTT.
# Výchozí hodnota MQTTCLIENT_DIR je aktuální adresář.
# Výchozí parametr TOOL_DIR je /Applications/Xcode.app/Contents/Developer/Platforms
# Výchozí hodnota OPENSSL_DIR je sdkroot/openssl, relativní k adresáři sdkroot/sdk/clients/c/
mqttv3c/src
# OPENSSL_DIR musí odkazovat na základní adresář obsahující sestavení OpenSSL .
# Příklad: make -f MQTTios.mak MQTTCLIENT_DIR=sdkroot/sdk/clients/c/mqttv3c/src all

ifndef MQTTCLIENT_DIR
  MQTTCLIENT_DIR = ${CURDIR}
endif
VPATH = ${MQTTCLIENT_DIR}
ifndef OPENSSL_DIR
  OPENSSL_DIR = ${MQTTCLIENT_DIR}/../../../../../openssl-1.0.1c
endif
ALL_SOURCE_FILES = ${wildcard ${MQTTCLIENT_DIR}/*.c}
ifndef TOOL_DIR
   TOOL_DIR = /Applications/Xcode.app/Contents/Developer/Platforms endif
IPHONE_SDK = iPhoneOS.platform/Developer/SDKs/iPhoneOS6.0.sdk
IPHONESIM_SDK = iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator6.0.sdk
SDK_ARM = ${TOOL_DIR}/${IPHONE_SDK}
```

```
SDK_i386 = ${TOOL_DIR}/${IPHONESIM_SDK}

MQTTLIB = mqttv3c
SOURCE_FILES = ${filter-out ${MQTTCLIENT_DIR}/MQTTAsync.c ${MQTTCLIENT_DIR}/SSLSocket.c, $
{ALL_SOURCE_FILES}}
MQTTLIB_DARWIN = darwin_x86_64/lib${MQTTLIB} .a
MQTTLIB_S = mqttv3cs
SOURCE_FILES_S = ${filter-out ${MQTTCLIENT_DIR}/MQTTAsync.c, ${ALL_SOURCE_FILES}}
MQTTLIB_DARWIN_S = darwin_x86_64/lib${MQTTLIB_S} .a
MQTTLIB_A = mqttv3a
SOURCE_FILES_A = ${filter-out ${MQTTCLIENT_DIR}/MQTTClient.c ${MQTTCLIENT_DIR}/SSLSocket.c, $
{ALL_SOURCE_FILES}}
MQTTLIB_DARWIN_A = darwin_x86_64/lib${MQTTLIB_A} .a
MQTTLIB_AS = mqttv3as
SOURCE_FILES_AS = ${filter-out ${MQTTCLIENT_DIR}/MQTTClient.c, ${ALL_SOURCE_FILES}}
MQTTLIB_DARWIN_AS = darwin_x86_64/lib${MQTTLIB_AS} .a

# compiler
CC = iPhoneOS.platform/Developer/usr/bin/gcc
CC_armv7 = ${TOOL_DIR}/${CC} -arch armv7
CC_armv7s = ${TOOL_DIR}/${CC} -arch armv7s
CC_i386 = ${TOOL_DIR}/${CC} -arch i386
CCFLAGS = -Os -Wall -fomit-frame-pointer
CCFLAGS_SO_ARM = ${CCFLAGS} -isyroot ${SDK_ARM} -I${OPENSSL_DIR}/include -L${SDK_ARM}/usr/lib/
system
CCFLAGS_SO_i386 = ${CCFLAGS} -isyssroot ${SDK_i386} -I${OPENSSL_DIR}/include -L$
{SDK_i386}/usr/lib/system

# targets
all: ${MQTTLIB_DARWIN} ${MQTTLIB_DARWIN_A} ${MQTTLIB_DARWIN_AS} ${MQTTLIB_DARWIN_S}

${MQTTLIB_DARWIN}: ${SOURCE_FILES}
    -mkdir darwin_x86_64
    ${CC_armv7} ${CCFLAGS_SO_ARM} -c ${SOURCE_FILES} -DUSE_NAMED_SEMAPHORES -DNOSIGPIPE
    libtool -static -syslibroot ${SDK_ARM} -o $@.armv7 *.o
    rm *.o
    ${CC_armv7s} ${CCFLAGS_SO_ARM} -c ${SOURCE_FILES} -DUSE_NAMED_SEMAPHORES -DNOSIGPIPE
    libtool -static -sysliroot ${SDK_ARM} -o $@.armv7s *.o
    rm *.o
    ${CC_i386} ${CCFLAGS_SO_i386} -c ${SOURCE_FILES} -DUSE_NAMED_SEMAPHORES -DNOSIGPIPE
    libtool -static -sysliroot ${SDK_i386} -o $@.i386 *.o
    rm *.o
    ligo -create $@.armv7 $@.armv7s $@.i386 -output $@

${MQTTLIB_DARWIN_A}: ${SOURCE_FILES_A}
    -mkdir darwin_x86_64
    ${CC_armv7} ${CCFLAGS_SO_ARM} -c ${SOURCE_FILES_A} -DMQTT_ASYNC -DUSE_NAMED_SEMAPHOES
-DNOSIGPIPE
    libtool -static -syslibroot ${SDK_ARM} -o $@.armv7 *.o
    rm *.o
    ${CC_armv7s} ${CCFLAGS_SO_ARM} -c ${SOURCE_FILES_A} -DMQTT_ASYNC -DUSE_NAMED_SEMAPHOES
-DNOSIGPIPE
    libtool -static -sysliroot ${SDK_ARM} -o $@.armv7s *.o
    rm *.o
    ${CC_i386} ${CCFLAGS_SO_i386} -c ${SOURCE_FILES_A} -DMQTT_ASYNC -DUSE_NAMED_SEMAPHOES
-DNOSIGPIPE
    libtool -static -sysliroot ${SDK_i386} -o $@.i386 *.o
    rm *.o
    ligo -create $@.armv7 $@.armv7s $@.i386 -output $@

${MQTTLIB_DARWIN_S}: ${SOURCE_FILES_S}
    -mkdir darwin_x86_64
    ${CC_armv7} ${CCFLAGS_SO_ARM} -c ${SOURCE_FILES_S} -DOPENSSL -DUSE_NAMED_SEMAPHOES
-DNOSIGPIPE
    libtool -static -syslibroot ${SDK_ARM} -L${OPENSSL_DIR}/armv7 -lssl -lcrypto -o $@.armv7 *.o
    rm *.o
    ${CC_armv7s} ${CCFLAGS_SO_ARM} -c ${SOURCE_FILES_S} -DOPENSSL -DUSE_NAMED_SEMAPHOES
-DNOSIGPIPE
    libtool -static -syslibroot ${SDK_ARM} -L${OPENSSL_DIR}/armv7s -lssl -lcrypto -o $@.armv7s
*.o
    rm *.o
    ${CC_i386} ${CCFLAGS_SO_i386} -c ${SOURCE_FILES_S} -DOPENSSL -DUSE_NAMED_SEMAPHOES
-DNOSIGPIPE
    libtool -static -syslibroot ${SDK_ARM} -L${OPENSSL_DIR}/i386 -lssl -lcrypto -o $@.i386 *.o
    rm *.o
    ligo -create $@.armv7 $@.armv7s $@.i386 -output $@

${MQTTLIB_DARWIN_AS}: ${SOURCE_FILES_AS}
    -mkdir darwin_x86_64
    ${CC_armv7} ${CCFLAGS_SO_ARM} -c ${SOURCE_FILES_AS} -DMQTT_ASYNC -DOPENSSL
-DUSE_NAMED_SEMAPHORES -DNOSIGPIPE
    libtool -static -syslibroot ${SDK_ARM} -L${OPENSSL_DIR}/armv7 -lssl -lcrypto -o $@.armv7 *.o
```

```
    rm *.o
    ${CC_armv7s} ${CCFLAGS_SO_ARM} -c ${SOURCE_FILES_AS} -DMQTT_ASYNC -DOPENSSL
-DUSE_NAMED_SEMAPHORES -DNOSIGPIPE
    libtool -static -syslibroot ${SDK_ARM} -L${OPENSSL_DIR}/armv7s -lssl -lcrypto -o $@.armv7s
*.o
    rm *.o
    ${CC_i386} ${CCFLAGS_SO_i386} -c ${SOURCE_FILES_AS} -DMQTT_ASYNC -DOPENSSL
-DUSE_NAMED_SEMAPHORES -DNOSIGPIPE
    libtool -static -syslibroot ${SDK_ARM} -L${OPENSSL_DIR}/i386 -lssl -lcrypto -o $@.i386 *.o
    rm *.o
    ligo -create $@.armv7 $@.armv7s $@.i386 -output $@

FOTY: čisté
čisté:
    -rm -f *.obj
    -rm -f -r darwin_x86_64
```

Windows *Sestavování knihoven produktu MQTT v systému Windows*

Postupujte takto, chcete-li napsat soubor Makefile pro sestavení knihoven klienta MQTT pro C na systému Windows.

Než začnete

1. V případě potřeby nainstalujte do pracovní stanice sestavení verzi produktu **Make** , která je kompatibilní se soubory Makefile napsanými pro příkaz Gnu make; v opačném případě stáhněte produkt Gnu a sestavte jej. Viz Gnu Make. Web, Make for Windows, poskytuje instalovatelnou verzi produktu **Make** for Windows.
2. Chcete-li použít čistý cíl v příkladu Makefile, musíte také použít příkaz Linux pro produkt Windows . Pro produkt Windows můžete získat příkazy Linux z webových stránek, jako je například Cygwin.

Informace o této úloze

Vytvořte soubor Makefile, který sestaví knihovny klienta MQTT pro jazyk C pro 32bitové prostředí Windows .

Tip: “Výpis Makefile MQTTwin.mak” na stránce 43 vypíše úplný soubor Makefile.

1. Zkopírujte a vložte výpis do souboru.
2. Převeďte úvodní znak každého řádku, který následuje za cílem na kartě; viz krok “7” na stránce 41.
3. Spusťte jej s příkazem uvedeným v kroku “9” na stránce 43 procedury.

Postup

1. Vytvořit soubor Makefile `MQTTwin.mak`

   Přidejte prolog:

   ```
   # Výstup sestavení se vytváří v aktuálním adresáři.
   # MQTTCLIENT_DIR musí odkazovat na základní adresář obsahující zdrojový kód klienta MQTT.
   # Výchozí hodnota MQTTCLIENT_DIR je aktuální adresář.
   # Výchozí hodnota OPENSSL_DIR je sdkroot\openSSL, relativní k adresáři
   sdkroot\sdk\clients\c\mqttv3c\src
   # OPENSSL_DIR musí odkazovat na základní adresář obsahující sestavení OpenSSL .
   # Příklad: make -f MQTTwin.mak MQTTCLIENT_DIR=sdkroot/sdk/clients/c/mqtv3c/src
   # Nastavte prostředí sestavení, například:
   # %comspec% /k "" C:\Program Files\Microsoft Visual Studio 9.0\VC\vcvarsall.bat"" x86
   # set path= %path%; C:\Program Files\GnuWin32\bin;C:\cygwin\bin
   ```

2. Nastavte umístění zdrojového kódu produktu MQTT .

   Spusťte soubor Makefile ve stejném adresáři jako zdrojové soubory produktu MQTT , nebo nastavte parametr příkazového řádku `MQTTCLIENT_DIR` :

   ```
   make -f makefile MQTTCLIENT_DIR=sdkroot/SDK/clients/c/mqttv3c/src
   ```

Přidejte následující řádky do souboru makefile:

```
ifndef MQTTCLIENT_DIR
  MQTTCLIENT_DIR = ${CURDIR}
endif
VPATH = ${MQTTCLIENT_DIR}
```

Příklad nastaví produkt VPATH na adresář, kde produkt **make** hledá zdrojové soubory, které nejsou explicitně identifikovány; například všechny soubory záhlaví, které jsou nezbytné v sestavení.

3. Volitelné: Nastavte umístění knihoven produktu OpenSSL .

   Tento krok je nezbytný pro sestavení verzí SSL klienta MQTT pro knihovny C.

   Nastavte výchozí cestu ke knihovnám OpenSSL do stejného adresáře tak, jak jste rozbalili sadu SDK produktu MQTT . Jinak nastavte OPENSSL_DIR jako parametr příkazového řádku.

   ```
   ifndef OPENSSL_DIR
     OPENSSL_DIR = ${MQTTCLIENT_DIR}/../../../../../openssl-1.0.1c
   endif
   ```

   Tip: *OpenSSL* je adresář OpenSSL , který obsahuje všechny podadresáře OpenSSL . Možná budete muset přesunout adresářový strom z místa, kde jste jej rozšířili, protože obsahuje nepotřebné prázdné nadřízené adresáře.
4. Vyberte všechny zdrojové soubory, které jsou nezbytné pro sestavení každé knihovny produktu MQTT . Také nastavte název a umístění knihovny MQTT pro sestavení.

   Chcete-li zobrazit seznam všech zdrojových souborů produktu MQTT , přidejte následující řádek do souboru Makefile:

   ```
   ALL_SOURCE_FILES = ${wildcard ${MQTTCLIENT_DIR}/*.c}
   ```

   Zdrojové soubory závisí na tom, zda sestavujete synchronní nebo asynchronní knihovnu a zda knihovna zahrnuje SSL, či nikoli.

   Přidejte jeden nebo více těchto řádků, které jsou závislé na cílech, které se mají sestavit. Sdílené knihovny a soubory typu manifest se vytvářejí v adresáři windows_ia32 .

   - Synchronní, nezabezpečené:

     ```
     MQTTLIB = mqttv3c
     MQTTDLL = windows_ia32/${MQTTLIB} .dll
     SOURCE_FILES = ${filter-out ${MQTTCLIENT_DIR}/MQTTAsync.c ${MQTTCLIENT_DIR}/SSLSocket.c,
     ${ALL_SOURCE_FILES}}
     MANIFEST = mt -manifest ${MQTTDLL}.manifest -outputresource: ${MQTTDLL}\; 2
     ```

   - Synchronní, zabezpečené:

     ```
     MQTTLIB_S = mqttv3cs
     MQTTDLL_S = windows_ia32/${MQTTLIB_S} .dll
     SOURCE_FILES_S = ${filter-out ${MQTTCLIENT_DIR}/MQTTAsync.c, ${ALL_SOURCE_FILES}}
     MANIFEST_S = mt -manifest ${MQTTDLL_S}.manifest -outputresource: ${MQTTDLL_S}\; 2
     ```

   - Asynchronní, nezabezpečené:

     ```
     MQTTLIB_A = mqttv3a
     MQTTDLL_A = windows_ia32/${MQTTLIB_A} .dll
     SOURCE_FILES_A = ${filter-out ${MQTTCLIENT_DIR}/MQTTClient.c ${MQTTCLIENT_DIR}/
     SSLSocket.c, ${ALL_SOURCE_FILES}}
     MANIFEST_S = mt -manifest ${MQTTDLL_S}.manifest -outputresource: ${MQTTDLL_S}\; 2
     ```

   - Asynchronní zabezpečení:

     ```
     MQTTLIB_AS = mqttv3as
     MQTTDLL_AS = windows_ia32/${MQTTLIB_AS} .dll
     SOURCE_FILES_AS = ${filter-out ${MQTTCLIENT_DIR}/MQTTClient.c, ${ALL_SOURCE_FILES}}
     MANIFEST_S = mt -manifest ${MQTTDLL_S}.manifest -outputresource: ${MQTTDLL_S}\; 2
     ```
5. Definujte kompilátor a volby kompilátoru.

Podívejte se na volby pro různé platformy, které jsou zobrazeny ve volbě Volby sestavení MQTT pro různé platformy.

a) Nastavte Microsoft Visual C++ jako kompilátor.

```
CC = cl
```

b) Přidejte volby před procesorem.

```
CPPFLAGS = /D "WIN32" /D "_UNICODE" /D "UNICODE" /D "_CRT_SECURE_NO_WARNINGS"
```

c) Přidejte volby kompilátoru.

```
CFLAGS = /nologo /c /O2 /W3 /Fd /MD /TC
```

d) Přidejte cesty Include.

```
INC = /I ${MQTTCLIENT_DIR} /I ${MQTTCLIENT_DIR}/..
```

e) Volitelné: Pokud sestavujete zabezpečenou knihovnu, přidejte předzpracovací volbu.

```
CPPFLAGS_S = ${CPPFLAGS} /D "OPENSSL"
```

f) Volitelné: Přidejte hlavičkové soubory produktu OpenSSL , pokud sestavujete zabezpečenou knihovnu.

```
INC_S = ${INC} /I ${OPENSSL_DIR}/inc32/
```

Tip: Soubory záhlaví jsou v souboru `${OPENSSL_DIR}/inc32/openssl`, ale soubor `ssl.h` je obsažen v souboru "`openssl/ssl.h`".

6. Nastavte volby linker a linker.

a) Nastavte Microsoft Visual C++ jako spojovací program.

```
LD = link
```

b) Přidejte volby linker.

```
LINKFLAGS = /nologo /machine:x86 /manifest /dll
```

c) Přidejte cesty ke knihovně.

```
WINLIBS = kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib\
          advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib\
          odbc32.lib odbccp32.lib ws2_32.lib
```

d) Přidejte intermediační výstupní soubory.

```
IMP = /implib: ${@:.dll=.lib}
LIBPDB = /pdb: ${@:.dll=.pdb}
LIBMAP = /map: ${@:.dll=.map}
```

e) Volitelné: Přidejte knihovny produktu OpenSSL , pokud sestavujete zabezpečenou knihovnu.

```
WINLIBS_S = ${WINLIBS} crypt32.lib ssleay32.lib libeay32.lib
```

f) Volitelné: Přidejte cestu ke knihovně produktu OpenSSL , pokud sestavujete zabezpečenou knihovnu.

```
LIBPATH_S = /LIBPATH:${OPENSSL_DIR}/lib
```

7. Definujte čtyři cíle sestavení.

a) Definujte cíl **all** .

Tip: Každý následující řádek, který definuje implementaci cíle, musí začínat znakem tabulátoru.

Cíl "all" sestaví všechny knihovny.

```
all: ${MQTTDLL} ${MQTTDLL_A} ${MQTTDLL_AS} ${MQTTDLL_S}
```

b) Sestavte synchronní, nezabezpečenou knihovnu, `mqttv3c.dll`.

```
${MQTTDLL}: ${SOURCE_FILES}
    -mkdir windows_ia32
    -rm ${CURDIR}/MQTTAsync.obj
    ${CC} ${CPPFLAGS} ${CFLAGS} ${INC} /Fo ${SOURCE_FILES}
    ${LD} ${LINKFLAGS} ${IMP} ${LIBPDB} ${LIBMAP} ${WINLIBS} *.obj /out: ${MQTTDLL}
    ${MANIFEST}
```

Příkaz `-rm ${CURDIR}/MQTTAsync.obj` odstraní všechny `MQTTAsync.obj` vytvořené pro dřívější cíl. `MQTTAsync.obj` a `MQTTClient.obj` se vzájemně vylučují.

c) Sestavte asynchronní, nezabezpečenou knihovnu, `mqttv3a.dll`.

```
${MQTTDLL_A}: ${SOURCE_FILES_A}
    -mkdir windows_ia32
    -rm ${CURDIR}/MQTTClient.obj
    ${CC} ${CPPFLAGS} ${CFLAGS} ${INC} /Fo ${SOURCE_FILES_A}
    ${LD} ${LINKFLAGS} ${IMP} ${LIBPDB} ${LIBMAP} ${WINLIBS} *.obj /out: ${MQTTDLL_A}
    ${MANIFEST_A}
```

Příkaz `-rm ${CURDIR}/MQTTClient.obj` odstraní všechny `MQTTClient.obj` vytvořené pro dřívější cíl. `MQTTAsync.obj` a `MQTTClient.obj` se vzájemně vylučují.

d) Sestavte synchronní, zabezpečenou knihovnu, `mqttv3cs.dll`.

```
${MQTTDLL_S}: ${SOURCE_FILES_S}
    -mkdir windows_ia32
    -rm ${CURDIR}/MQTTAsync.obj
    ${CC} ${CPPFLAGS_S} ${CFLAGS} ${INC_S} /Fo ${SOURCE_FILES}
    ${LD} ${LINKFLAGS} ${IMP} ${LIBMAP} ${WINLIBS_S} ${LIBPATH_S} *.obj. /out: $
{MQTTDLL_S}
    ${MANIFEST_S}
```

Příkaz `-rm ${CURDIR}/MQTTAsync.obj` odstraní všechny `MQTTAsync.obj` vytvořené pro dřívější cíl. `MQTTAsync.obj` a `MQTTClient.obj` se vzájemně vylučují.

e) Sestavte asynchronní, zabezpečenou knihovnu, `mqttv3as.dll`.

```
${MQTTDLL_AS}: ${SOURCE_FILES_AS}
    -rm ${CURDIR}/MQTTClient.obj
    ${CC} ${CPPFLAGS_S} ${CFLAGS} ${INC_S} /Fo ${SOURCE_FILES_AS}
    ${LD} ${LINKFLAGS} ${IMP} ${LIBPDB} ${LIBMAP} ${WINLIBS_S} ${LIBPATH_S} *.obj /out:
$(MQTTDLL_AS}
    $(MANIFEST_AS}
```

Příkaz `-rm $(CURDIR}/MQTTClient.obj` odstraní všechny `MQTTClient.obj` vytvořené pro dřívější cíl. `MQTTAsync.obj` a `MQTTClient.obj` se vzájemně vylučují.

f) Definujte cíl **`clean`** .

Cíl "clean" odebere všechny soubory a adresáře, které jsou generovány souborem Makefile

```
FOTY: čisté
čisté:
    -rm -f *.obj
    -rm -f -r windows_ia32
```

8. Nastavte cestu Windows ke spuštění souboru Makefile.

Nastavte části psané kurzívou tak, aby odpovídaly vaší instalaci.

a) Nastavte prostředí produktu Microsoft Visual Studio.

```
%comspec% /k ""C:\Program Files\Microsoft Visual Studio 9.0\VC\vcvarsall.bat"" x86
```

b) Nastavte proměnnou `Path` tak, aby zahrnovala program make a příkazové prostředí Linux .

```
set path=%path%;C:\Program Files\GnuWin32\bin;C:\cygwin\bin
```

9. Spusťte soubor Makefile.

```
make -f MQTTwin.mak MQTTCLIENT_DIR=sdkroot/SDK/clients/c/mqttv3c/src
```

Tip: Oddělovací znak souboru musí být dopředné lomítko, ne zpětné lomítko.

Výsledky

V adresáři *sdkroot**SDK**clients**c**mqttv3c**src*\windows_ia32 jsou vytvořeny následující soubory.

```
mqttv3a.dll
mqttv3a.dll.manifest
mqttv3a.exp
mqttv3a.lib
mqttv3a.map
mqttv3as.dll
mqttv3as.dll.manifest
mqttv3as.exp
mqttv3as.lib
mqttv3as.map
mqttv3c.dll
mqttv3c.dll.manifest
mqttv3c.exp
mqttv3c.lib
mqttv3c.map
mqttv3cs.dll
mqttv3cs.dll.manifest
mqttv3cs.exp
mqttv3cs.lib
mqttv3cs.map
```

Výpis Makefile `MQTTwin.mak`

```
# Výstup sestavení se vytváří v aktuálním adresáři.
# MQTTCLIENT_DIR musí odkazovat na základní adresář obsahující zdrojový kód klienta MQTT.
# Výchozí hodnota MQTTCLIENT_DIR je aktuální adresář.
# Výchozí hodnota OPENSSL_DIR je sdkroot\openSSL, relativní k adresáři
sdkroot\sdk\clients\c\mqttv3c\src
# OPENSSL_DIR musí odkazovat na základní adresář obsahující sestavení OpenSSL .
# Příklad: make -f MQTTwin.mak MQTTCLIENT_DIR=sdkroot/sdk/clients/c/mqtv3c/src
# Nastavte prostředí sestavení, například:
# %comspec% /k "" C:\Program Files\Microsoft Visual Studio 9.0\VC\vcvarsall.bat"" x86
# set path= %path%; C:\Program Files\GnuWin32\bin;C:\cygwin\bin
ifndef MQTTCLIENT_DIR
  MQTTCLIENT_DIR = ${CURDIR}
endif
VPATH = ${MQTTCLIENT_DIR}
ifndef OPENSSL_DIR
  OPENSSL_DIR = ${MQTTCLIENT_DIR}/../../../../../openssl-1.0.1c
endif

ALL_SOURCE_FILES = ${wildcard ${MQTTCLIENT_DIR}/*.c}

MQTTLIB = mqttv3c
MQTTDLL = windows_ia32/${MQTTLIB} .dll
SOURCE_FILES = ${filter-out ${MQTTCLIENT_DIR}/MQTTAsync.c ${MQTTCLIENT_DIR}/SSLSocket.c, $
{ALL_SOURCE_FILES}}
MANIFEST = mt -manifest ${MQTTDLL}.manifest -outputresource: ${MQTTDLL}\; 2
MQTTLIB_S = mqttv3cs
MQTTDLL_S = windows_ia32/${MQTTLIB_S} .dll
SOURCE_FILES_S = ${filter-out ${MQTTCLIENT_DIR}/MQTTAsync.c, ${ALL_SOURCE_FILES}}
MANIFEST_S = mt -manifest ${MQTTDLL_S}.manifest -outputresource: ${MQTTDLL_S}\; 2
MQTTLIB_A = mqttv3a
MQTTDLL_A = windows_ia32/${MQTTLIB_A} .dll
SOURCE_FILES_A = ${filter-out ${MQTTCLIENT_DIR}/MQTTClient.c ${MQTTCLIENT_DIR}/SSLSocket.c, $
{ALL_SOURCE_FILES}}
MANIFEST_S = mt -manifest ${MQTTDLL_S}.manifest -outputresource: ${MQTTDLL_S}\; 2
MQTTLIB_AS = mqttv3as
MQTTDLL_AS = windows_ia32/${MQTTLIB_AS} .dll
SOURCE_FILES_AS = ${filter-out ${MQTTCLIENT_DIR}/MQTTClient.c, ${ALL_SOURCE_FILES}}
MANIFEST_S = mt -manifest ${MQTTDLL_S}.manifest -outputresource: ${MQTTDLL_S}\; 2

# compiler
CC = cl
CPPFLAGS = /D "WIN32" /D "_UNICODE" /D "UNICODE" /D "_CRT_SECURE_NO_WARNINGS"
```

```
CFLAGS = /nologo /c /O2 /W3 /Fd /MD /TC
INC = /I ${MQTTCLIENT_DIR} /I ${MQTTCLIENT_DIR}/..
CPPFLAGS_S = ${CPPFLAGS} /D "OPENSSL"
INC_S = ${INC} /I ${OPENSSL_DIR}/inc32/

# linker
LD = link
LINKFLAGS = /nologo /machine:x86 /manifest /dll
WINLIBS = kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib\
          advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib\
          odbc32.lib odbccp32.lib ws2_32.lib
IMP = /implib: ${@:.dll=.lib}
LIBPDB = /pdb: ${@:.dll=.pdb}
LIBMAP = /map: ${@:.dll=.map}
WINLIBS_S = ${WINLIBS} crypt32.lib ssleay32.lib libeay32.lib
LIBPATH_S = /LIBPATH:${OPENSSL_DIR}/lib

# targets
all: ${MQTTDLL} ${MQTTDLL_A} ${MQTTDLL_AS} ${MQTTDLL_S}

${MQTTDLL}: ${SOURCE_FILES}
    -mkdir windows_ia32
    -rm ${CURDIR}/MQTTAsync.obj
    ${CC} ${CPPFLAGS} ${CFLAGS} ${INC} /Fo ${SOURCE_FILES}
    ${LD} ${LINKFLAGS} ${IMP} ${LIBPDB} ${LIBMAP} ${WINLIBS} *.obj /out: ${MQTTDLL}
    ${MANIFEST}

${MQTTDLL_A}: ${SOURCE_FILES_A}
    -mkdir windows_ia32
    -rm ${CURDIR}/MQTTClient.obj
    ${CC} ${CPPFLAGS} ${CFLAGS} ${INC} /Fo ${SOURCE_FILES_A}
    ${LD} ${LINKFLAGS} ${IMP} ${LIBPDB} ${LIBMAP} ${WINLIBS} *.obj /out: ${MQTTDLL_A}
    ${MANIFEST_A}

${MQTTDLL_S}: ${SOURCE_FILES_S}
    -mkdir windows_ia32
    -rm ${CURDIR}/MQTTAsync.obj
    ${CC} ${CPPFLAGS_S} ${CFLAGS} ${INC_S} /Fo ${SOURCE_FILES}
    ${LD} ${LINKFLAGS} ${IMP} ${LIBMAP} ${WINLIBS_S} ${LIBPATH_S} *.obj. /out: ${MQTTDLL_S}
    ${MANIFEST_S}

${MQTTDLL_AS}: ${SOURCE_FILES_AS}
    -rm ${CURDIR}/MQTTClient.obj
    ${CC} ${CPPFLAGS_S} ${CFLAGS} ${INC_S} /Fo ${SOURCE_FILES_AS}
    ${LD} ${LINKFLAGS} ${IMP} ${LIBPDB} ${LIBMAP} ${WINLIBS_S} ${LIBPATH_S} *.obj /out: $
(MQTTDLL_AS}
    $(MANIFEST_AS}

FOTY: čisté
čisté:
    -rm -f *.obj
    -rm -f -r windows_ia32
```

Sestavení balíku produktu OpenSSL

Sestavte balík produktu OpenSSL před sestavením zabezpečených knihoven klienta MQTT pro jazyky C, `mqttv3cs` a `mqttv3as`. Sestavení vytvoří knihovny, které jsou nezbytné pro sestavení zabezpečené verze klienta MQTT pro knihovnu C a nástroje pro správu certifikátů produktu OpenSSL .

Než začnete

1. iOS Přizpůsobení souboru Makefile iOS je pro cílová zařízení, která spouští produkt iOS6. Přizpůsobení se může lišit pro dřívější nebo novější verze produktu iOS.
2. Windows Přizpůsobení souboru Makefile produktu Windows se používá pro 32bitová okna.

Informace o této úloze

Stáhněte a nainstalujte balík produktu OpenSSL a veškerý předem vyžadovaný software. Přizpůsobte soubory Makefile produktu OpenSSL a sestavte knihovny OpenSSL pro cílovou platformu. V systémech Windows a Linuxvytvořte také nástroj pro vytváření a správu klíčů produktu OpenSSL .

Postup

1. Nainstalujte balík produktu OpenSSL .
   a) Stáhněte balík produktu OpenSSL z produktu OpenSSL

      Důležité: Stažení a redistribuce balíku produktu OpenSSL podléhá přísnému režimu importu a exportu a podmínkám licencování typu open source. Všimněte si omezení a varování před tím, než se rozhodnete, zda stáhnout balík.

   b) Rozbalte obsah komprimovaného souboru do *sdkroot*.

      Podívejte se na kartu **Novinky** na webu OpenSSL , kde naleznete umístění pro stažení nejnovějšího balíku. Balík je komprimovaný jako soubor tar s příponou `tar.gz`. Při rozbalení vytvoří balík složku nejvyšší úrovně `openssl`*version*, například *openssl-1.0.1c*. Příklady odkazují na cestu ke složce jako %openssl% v Windows a $openssl na iOS; například, v Windows, %openssl% je *sdkroot*\openssl-1.0.1c.

      Tip: Zkontrolujte cestu k adresáři, která je vytvořena extrahováním balíku produktu OpenSSL . Některé balíky mají duplicitní úrovně složky `openssl`*version* .

2. Windows

   Volitelné: V Windowsstáhněte a nainstalujte perl. Viz perl.org.

   V tomto případě byl nástroj Perl stažen z produktu ActivePerl Soubory ke stažení.

3. iOS

   Volitelné: V systému iOSvytvořte tři další adresáře.

   ```
   $ssarm7 ≡ $openssl/arm7
   $sslarm7s ≡ $openssl/arm7s
   $ssli386 ≡ $openssl/i386
   ```

   Pro produkt iOS musíte sestavit balík produktu OpenSSL pro tři různé hardwarové platformy.

4. Generujte soubor Makefile produktu OpenSSL k sestavení balíku produktu OpenSSL pro váš hardware a operační systém.
   a) Otevřete příkazové okno v adresáři %openssl% nebo $openssl .
   b) Spusťte příkaz **Configure** Perl s odpovídajícími parametry.

   - Windows

     ```
     perl Configure VC-WIN32 enable-capieng no-asm no-idea no-mdc2 no-rc5 --prefix=%openssl%
     ms\do_ms.bat
     ```

   - iOS

     ```
     ./Configure BSD-generic32 no-idea no-mdc2 no-rc5 --prefix=$openssl
     ```

5. iOS

   V produktu iOSupravte vygenerovaný soubor Makefile produktu OpenSSL pro různá zařízení Apple .
   a) Vytvořte tři kopie generovaného souboru Makefile, $openssl/Makefile

      ```
      $openssl/Makefile_armv7
      $openssl/Makefile_armv7s
      $openssl/Makefile_i386
      ```

   b) Změňte příkaz "CC=gcc" v každém souboru Makefile.

      Příkaz CC=gcc je na řádku 62 nebo na místě. Změňte jej na následující příkazy:

      $openssl/Makefile_armv7

      ```
      CC=/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/
      Developer/usr/bin/gcc -arch armv7
      ```

$openssl/Makefile_armv7s

```
CC=/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/
Developer/usr/bin/gcc -arch armv7s
```

$openssl/Makefile_i386

```
CC=/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/
Developer/usr/bin/gcc -arch i386
```

c) Změňte příkaz "CFLAG=..." v každém souboru Makefile.

Příkaz je na řádku 63 nebo v místě (rozbito na tři řádky pro čitelnost):

```
CFLAG= -DOPENSSL_THREADS -pthread -D_THREAD_SAFE
-D_REENTRANT -DDSO_DLFCN -DHAVE_DLFCN_H -DTERMIOS
-O3 -fomit-frame-pointer -Wall
```

Umístění zobrazených SDKs je závislé na volbách instalace produktu Xcode . Verze SDK je závislá na úrovni operačního systému, pro kterou sestavujete soubor Makefile.

Simulátor iPhone
Soubor Makefile simulátoru produktu iPhone je $openssl/Makefile_i386.

```
CFLAG= -isysroot /Applications/Xcode.app/Contents/Developer/Platforms/
iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator6.0.sdk
-DOPENSSL_THREADS -pthread -D_THREAD_SAFE
-D_REENTRANT -DDSO_DLFCN -DHAVE_DLFCN_H -DTERMIOS
-O3 -fomit-frame-pointer -Wall
```

iOS
Soubory Makefile produktu iOS jsou $openssl/Makefile_arm7 a $openssl/Makefile_arm7s.

```
CFLAG= -isysroot /Applications/Xcode.app/Contents/Developer/Platforms/
iPhoneOS.platform/Developer/SDKs/iPhoneOS6.0.sdk
-DOPENSSL_THREADS -pthread -D_THREAD_SAFE
-D_REENTRANT -DDSO_DLFCN -DHAVE_DLFCN_H -DTERMIOS
-O3 -fomit-frame-pointer -Wall
```

6. Spusťte vygenerovaný soubor Makefile.

- Windows

```
nmake -clean
nmake -f ms\nt.mak
nmake -f ms\nt.mak install
```

- iOS V systému iOS:

```
make clean
make -f $openssl/Makefile_arm7
mv $openssl/libcrypto.a $ssarm7/libcrypto.a
mv $openssl/libssl.a $ssarm7/libssl.a
make clean
make -f $openssl/Makefile_arm7s
mv $openssl/libcrypto.a $sslarm7s/libcrypto.a
mv $openssl/libssl.a $sslarm7s/libssl.a
make clean
make -f $openssl/Makefile_i386
mv $openssl/libcrypto.a $ssli386/libcrypto.a
mv $openssl/libssl.a $ssli386/libssl.a
```

Výsledky

Sestavení generuje sdílené knihovny, knihovny a soubory záhlaví, které jsou nezbytné pro sestavení zabezpečených verzí knihovny klienta MQTT pro C.

Začínáme s klientem MQTT pro C na systému iOS

Naučte se, jak získat aplikace iOS pro výměnu zpráv se serverem MQTT . Pro použití na zařízeních iOS (tj. iPhone a iPad) musíte sestavit knihovnu klienta MQTT pro C ze zdrojového kódu, který je poskytován jako součást sady SDK softwaru MQTT .

Než začnete

1. Odkaz na iOS Dev Center a víme, jak vyvíjet aplikace pro produkt iOS.
2. Získejte Apple Mac s OS X 10.8.2nebo novější, chcete-li spustit integrované vývojové prostředí Xcode (IDE).
3. (Volitelné) Nakonfigurujte vývojové prostředí C na serveru Windows nebo Linux. Je užitečné sestavit a spustit ukázkové aplikace C klienta MQTT v produktu Windows nebo Linux před vývojem aplikace MQTT iOS . Případně můžete prostudovat ukázkový zdrojový kód bez sestavení ukázek.
4. Podporované a referenční platformy klienta MQTT pro jazyk C viz Systémové požadavky pro IBM Mobile Messaging and M2M Client Pack.
5. Je-li mezi vaším klientem a serverem brána firewall, zkontrolujte, zda neblokuje provoz produktu MQTT .

Informace o této úloze

Postup vás provede následujícími kroky:

1. Informace o programování pro produkt MQTT prostřednictvím studia, sestavení a spuštění ukázkových aplikací klienta MQTT a klientských knihoven produktu MQTT pro jazyk C.
2. Nainstalujte vývojové prostředí produktu Xcode pro produkt iOS v prostředí Apple Mac.
3. Chcete-li sestavit klienta MQTT pro knihovny C pro zařízení iOS , použijte úlohu “Sestavení klienta MQTT pro knihovny C” na stránce 30 .

Postup

1. Zvolte server MQTT , ke kterému se můžete připojit k aplikaci klienta.

   Server musí podporovat protokol MQTT version 3.1 . Všechny servery MQTT z IBM to dělají, včetně IBM WebSphere MQ a IBM MessageSight. Viz “Začínáme se servery MQTT” na stránce 135.
2. Stáhněte si produkt Mobile Messaging and M2M Client Pack a nainstalujte sadu SDK produktu MQTT .

   K dispozici není žádný instalační program, stačí jen rozbalit stažený soubor.

   a. Stáhněte si Mobile Messaging and M2M Client Pack.

   b. Vytvořte složku, do které budete instalovat sadu SDK.

      Je možné, že budete chtít pojmenovat složku `MQTT`. Cesta k této službě je zde označována jako *`sdkroot`*.

   c. Rozbalte komprimovaný obsah souboru Mobile Messaging and M2M Client Pack do *`sdkroot`*. Rozšíření vytváří adresářový strom, který začíná v *`sdkroot`*\SDK.
3. Volitelné: Seznamte se s rozhraním API produktu MQTT tím, že studujete Ukázková aplikace C klienta MQTT.

   a) Sestavte ukázkovou ukázkovou aplikaci C `MQTTV3sample.c` klienta MQTT pro produkt Windows nebo Linux. Viz “Začínáme s klientem MQTT pro jazyk C” na stránce 25.

   b) Připojte se k serveru MQTT version 3 a publikujte a přihlaste se k tématům na serveru.

   c) Prostudujte si zdrojový kód a dokumentaci rozhraní API produktu MQTT . Pro odkazy na dokumentaci ke klientským rozhraním API pro knihovny klienta MQTT viz Odkaz na programování klienta MQTT.

      Naučte se vytvářet a pokračovat v klientech produktu MQTT a publikovat a odebírat témata produktu MQTT prostřednictvím studování synchronní ukázky. Synchronní ukázka je jednodušší než

asynchronní ukázka. Pokud jste dříve neprogramovali MQTT , zapište synchronní program MQTT , abyste se seznámili s programovacím modelem MQTT a rozhraním API.

d) Sestavte asynchronní knihovnu klienta MQTT pro C na Windows nebo Linux. Viz “Sestavení klienta MQTT pro knihovny C” na stránce 30.

e) Sestavte a spusťte ukázkovou aplikaci jazyka C pro asynchronní publikování a odběr klienta produktu MQTT .

f) Study the MQTT client sample asynchronous C app source code and MQTT reference documentation.

Chcete-li pro mobilní zařízení napsat aplikaci MQTT , musíte použít asynchronní rozhraní. Dobře napsané aplikace, které volají asynchronní rozhraní, jsou více citlivé a protáhnou životnost baterie, než aplikace napsané pro synchronní rozhraní.

Asynchronní rozhraní má dva stupně asynchnicity:

i) První stupeň je odblokování aplikace, zatímco knihovna klienta MQTT čeká na publikování ze serveru.

ii) Druhý stupeň je odblokování aplikace, zatímco se knihovna klienta připojuje k serveru, vytváří odběry a publikují publikace.

4. Stáhněte a nainstalujte vývojové nástroje produktu iOS .
   a. Přihlaste se s ID uživatele, který má oprávnění k administraci.
   b. Zkontrolujte Apple Mac ve verzi 10.8.2 nebo novější.
   c. Přejděte na webový server Xcode , chcete-li stáhnout produkt Xcode z obchodu s aplikacemi produktu Mac .
   d. Nainstalujte produkt Xcode, prostředí příkazového řádku a simulátor.

   Pokud má aplikace Mac App Store více verzí simulátoru, vyberte verzi, která je kompatibilní s úrovní iOS , kterou cílíte pro vaši aplikaci.
5. Sestavte knihovny klienta MQTT pro C na systému iOS. Viz “Sestavení klienta MQTT pro knihovny C” na stránce 30.

Jak pokračovat dále

1. Ověřte knihovny klienta MQTT pro C, které jste vytvořili:
   a. Použijte vývojové prostředí produktu Xcode ke kompilaci asynchronního prostoru Ukázková aplikace C klienta MQTTa připojte se k nezabezpečené asynchronní knihovně klienta MQTT pro C.
   b. Použijte vývojové prostředí produktu Xcode ke spuštění ukázkové ukázkové aplikace C klienta MQTT na zařízení iOS . Připojte vzorek k serveru MQTT version 3 , který jste nakonfigurovali, viz “Konfigurace služby MQTT z příkazového řádku” na stránce 139.
2. Create an MQTT client C app for iOS. Příklady kódování pro asynchronní aplikaci C se mohou ukázat jako užitečné. Příklady jsou `MQTTV3ASample.c` a `MQTTV3ASSample.c` v *sdkroot*`\SDK\clients\c\samples`. Jako cvičení začněte implementací ukázky publikování/odběru MQTT .

   Tip: Chcete-li získat představu o tom, jak by aplikace mohla vypadat a co dělat, podívejte se na snímky obrazovky Ukázková aplikace C klienta MQTT. Viz “Začínáme s produktem Klient MQTT pro produkt Java on Android” na stránce 17.

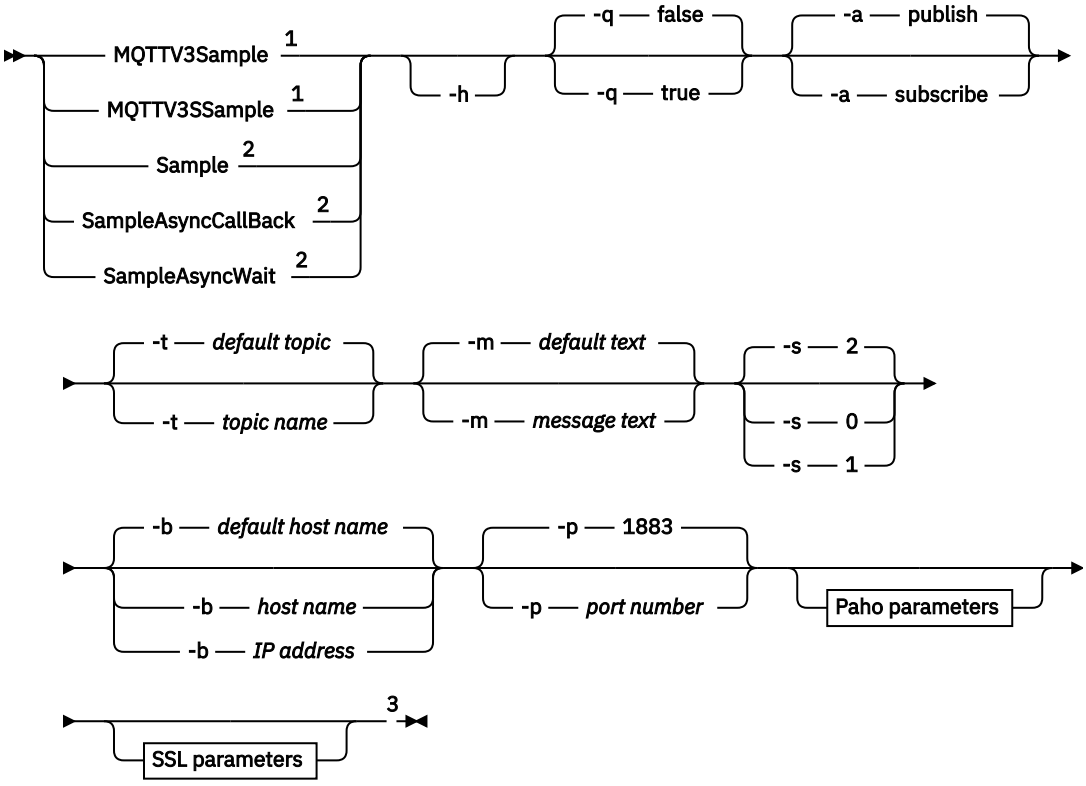
Ukázkové programy příkazového řádku produktu MQTT

Syntaxe a parametry ukázkových programů příkazového řádku MQTT .

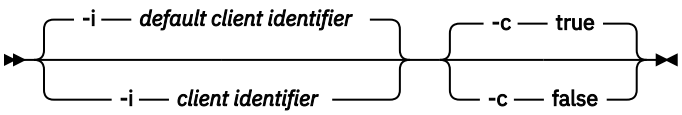
Účel

Publikovat a přihlásit se k odběru tématu.

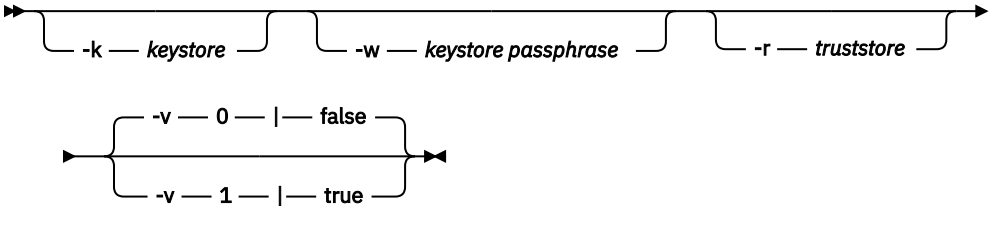
Syntax



Paho parameters



SSL parameters



Poznámky:

[1] IBM WebSphere MQ sample
[2] Paho sample
[3] *Not* `MQTTV3Sample`.

Parametry

`-h`
Vytisknout tento text nápovědy a ukončit

`-q`
Nastavte tichý režim místo použití výchozího režimu false.

`-a publish|subscribe`
Nastavte akci na `publish` nebo `subscribe`, místo toho, abyste předpokládali výchozí akci publikování.

-t ***název tématu***
Publikujte nebo odebíráte produkt *topic name*místo publikování nebo přihlášení k odběru výchozího tématu. Výchozí témata jsou následující:

Ukázky Paho

Publikovat
`Sample/Java/v3`

Odebírat
`Sample/#`

Ukázky produktu IBM WebSphere MQ

Publikovat
`MQTTV3Sample/Java/v3` nebo `MQTTV3Sample/C/v3`

Odebírat
`MQTTV3Sample/#`

-m ***text zprávy***
Publikovat *message text* místo odeslání výchozího textu. Výchozí text je buď "`Message from MQTTv3 C client`", nebo "`Message from MQTTv3 Java client`"

-s 0|1|2
Nastavte kvalitu služby (QoS) místo použití výchozí hodnoty QoS, 2.

-b ***název hostitele***
Připojte se k produktu *host name* nebo k adrese IP místo připojení k výchozímu názvu hostitele. Výchozí název hostitele pro ukázky Paho je `m2m.eclipse.org`. Pro ukázky produktu IBM WebSphere MQ je to `localhost`.

-p ***číslo portu***
Použijte port *port number* místo použití výchozího portu 1883.

Parametry příkazu Paho

-i ***identifikátor klienta***
Nastavte identifikátor klienta na *client identifier.* . Výchozí identifikátor klienta je `SampleJavaV3_"+action`, kde `action` je `publish` nebo `subscribe`.

-c true|false
Nastavte příznak čisté relace. Předvolba je `true`: odběry nejsou trvalé.

Parametry zabezpečení SSL

-k ***úložiště klíčů***
Nastavte cestu k úložišti klíčů obsahujícímu soukromý klíč, který identifikuje klienta na serveru *keystore*. V případě ukázek C se jedná o soubor PEM (Privacy-Enhanced Mail). Pro ukázky Java je to úložiště klíčů Java (JKS).

-w ***Heslo úložiště klíčů***
Nastavte přístupovou frázi, abyste autorizovali klienta pro přístup k úložišti klíčů serveru *keystore passphrase*.

-r ***úložiště údajů o důvěryhodnosti***
Nastavte cestu k úložišti klíčů obsahujícímu veřejné klíče serverů MQTT , kterým klient důvěřuje, do produktu *truststore*. Úložiště klíčů je soubor PEM (Privacy-Enhanced Mail). V případě ukázek C se jedná o soubor PEM (Privacy-Enhanced Mail). Pro ukázky Java je to úložiště klíčů Java (JKS).

-v 0|false|1|true
Nastavte volbu ověření na 1 | `true` , chcete-li vyžadovat certifikát serveru. Předvolba je 0 | `false`: certifikát serveru se nekontroluje. Kanál SSL je vždy šifrován.

Nastavte volbu na `0` | `1` pro programy v jazyce C a `true` | `false` pro programy Java .

Související úlohy
“Začínáme s klientem produktu MQTT pro produkt Java” na stránce 11

s ukázkovými aplikacemi produktu Java pro ukázkové aplikace produktu MQTT s použitím produktu buď IBM MessageSight , nebo IBM WebSphere MQ jako server MQTT. Ukázkové aplikace používají knihovnu klienta ze sady nástrojů pro vývoj softwaru MQTT (SDK) od společnosti IBM. Ukázková aplikace produktu `SampleAsyncCallBack` je modelem pro zápis aplikací produktu MQTT pro operační systém Android a dalších operačních systémů založených na události.

"Začínáme s klientem MQTT pro jazyk C" na stránce 25
Vstaňte a běží s ukázkovým klientem MQTT pro C na libovolné platformě, na které můžete kompilovat zdroj C. Ověřte, že můžete spustit ukázkového klienta MQTT pro prostředí C s produktem buď IBM MessageSight , nebo IBM WebSphere MQ jako server MQTT.

"Sestavení klienta MQTT pro knihovny C" na stránce 30
Chcete-li sestavit klienta produktu MQTT pro knihovny jazyka C, postupujte podle následujících kroků. Téma obsahuje přepínače kompilace a linkování pro řadu platforem a příklady sestavení knihoven na systémech iOS a Windows.

Zabezpečení produktu MQTT

Tři koncepty jsou základní pro zabezpečení produktu MQTT : identita, ověřování a autorizace. Identita je o pojmenování klienta, který je autorizován a kterému je uděleno oprávnění. Ověřování je o prokazování identity klienta a autorizace se týká správy práv, která jsou klientovi udělena.

Zkuste ukázky zabezpečení

- "Sestavení a spuštění zabezpečeného produktu Ukázková aplikace klienta MQTT Java" na stránce 54
- "Připojení ukázkové aplikace Java klienta MQTT k produktu Android přes SSL" na stránce 62
- "Authenticating an MQTT client Java app with JAAS" na stránce 72
- V 7.5.0.1 "Připojení Klient systému zpráv MQTT pro produkt JavaScript přes SSL a WebSockets" na stránce 77
- "Sestavení a spuštění zabezpečeného produktu Ukázková aplikace C klienta MQTT" na stránce 85

Identita

Identifikujte klienta MQTT pomocí jeho identifikátoru klienta, ID uživatele nebo veřejného digitálního certifikátu. Jeden nebo druhý z těchto atributů definuje identitu klienta. Server MQTT ověřuje certifikát odeslaný klientem s protokolem SSL nebo identitu klienta pomocí hesla nastavovacího klientem. Server řídí, ke kterým prostředkům může klient přistupovat, a to na základě identity klienta.

Server MQTT se identifikuje klientovi se svou IP adresou a digitálním certifikátem. Klient MQTT používá protokol SSL k ověřování identity certifikátu odeslaného serverem. V některých případech používá název DNS serveru k ověření serveru, který jej odeslal, je certifikát registrován jako držitel certifikátu.

Nastavte identitu klienta jedním z následujících způsobů:

Identifikátor klienta
Třída `MqttClient` (`MQTTClient_create` nebo `MQTTAsync_create` v C) nastavuje identifikátor klienta. Volejte konstruktor třídy, abyste nastavili identifikátor klienta jako parametr, nebo vraceli náhodně generovaný identifikátor klienta. Identifikátor klienta musí být jedinečný pro všechny klienty, kteří se připojují k serveru, a nesmí být stejný jako název správce front na serveru. Všichni klienti musí mít identifikátor klienta, i když se nepoužívají pro kontrolu identity. Viz "Identifikátor klienta" na stránce 127.

Jméno uživatele
Třída `MqttClient` (`MQTTClient_create` nebo `MQTTAsync_create` v C) nastavuje ID uživatele klienta jako atribut `MqttConnectOptions` (`MqttClient_ConnectOptions` in C). ID uživatele nemusí být pro klienta jedinečné.

Digitální certifikát klienta
Digitální certifikát klienta je uložen v úložišti klíčů klienta. Umístění úložiště klíčů závisí na klientovi:

- **Java**

  Nastavte umístění a vlastnosti úložiště klíčů klienta voláním metody `setSSLProperties` produktu `MqttConnectOptions` a předáním vlastností úložiště klíčů. Viz téma Úpravy protokolu SSL na Example.java. Nástroj **keytool** spravuje klíče Java a úložiště klíčů.

- **C**

  `MQTTClient_create` nebo `MQTTAsync_create` nastavuje vlastnosti úložiště klíčů jako atributy `MQTTClient_SSLOptions ssl_opts`. Nástroj **openSSL** vytváří a spravuje klíče a úložiště klíčů, ke kterým má přístup klient MQTT pro C.

- **Android**

  Správa úložiště klíčů zařízení Android z nabídky **Nastavení** > **Zabezpečení** . Načtěte nové certifikáty z karty SD.

Nastavte identitu serveru tím, že uložíte jeho soukromý klíč do úložiště klíčů serveru:

IBM WebSphere MQ
Úložiště klíčů serveru MQTT je atributem kanálu telemetrie, ke kterému je klient připojen.

Nastavte umístění úložiště klíčů a atributy s příkazem IBM WebSphere MQ Explorernebo příkazem **DEFINE CHANNEL** ; viz DEFINE CHANNEL (MQTT). Úložiště klíčů může sdílet více kanálů.

Ověřování

Klient produktu MQTT může ověřit identitu serveru MQTT , ke kterému se připojuje, a server může ověřit klienta, který se k němu připojuje.

Klient ověřuje server pomocí protokolu SSL. Server MQTT ověřuje klienta pomocí protokolu SSL nebo pomocí hesla, případně obou.

Pokud klient autentizuje server, ale server neověřuje klienta, klient je často znám jako anonymní klient. Je běžné zavést anonymní připojení klienta přes SSL a pak autentizovat klienta s heslem zašifrovaným relací SSL. Je mnohem obvyklejší ověření klienta s heslem, než s certifikátem klienta, kvůli distribuci certifikátu a problému správy. Je pravděpodobné, že budete hledat certifikáty klientů použité ve vysoce hodnotovém zařízení, jako jsou bankomaty a čipové a kolíkové stroje a ve vlastních zařízeních, jako jsou například inteligentní elektroměry.

Ověření serveru pomocí klienta
Klient produktu MQTT ověřuje, zda je připojen ke správnému serveru, ověřením certifikátu serveru pomocí protokolu SSL. Tato forma ověření je vám známa, když procházíte webový server přes protokol HTTPS.

Server odešle svůj veřejný certifikát, podepsaný certifikační autoritou, na klienta. Klient používá veřejný klíč vydavatele certifikátů k ověření podpisu certifikační autority na certifikátu serveru. Také kontroluje, zda je certifikát aktuální. Tyto kontroly stanoví, že osvědčení je platné.

Certifikáty certifikační autority, často nazývané kořenové certifikáty, jsou uloženy v úložišti údajů o důvěryhodnosti klienta:

- **Java**

  Volejte metodu `setSSLProperties` produktu `MqttConnectOptions` a předejte vlastnosti úložiště údajů o důvěryhodnosti, abyste nastavili umístění a vlastnosti úložiště údajů o důvěryhodnosti klienta. Viz téma Úpravy protokolu SSL na Example.java. Správa certifikátů a důvěryhodných úložišť pomocí nástroje **keytool** .

- **C**

  `MQTTClient_create` nebo `MQTTAsync_create` nastavuje vlastnosti úložiště údajů o důvěryhodnosti jako atributy produktu `MQTTClient_SSLOptions ssl_opts`. Správa certifikátů a důvěryhodných úložišť pomocí nástroje **openSSL** .

- **Android**

Správa úložiště údajů o důvěryhodnosti zařízení Android z nabídky **Nastavení** > **Zabezpečení** . Načtěte nové kořenové certifikáty z karty SD.

Ověření klienta pomocí serveru

Server MQTT ověřuje, zda je připojen ke správnému klientovi ověřením certifikátu klienta pomocí protokolu SSL, nebo ověřením identity klienta pomocí hesla.

Ověřuje klienta s heslem, které posílá klient na server v záhlaví MQTT protocol . Server se může rozhodnout pro ověření identifikátoru klienta, ID uživatele nebo certifikátu s heslem. Záleží na serveru. Obvykle server ověřuje ID uživatele. Ověřte hesla přes připojení přes SSL, které bylo zabezpečeno ověřením serveru, abyste se vyhnuli posílání hesel v čistém umístění.

- **IBM WebSphere MQ**

  Produkt IBM WebSphere MQ ověřuje certifikát klienta pomocí protokolu SSL. Uložte kořenové certifikáty v úložišti klíčů produktu IBM WebSphere MQ Telemetry . Ověření platnosti certifikátu klienta můžete provést pouze jako součást vzájemného ověření SSL. To znamená, že musíte poskytnout klientovi veřejný certifikát serveru a zároveň poskytnout server s veřejným certifikátem klienta.

  Produkt IBM WebSphere MQ Telemetry používá stejné úložiště pro vlastní soukromý i veřejný certifikát i pro jiné veřejné certifikáty, jako jsou například kořenové certifikáty poskytované certifikačních autorit.

  Nastavte umístění úložiště klíčů a atributy s příkazem IBM WebSphere MQ Explorernebo příkazem **`DEFINE CHANNEL`** ; viz DEFINE CHANNEL (MQTT). Úložiště klíčů může sdílet více kanálů.

  Produkt IBM WebSphere MQ ověřuje ID uživatele klienta nebo identifikátor klienta vyvoláním ověřovací a autorizační služby produktu Java (JAAS).

  Konfigurujte službu JAAS v konfiguračním oddílu `MQXRConfig` , který je uložen v souboru `jaas.config` . Soubor je uložen v adresáři `qmgrs\`*`QmgrName`*`\mqxr` v cestě k datům IBM WebSphere MQ .

  Zkontrolujte pravost klienta tím, že napíšete metodu `login` pro `JAASLoginModule`.
  Viz “Konfigurace kanálu JAAS kanálu telemetrie” na stránce 114.

  Příkaz IBM WebSphere MQ Telemetry předává metodě `JAASLoginModule.login` následující parametry:

  – Jméno uživatele
  – Heslo
  – Identifikátor klienta
  – identifikátor sítě
  – Název kanálu
  – ValidPrompts

Autorizace

Autorizace není součástí produktu MQTT protocol. Je poskytován serverem MQTT . Co je oprávněné, závisí na tom, co server dělá. Servery MQTT jsou zprostředkovatelé publikování/odběru a užitečná MQTT autorizační pravidla řídí, kteří klienti se mohou připojit k serveru a která témata může klient publikovat nebo odebírat. Pokud server MQTT může spravovat server, více autorizačních pravidel řídí, kteří klienti mohou spravovat různé aspekty serveru.

Počet možných klientů je obrovský, takže není možné autorizovat každého klienta zvlášť. Server MQTT bude mít prostředky pro seskupení klientů podle profilů nebo skupin.

Identita klienta, z pohledu přístupu a autorizace, není něco, co je pro klienta MQTT jedinečné. Nezaměňovat identitu klienta s identifikátorem klienta. Mohou být stejné, ale jsou běžně odlišné. Například, pravděpodobně máte jméno uživatele, které je společné přes určitý počet služeb, a některé

z těchto služeb spoluoperují v "jednotném přihlášení". Server podnikových měřítek MQTT pravděpodobně zavolá autorizační službu, která nabízí společné identity a oprávnění pro různé aplikace.

IBM WebSphere MQ

Produkt IBM WebSphere MQ má přídavnou autorizační službu. Výchozí autorizační služba, která je poskytnuta na serveru Windows a Linux je správce oprávnění k objektu (OAM). Viz Řízení přístupu k objektům pomocí OAM v systémech UNIX, Linux a Windows. Přidružuje ID uživatelů operačního systému a skupiny k operacím na objektech IBM WebSphere MQ , jako jsou témata a fronty.

Kanál telemetrie je možné konfigurovat pro přístup k produktu IBM WebSphere MQ s pevným ID uživatele. Tímto způsobem je nastaven ukázkový kanál. Nebo máte přístup k produktu IBM WebSphere MQ s ID uživatele nastaveným klientem MQTT . Autorizace klientů MQTT pro přístup k objektům produktu WebSphere MQ popisuje způsoby nastavení produktu IBM WebSphere MQ Telemetry za účelem dosažení hrubého, středního a pokutového řízení přístupu klienta s vysokou úrovní granularity.

Související úlohy

“Sestavení a spuštění zabezpečeného produktu Ukázková aplikace C klienta MQTT” na stránce 85
Na základě příkladu Windows můžete začít pracovat se zabezpečeným vzorovým aplikací C na libovolném operačním systému, pro který můžete kompilovat zdroj C. Ověřte, že můžete spustit ukázkovou aplikaci C na serveru buď IBM MessageSight , nebo IBM WebSphere MQ jako server MQTT.

“Sestavení a spuštění zabezpečeného produktu Ukázková aplikace klienta MQTT Java” na stránce 54
Na základě příkladu Windows můžete začít pracovat se zabezpečenou ukázkovou aplikací Java na serveru buď IBM MessageSight , nebo IBM WebSphere MQ jako server MQTT. Aplikaci Klient MQTT pro produkt Java můžete spustit na libovolné platformě s JSE 1.5 nebo vyšším, které je "Java Kompatibilní"

“Připojení Klient systému zpráv MQTT pro produkt JavaScript přes SSL a WebSockets” na stránce 77
Webovou aplikaci bezpečně připojte k produktu IBM WebSphere MQ pomocí ukázkových HTML stránek Klient systému zpráv MQTT pro produkt JavaScript s SSL a WebSocket protocol.

“Připojení ukázkové aplikace Java klienta MQTT k produktu Android přes SSL” na stránce 62
Vstupte a vyběhněte s ukázkovým klientem Android MQTT připojeným k produktu IBM WebSphere MQ přes SSL.

“Authenticating an MQTT client Java app with JAAS” na stránce 72
Naučte se, jak ověřit klienta pomocí produktu JAAS. Complete the steps in this task to modify the sample program `JAASLoginModule.java` and configure IBM WebSphere MQ to authenticate an MQTT client Java app with JAAS.

Sestavení a spuštění zabezpečeného produktu Ukázková aplikace klienta MQTT Java

Na základě příkladu Windows můžete začít pracovat se zabezpečenou ukázkovou aplikací Java na serveru buď IBM MessageSight , nebo IBM WebSphere MQ jako server MQTT. Aplikaci Klient MQTT pro produkt Java můžete spustit na libovolné platformě s JSE 1.5 nebo vyšším, které je "Java Kompatibilní"

Než začnete

1. Musíte mít přístup k serveru MQTT version 3.1 , který podporuje protokol MQTT protocol přes SSL.
2. Je-li mezi vaším klientem a serverem brána firewall, zkontrolujte, zda neblokuje provoz produktu MQTT .
3. Aplikaci Klient MQTT pro produkt Java můžete spustit na libovolné platformě s JSE 1.5 nebo vyšším, které je "Java Kompatibilní". Viz téma Systémové požadavky pro IBM Mobile Messaging and M2M Client Pack.
4. Kanály SSL musí být spuštěny.

Informace o této úloze

Tento článek vám názorně ukazuje, jak kompilovat a spustit zabezpečení Ukázková aplikace klienta MQTT Java v systému Windows z příkazového řádku.

Zabezpečte kanál SSL buď pomocí klíčů podepsanými certifikační autoritou, nebo pomocí klíčů podepsaným držitelem.

Postup

1. Zvolte server MQTT , ke kterému se můžete připojit k aplikaci klienta.

   Server musí podporovat protokol MQTT version 3.1 přes SSL. Všechny servery MQTT z IBM to dělají, včetně IBM WebSphere MQ a IBM MessageSight. Viz “Začínáme se servery MQTT” na stránce 135.
2. Volitelné: Nainstalujte vývojovou sadu produktu Java (JDK) ve verzi 7 nebo novější.

   verze 7 je vyžadován ke spuštění příkazu **keytool** pro certifikaci certifikátů. Pokud certifikáty certifikovat nechcete, není třeba mít sadu JDK verze 7.
3. Stáhněte si produkt Mobile Messaging and M2M Client Pack a nainstalujte sadu SDK produktu MQTT .

   K dispozici není žádný instalační program, stačí jen rozbalit stažený soubor.

   a. Stáhněte si Mobile Messaging and M2M Client Pack.

   b. Vytvořte složku, do které budete instalovat sadu SDK.

      Je možné, že budete chtít pojmenovat složku MQTT. Cesta k této službě je zde označována jako *sdkroot*.

   c. Rozbalte komprimovaný obsah souboru Mobile Messaging and M2M Client Pack do *sdkroot*. Rozšíření vytváří adresářový strom, který začíná v *sdkroot*\SDK.
4. Vytvořte a spusťte skripty pro generování dvojic klíčů a certifikátů a nakonfigurujte produkt IBM WebSphere MQ jako server MQTT .

   Postupujte podle kroků v části “Generování klíčů a certifikátů” na stránce 95 a vytvořte a spusťte skripty. Tyto skripty jsou také vypsány v “Ukázkové skripty pro konfiguraci certifikátů SSL pro produkt Windows” na stránce 57.
5. Zkontrolujte, zda jsou kanály zabezpečení SSL spuštěny a nastaveny dle očekávání.

   V IBM WebSphere MQ zadejte následující příkaz do okna příkazového řádku:

   - Linux

     ```
     echo 'DISPLAY CHSTATUS(SSL*) CHLTYPE(MQTT) ALL' | runmqsc MQXR_SAMPLE_QM
     echo 'DISPLAY CHANNEL(SSL*) CHLTYPE(MQTT) ALL' | runmqsc MQXR_SAMPLE_QM
     ```

   - Windows

     ```
     echo DISPLAY CHSTATUS(SSL*) CHLTYPE(MQTT) ALL | runmqsc MQXR_SAMPLE_QM
     echo DISPLAY CHANNEL(SSL*) CHLTYPE(MQTT) ALL | runmqsc MQXR_SAMPLE_QM
     ```
6. Vytvořte skripty pro sestavení a spuštění zabezpečeného produktu Ukázková aplikace klienta MQTT Java.

   a) Vytvořte a spusťte `ssjavaclient.bat` , abyste otestoval kanál SSL, který je zabezpečen s certifikáty s vlastním podpisem.

   b) Vytvořte a spusťte `cajavaclient.bat` , abyste otestoval kanál SSL, který je zabezpečen pomocí certifikátů podepsaných certifikační autoritou.

Skripty pro spuštění zabezpečeného klienta Java MQTT

Spusťte skripty v produktu “Ukázkové skripty pro konfiguraci certifikátů SSL pro produkt Windows” na stránce 57 před spuštěním těchto skriptů.

MQTT zabezpečeného klienta Java s certifikáty s vlastním podpisem.

Spusťte tento skript s certifikáty s vlastním podpisem, které jste vytvořili spuštěním skriptu `sscerts.bat` .

```
@echo off
setlocal
cd %jsamppath%
set classpath=
set JAVADIR=C:\Program Files\IBM\Java70\bin
cd %
"%JAVADIR%\javac"
-cp ..\org.eclipse.paho.client.mqttv3.jar .\org\eclipse\paho\sample\mqttv3app\Sample.java
ping -n 2 127.0.0.1 > NUL 2>&1
start "Sample Subscriber" "%JAVADIR%\java" -cp  .;..\org.eclipse.paho.client.mqttv3.jar
org.eclipse.paho.sample.mqttv3app.Sample -a subscribe -b %host% -p %sslportopt% -k
%cltjkskeystore% -w %cltjkskeystorepass% -r %cltsrvjkstruststore% -v true
@rem Sleep for 2 seconds
ping -n 2 127.0.0.1 > NUL 2>&1
"%JAVADIR%\java" -cp  .;..\org.eclipse.paho.client.mqttv3.jar
org.eclipse.paho.sample.mqttv3app.Sample -b %host% -p %sslportopt% -k %cltjkskeystore% -w
%cltjkskeystorepass% -r %cltsrvjkstruststore% -v true
pause
ping -n 2 127.0.0.1 > NUL 2>&1
start "Sample Subscriber" "%JAVADIR%\java" -cp  .;..\org.eclipse.paho.client.mqttv3.jar
org.eclipse.paho.sample.mqttv3app.Sample -a subscribe -b %host% -p %sslportreq% -k
%cltjkskeystore% -w %cltjkskeystorepass% -r %cltsrvjkstruststore% -v true
@rem Sleep for 2 seconds
ping -n 2 127.0.0.1 > NUL 2>&1
"%JAVADIR%\java" -cp  .;..\org.eclipse.paho.client.mqttv3.jar
org.eclipse.paho.sample.mqttv3app.Sample -b %host% -p %sslportreq% -k %cltjkskeystore% -w
%cltjkskeystorepass% -r %cltsrvjkstruststore% -v true
pause
endlocal
```

Obrázek 14. `ssjavaclient.bat`

Spusťte zabezpečeného klienta Java MQTT s certifikáty podepsanými certifikační autoritou.

Spusťte tento skript spolu s certifikáty podepsanými certifikační autoritou, které jste vytvořili spuštěním skriptu `cacerts.bat` .

```
@echo off
setlocal
cd %jsamppath%
set classpath=
set JAVADIR=C:\Program Files\IBM\Java70\bin
cd %
"%JAVADIR%\javac"
-cp ..\org.eclipse.paho.client.mqttv3.jar .\org\eclipse\paho\sample\mqttv3app\Sample.java
ping -n 2 127.0.0.1 > NUL 2>&1
start "Sample Subscriber" "%JAVADIR%\java" -cp  .;..\org.eclipse.paho.client.mqttv3.jar
org.eclipse.paho.sample.mqttv3app.Sample -a subscribe -b %host% -p %sslportopt% -k
%cltjkskeystore% -w %cltjkskeystorepass% -r %cltcajkstruststore% -v true
@rem Sleep for 2 seconds
ping -n 2 127.0.0.1 > NUL 2>&1
"%JAVADIR%\java" -cp  .;..\org.eclipse.paho.client.mqttv3.jar
org.eclipse.paho.sample.mqttv3app.Sample -b %host% -p %sslportopt% -k %cltjkskeystore% -w
%cltjkskeystorepass% -r %cltcajkstruststore% -v true
pause
ping -n 2 127.0.0.1 > NUL 2>&1
start "Sample Subscriber" "%JAVADIR%\java" -cp  .;..\org.eclipse.paho.client.mqttv3.jar
org.eclipse.paho.sample.mqttv3app.Sample -a subscribe -b %host% -p %sslportreq% -k
%cltjkskeystore% -w %cltjkskeystorepass% -r %cltcajkstruststore% -v true
@rem Sleep for 2 seconds
ping -n 2 127.0.0.1 > NUL 2>&1
"%JAVADIR%\java" -cp  .;..\org.eclipse.paho.client.mqttv3.jar
org.eclipse.paho.sample.mqttv3app.Sample -b %host% -p %sslportreq% -k %cltjkskeystore% -w
%cltjkskeystorepass% -r %cltcajkstruststore% -v true
pause
endlocal
```

Obrázek 15. `cajavaclient.bat`

Související pojmy

“Zabezpečení produktu MQTT” na stránce 51

Tři koncepty jsou základní pro zabezpečení produktu MQTT : identita, ověřování a autorizace. Identita je o pojmenování klienta, který je autorizován a kterému je uděleno oprávnění. Ověřování je o prokazování identity klienta a autorizace se týká správy práv, která jsou klientovi udělena.

Související úlohy

“Generování klíčů a certifikátů” na stránce 95
Postupujte podle této procedury, chcete-li generovat klíče a certifikáty pro klienty Java a C, včetně aplikací Android a iOS , a serverů IBM WebSphere MQ a IBM MessageSight .

“Připojení ukázkové aplikace Java klienta MQTT k produktu Android přes SSL” na stránce 62
Vstupte a vyběhněte s ukázkovým klientem Android MQTT připojeným k produktu IBM WebSphere MQ přes SSL.

“Authenticating an MQTT client Java app with JAAS” na stránce 72
Naučte se, jak ověřit klienta pomocí produktu JAAS. Complete the steps in this task to modify the sample program `JAASLoginModule.java` and configure IBM WebSphere MQ to authenticate an MQTT client Java app with JAAS.

Ukázkové skripty pro konfiguraci certifikátů SSL pro produkt Windows

** **

Ukázkové příkazové soubory vytvoří certifikáty a úložiště certifikátů, jak je popsáno v krocích v úloze. Kromě toho tento příklad nastaví správce front klienta MQTT pro použití úložiště certifikátů serveru. Příklad odstraní a znovu vytvoří správce front voláním skriptu `SampleMQM.bat`, který je poskytnut s IBM WebSphere MQ.

`initcert.bat`

`initcert.bat` nastavuje názvy a cesty k certifikátům a jiným parametrům, které jsou vyžadovány příkazy **`keytool`** a **openSSL**. Nastavení jsou popsána v komentářích ve skriptu.

```
@echo off
@rem Set the path where you installed the MQTT SDK
@rem and short cuts to the samples directories.
set SDKRoot=C:\MQTT
set jsamppath=%SDKRoot%\sdk\clients\java\samples
set csamppath=%SDKRoot%\sdk\clients\c\samples
```

```
@rem Set the paths to Version 7 of the JDK
@rem and to the directory where you built the openSSL package.
@rem Set short cuts to the tools.
set javapath=C:\Program Files\IBM\Java70
set keytool="%javapath%\jre\bin\keytool.exe"
set ikeyman="%javapath%\jre\bin\ikeyman.exe"
set openssl=%SDKRoot%\openSSL
set runopenssl="%openssl%\bin\openssl"
```

```
@rem Set the path to where certificates are to be stored,
@rem and set global security parameters.
@rem Omit set password, and the security tools prompt you for passwords.
@rem Validity is the expiry time of the certificates in days.
set certpath=%SDKRoot%\Certificates
set password=password
set validity=5000
set algorithm=RSA
```

```
@rem Set the certificate authority (CA) jks keystore and certificate parameters.
@rem Omit this step, unless you are defining your own certificate authority.
@rem The CA keystore contains the key-pair for your own certificate authority.
@rem You must protect the CA keystore.
@rem The CA certificate is the self-signed certificate authority public certificate.
@rem It is commonly known as the CA root certificate.
set caalias=caalias
set cadname="CN=mqttca.ibm.id.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
set cakeypass=%password%
@rem ca key store
set cajkskeystore=%certpath%\cakeystore.jks
set cajkskeystorepass=%password%
```

```
@rem ca certificate (root certificate)
set cacert=%certpath%\cacert.cer
```

```
@rem Set the server jks keystore and certificate parameters.
@rem The server keystore contains the key-pair for the server.
@rem You must protect the server keystore.
@rem If you then export the server certificate it is self-signed.
@rem Alternatively, if you export a certificate signing request (CSR)
@rem from the server keystore for the server key,
@rem and import the signed certificate back into the same keystore,
@rem it forms a certificate chain.
@rem The certificate chain links the server certificate to the CA.
@rem When you now export the server certificate,
@rem the exported certificate includes the certificate chain.
set srvalias=srvalias
set srvdname="CN=mqttserver.ibm.id.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
set srvkeypass=%password%
@rem server key stores
set srvjkskeystore=%certpath%\srvkeystore.jks
set srvjkskeystorepass=%password%
@rem server certificates
set srvcertreq=%certpath%\srvcertreq.csr
set srvcertcasigned=%certpath%\srvcertcasigned.cer
set srvcertselfsigned=%certpath%\srvcertselfsigned.cer
```

```
@rem Set the client jks keystore and certificate parameters
@rem Omit this step, unless you are authenticating clients.
@rem The client keystore contains the key-pair for the client.
@rem You must protect the client keystore.
@rem If you then export the client certificate it is self-signed.
@rem Alternatively, if you export a certificate signing request (CSR)
@rem from the client keystore for the client key,
@rem and import the signed certificate back into the same keystore,
@rem it forms a certificate chain.
@rem The certificate chain links the client certificate to the CA.
@rem When you now export the client certificate,
@rem the exported certificate includes the certificate chain.
set cltalias=cltalias
set cltdname="CN=mqttclient.ibm.id.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
set cltkeypass=%password%
@rem client key stores
set cltjkskeystore=%certpath%\cltkeystore.jks
set cltjkskeystorepass=%password%
set cltcertreq=%certpath%\cltcertreq.csr
set cltcertcasigned=%certpath%\cltcacertsigned.cer
set cltcertselfsigned=%certpath%\cltcertselfsigned.cer
```

```
@rem Set the paths to the client truststores signed by CA and signed by server key.
@rem You only need to define one of the trust stores.
@rem A trust store holds certificates that you trust,
@rem which are used to authenticate untrusted certificates.
@rem In this example, when the client authenticates the MQTT server it connects to,
@rem it authenticates the certificate it is sent by the server
@rem with the certificates in its trust store.
@rem For example, the MQTT server sends its server certificate,
@rem and the client authenticates it with either the same server certificate
@rem that you have stored in the cltsrvtruststore.jks trust store,
@rem or against the CA certificate, if the server certificate is signed by the CA.
set cltcajkstruststore=%certpath%\cltcatruststore.jks
set cltcajkstruststorepass=%password%
set cltsrvjkstruststore=%certpath%\cltsrvtruststore.jks
set cltsrvjkstruststorepass=%password%
```

```
@rem Set the paths to the client PKCS12 and PEM key and trust stores.
@rem Omit this step, unless you are configuring a C or iOS client.
@rem You only need to define either one of the trust stores for storing CA
@rem or server signed server certificates.
set cltp12keystore=%certpath%\cltkeystore.p12
set cltp12keystorepass=%password%
set cltpemkeystore=%certpath%\cltkeystore.pem
set cltpemkeystorepass=%password%
set cltcap12truststore=%certpath%\cltcatruststore.p12
set cltcap12truststorepass=%password%
set cltcapemtruststore=%certpath%\cltcatruststore.pem
set cltcapemtruststorepass=%password%
set cltsrvp12truststore=%certpath%\cltsrvtruststore.p12
set cltsrvp12truststorepass=%password%
```

```
set cltsrvpemtruststore=%certpath%\cltsrvtruststore.pem
set cltsrvpemtruststorepass=%password%
```

```
@rem set WMQ Variables
set authopt=NEVER
set authreq=REQUIRED
set qm=MQXR_SAMPLE_QM
set host=localhost
set mcauser='Guest'
set portsslopt=8884
set chlopt=SSLOPT
set portsslreq=8885
set chlreq=SSLREQ
V7.5.0.1 set portws=1886
set chlws=PLAINWS
set chlssloptws=SSLOPTWS
set portssloptws=8886
set chlsslreqws=SSLREQWS
set portsslreqws=8887
set mqlog=%certpath%\wmq.log
```

cleancert.bat

Příkazy ve skriptu `cleancert.bat` odstraní správce front klienta MQTT, aby se zajistilo, že úložiště certifikátů serveru není uzamčeno, a potom odstraní všechna úložiště klíčů a certifikáty, které jsou vytvořeny ukázkovými skripty zabezpečení.

```
@rem Delete the MQTT sample queue manager, MQXR_SAMPLE_QM
call "%MQ_FILE_PATH%\bin\setmqenv" -s
endmqm -i %qm%
dltmqm %qm%
```

```
@rem Erase all the certificates and key stores created by the sample scripts.
erase %cajkskeystore%
erase %cacert%
erase %srvjkskeystore%
erase %srvcertreq%
erase %srvcertcasigned%
erase %srvcertselfsigned%
erase %cltjkskeystore%
erase %cltp12keystore%
erase %cltpemkeystore%
erase %cltcertreq%
erase %cltcertcasigned%
erase %cltcertselfsigned%
erase %cltcajkstruststore%
erase %cltcap12truststore%
erase %cltcapemtruststore%
erase %cltsrvjkstruststore%
erase %cltsrvp12truststore%
erase %cltsrvpemtruststore%
erase %mqlog%
@echo Cleared all certificates
dir %certpath%\*.* /b
```

genkeys.bat

Příkazy ve skriptu `genkeys.bat` vytvářejí dvojice klíčů pro soukromé certifikační autority, server a klienta.

```
@rem
@echo ____________________________________________________________________________
@echo Generate %caalias%, %srvalias%, and %cltalias% key-pairs in %cajkskeystore%,
%srvjkskeystore%, and %cltjkskeystore%
@rem
@rem -- Generate a client certificate and a private key pair
@rem Omit this step, unless you are authenticating clients.
%keytool% -genkeypair -noprompt -alias %cltalias% -dname %cltdname% -keystore
%cltjkskeystore% -storepass %cltjkskeystorepass% -keypass %cltkeypass% -keyalg %algorithm%
-validity %validity%
```

```
@rem -- Generate a server certificate and private key pair
%keytool% -genkeypair -noprompt -alias %srvalias% -dname %srvdname% -keystore
%srvjkskeystore% -storepass %srvjkskeystorepass% -keypass %srvkeypass% -keyalg %algorithm%
-validity %validity%
```

```
@rem Create CA, client and server key-pairs
@rem  -- Generate a CA certificate and private key pair - The extension asserts this is a
certificate authority certificate, which is required to import into firefox
%keytool% -genkeypair -noprompt -ext bc=ca:true -alias %caalias%  -dname %cadname%
-keystore %cajkskeystore% -storepass %cajkskeystorepass%  -keypass %cakeypass% -keyalg
%algorithm% -validity %validity%
```

sscerts.bat

Příkazy ve skriptu `sscerts.bat` exportují certifikáty podepsané držitelem klienta a serveru z vlastních úložišť klíčů a importují certifikát serveru do úložiště údajů o důvěryhodnosti klienta a certifikát klienta do úložiště klíčů serveru. Server nemá úložiště údajů o důvěryhodnosti. Příkazy vytvoří úložiště údajů o důvěryhodnosti klienta ve formátu PEM z úložiště údajů o důvěryhodnosti JKS klienta.

```
@rem
@echo ______________________________________________________________________
@echo Export self-signed certificates:  %srvcertselfsigned% and %cltcertselfsigned%
@rem Export Server public certificate
%keytool% -exportcert -noprompt -rfc -alias %srvalias% -keystore %srvjkskeystore%
-storepass %srvjkskeystorepass% -file %srvcertselfsigned%
@rem Export Client public certificate
@rem Omit this step, unless you are authenticating clients.
%keytool% -exportcert -noprompt -rfc -alias %cltalias% -keystore %cltjkskeystore%
-storepass %cltjkskeystorepass% -file %cltcertselfsigned%
```

```
@rem
@echo ______________________________________________________________________
@echo Add selfsigned server certificate %srvcertselfsigned% to client trust store:
%cltsrvjkstruststore%
@rem Import the server certificate into the client-server trust store (for server self-
signed authentication)
%keytool% -import -noprompt -alias %srvalias% -file %srvcertselfsigned% -keystore
%cltsrvjkstruststore% -storepass %cltsrvjkstruststorepass%
```

```
@rem
@echo ______________________________________________________________________
@echo Add selfsigned client certificate %cltcertselfsigned% to server trust store:
%srvjkskeystore%
@rem Import the client certificate into the server trust store (for client self-signed
authentication)
@rem Omit this step, unless you are authenticating clients.
%keytool% -import -noprompt -alias %cltalias% -file %cltcertselfsigned% -keystore
%srvjkskeystore% -storepass %srvjkskeystorepass%
```

```
@rem
@echo ______________________________________________________________________
@echo Create a pem client-server trust store from the jks client-server trust store:
%cltsrvpemtruststore%
@rem
%keytool% -importkeystore -noprompt -srckeystore %cltsrvjkstruststore% -destkeystore
%cltsrvp12truststore% -srcstoretype jks -deststoretype pkcs12 -srcstorepass
%cltsrvjkstruststorepass% -deststorepass %cltsrvp12truststorepass%
%openssl%\bin\openssl pkcs12 -in %cltsrvp12truststore% -out %cltsrvpemtruststore% -passin
pass:%cltsrvp12truststorepass% -passout pass:%cltsrvpemtruststorepass%@rem
@rem
@echo ______________________________________________________________________
@echo Create a pem client key store from the jks client keystore
@rem Omit this step, unless you are configuring a C or iOS client.
@rem
%keytool% -importkeystore -noprompt -srckeystore %cltjkskeystore%      -destkeystore
%cltp12keystore%     -srcstoretype jks -deststoretype pkcs12 -srcstorepass
%cltjkskeystorepass%     -deststorepass %cltp12keystorepass%
%openssl%\bin\openssl pkcs12 -in %cltp12keystore%     -out %cltpemkeystore%      -passin
pass:%cltp12keystorepass% -passout pass:%cltpemkeystorepass%
```

cacerts.bat

Skript importuje kořenový certifikát certifikační autority soukromých úložišť klíčů. Kořenový certifikát CA je potřebný k vytvoření svazku klíčů mezi kořenovým certifikátem a podepsaným certifikátem. Skript `cacerts.bat` exportuje požadavky na certifikáty klienta a serveru ze svých úložišť klíčů. Skript podepíše požadavky na certifikáty s klíčem soukromé certifikační autority v úložišti klíčů

cajkskeystore.jks a poté naimportuje podepsané certifikáty zpět do stejných úložišť klíčů, ze kterých přišly požadavky. Import vytvoří řetěz certifikátů s kořenovým certifikátem CA. Skript vytvoří úložiště údajů o důvěryhodnosti klienta ve formátu PEM z úložiště údajů o důvěryhodnosti JKS klienta.

```
@rem
@echo ------------------------------------------------------------------------------
@echo Export self-signed certificates: %cacert%
@rem
@rem Export CA public certificate
%keytool% -exportcert -noprompt -rfc -alias %caalias%  -keystore %cajkskeystore% -storepass
%cajkskeystorepass%  -file %cacert%
```

```
@rem
@echo ------------------------------------------------------------------------------
@echo Add CA to server key and client key and trust stores: %srvjkskeystore%,
%cltjkskeystore%, %cltcajkstruststore%,
@rem The CA certificate is necessary to create key chains in the client and server key
stores,
@rem and to certify key chains in the server key store and the client trust store
@rem
@rem Import the CA root certificate into the server key store
%keytool% -import -noprompt -alias %caalias% -file %cacert% -keystore %srvjkskeystore%
-storepass %srvjkskeystorepass%
@rem Import the CA root certificate into the client key store
@rem Omit this step, unless you are authenticating clients.
%keytool% -import -noprompt -alias %caalias% -file %cacert% -keystore %cltjkskeystore%
-storepass %cltjkskeystorepass%
@rem Import the CA root certificate into the client ca-trust store (for ca chained
authentication)
%keytool% -import -noprompt -alias %caalias% -file %cacert% -keystore %cltcajkstruststore%
-storepass %cltcajkstruststorepass%
```

```
@rem
@echo ------------------------------------------------------------------------------
@echo Create certificate signing requests: %srvcertreq% and %cltcertreq%
@rem
@rem Create a certificate signing request (CSR) for the server key
%keytool% -certreq -alias %srvalias% -file %srvcertreq% -keypass %srvkeypass% -keystore
%srvjkskeystore% -storepass %srvjkskeystorepass%
@rem Create a certificate signing request (CSR) for the client key
%keytool% -certreq -alias %cltalias% -file %cltcertreq% -keypass %cltkeypass% -keystore
%cltjkskeystore% -storepass %cltjkskeystorepass%
```

```
@rem
@echo ------------------------------------------------------------------------------
@echo Sign certificate requests: %srvcertcasigned% and %cltcertcasigned%
@rem The requests are signed with the ca key in the cajkskeystore.jks keystore
@rem
@rem Sign server certificate request
%keytool% -gencert -infile %srvcertreq% -outfile %srvcertcasigned% -alias %caalias%
-keystore %cajkskeystore% -storepass %cajkskeystorepass% -keypass %cakeypass%
@rem Sign client certificate request
@rem Omit this step, unless you are authenticating clients.
%keytool% -gencert -infile %cltcertreq% -outfile %cltcertcasigned% -alias %caalias%
-keystore %cajkskeystore% -storepass %cajkskeystorepass% -keypass %cakeypass%
```

```
@rem
@echo ------------------------------------------------------------------------------
@echo Import the signed certificates back into the key stores to create the key chain:
%srvjkskeystore% and %cltjkskeystore%
@rem
@rem Import the signed server certificate
%keytool% -import -noprompt -alias %srvalias% -file %srvcertcasigned% -keypass %srvkeypass%
-keystore %srvjkskeystore% -storepass %srvjkskeystorepass%
@rem Import the signed client certificate and key chain back into the client keystore
%keytool% -import -noprompt -alias %cltalias% -file %cltcertcasigned% -keypass %cltkeypass%
-keystore %cltjkskeystore% -storepass %cltjkskeystorepass%
@rem
@rem The CA certificate is needed in the server key store, and the client trust store
@rem to verify the key chain sent from the client or server
@echo Delete the CA certificate from %cltjkskeystore%: it causes a problem in converting
keystore to pem
@rem Omit this step, unless you are authenticating clients.
%keytool% -delete -alias %caalias% -keystore %cltjkskeystore% -storepass
%cltjkskeystorepass%
```

```
@rem
@echo ------------------------------------------------------------------------------
@echo Create a pem client-ca trust store from the jks client-ca trust store:
%cltcapemtruststore%
@rem Omit this step, unless you are configuring a C or iOS client.
@rem
%keytool% -importkeystore -noprompt -srckeystore %cltcajkstruststore%  -destkeystore
%cltcap12truststore%  -srcstoretype jks -deststoretype pkcs12 -srcstorepass
%cltcajkstruststorepass%  -deststorepass %cltcap12truststorepass%
%openssl%\bin\openssl pkcs12 -in %cltcap12truststore%  -out %cltcapemtruststore%  -passin
pass:%cltcap12truststorepass%  -passout pass:%cltpemtruststorepass%
```

```
@rem
@echo ------------------------------------------------------------------------------
@echo Create a pem client key store from the jks client keystore
@rem Omit this step, unless you are configuring a C or iOS client.
@rem
%keytool% -importkeystore -noprompt -srckeystore %cltjkskeystore%      -destkeystore
%cltp12keystore%     -srcstoretype jks -deststoretype pkcs12 -srcstorepass
%cltjkskeystorepass%     -deststorepass %cltp12keystorepass%
%openssl%\bin\openssl pkcs12 -in %cltp12keystore%     -out %cltpemkeystore%      -passin
pass:%cltp12keystorepass% -passout pass:%cltpemkeystorepass%
```

mqcerts.bat

Skript uvádí úložiště klíčů a certifikáty v adresáři certifikátů. Poté vytvoří ukázkového správce front MQTT a nakonfiguruje zabezpečené kanály telemetrie.

```
@echo ------------------------------------------------------------------------------
@echo List keystores and certificates
dir %certpath%\*.* /b
```

```
@rem
@echo Create queue manager and define mqtt channels and certificate stores
call "%MQ_FILE_PATH%\mqxr\Samples\SampleMQM" >> %mqlog%
echo DEFINE CHANNEL(%chlreq%)      CHLTYPE(MQTT) TRPTYPE(TCP) PORT(%portsslreq%)
SSLCAUTH(%authreq%) SSLKEYR('%srvjkskeystore%') SSLKEYP('%srvjkskeystorepass%')
MCAUSER(%mcauser%) | runmqsc %qm% >> %mqlog%
echo DEFINE CHANNEL(%chlopt%)      CHLTYPE(MQTT) TRPTYPE(TCP) PORT(%portsslopt%)
SSLCAUTH(%authopt%) SSLKEYR('%srvjkskeystore%') SSLKEYP('%srvjkskeystorepass%')
MCAUSER(%mcauser%) | runmqsc %qm% >> %mqlog%
V 7.5.0.1
echo DEFINE CHANNEL(%chlsslreqws%) CHLTYPE(MQTT) TRPTYPE(TCP) PORT(%portsslreqws%)
SSLCAUTH(%authreq%) SSLKEYR('%srvjkskeystore%') SSLKEYP('%srvjkskeystorepass%')
MCAUSER(%mcauser%) PROTOCOL(HTTP) | runmqsc %qm% >> %mqlog%
echo DEFINE CHANNEL(%chlssloptws%) CHLTYPE(MQTT) TRPTYPE(TCP) PORT(%portssloptws%)
SSLCAUTH(%authopt%) SSLKEYR('%srvjkskeystore%') SSLKEYP('%srvjkskeystorepass%')
MCAUSER(%mcauser%) PROTOCOL(HTTP) | runmqsc %qm% >> %mqlog%
echo DEFINE CHANNEL(%chlws%)       CHLTYPE(MQTT) TRPTYPE(TCP) PORT(%portws%)
MCAUSER(%mcauser%) PROTOCOL(HTTP) | runmqsc %qm% >> %mqlog%
@echo MQ logs saved in %mqlog%echo
```

Připojení ukázkové aplikace Java klienta MQTT k produktu Android přes SSL

Vstupte a vyběhněte s ukázkovým klientem Android MQTT připojeným k produktu IBM WebSphere MQ přes SSL.

Než začnete

Tento článek předpokládá, že jste spustili alespoň úroveň rozhraní API systému Android 14 (ICS 4.0). Dřívější úrovně měly úložiště klíčů, ale k němu mají přístup pouze systémové aplikace.

1. Musíte mít přístup k serveru MQTT version 3.1 , který podporuje protokol MQTT protocol přes SSL.
2. Je-li mezi vaším klientem a serverem brána firewall, zkontrolujte, zda neblokuje provoz produktu MQTT .
3. Pokud testujete připojení na začátku zařízení Android , budete možná potřebovat kartu SD pro přenos certifikátu do zařízení.
4. Pokud testujete připojení na virtuálním zařízení Android , nakonfigurujte SD kartu pro virtuální zařízení.
5. Kanály SSL musí být spuštěny.

Informace o této úloze

Dokončete tuto úlohu ke spuštění produktu Ukázková aplikace klienta MQTT Java pro produkt Android přes SSL. Úspěšné připojení SSL vytváří zabezpečený šifrovaný kanál mezi vaším zařízením Android a serverem MQTT . Identita serveru je ověřena.

Pomocí Androidmůžete ověřit server pomocí SSL. Zařízení můžete také ověřit, ačkoli ukázková aplikace toto zařízení nepodporuje. Chcete-li ověřit zařízení, buď použijte rozhraní KeyChain API, nebo použijte JAAS k ověření identifikátoru klienta, IP adresy klienta nebo jména uživatele a hesla poskytnutého aplikací MQTT Android .

Libovolné certifikáty X.509 , které nainstalujete do úložiště údajů o důvěryhodnosti serveru Android , musí být podepsány certifikační autoritou. V tomto příkladu vytvoříte certifikační autoritu, která podepisuje certifikát, který jste nainstalovali ve svém zařízení Android . Několik kořenových certifikátů je předinstalováno do zařízení Android .

Než budete instalovat důvěryhodný certifikát, musíte před instalací důvěryhodného certifikátu vytvořit zámek na svém zařízení Android . Zámek zabraňuje tomu, aby někdo instaloval certifikáty na zařízení bez vašich znalostí.

Postup

1. Zvolte server MQTT , ke kterému se můžete připojit k aplikaci klienta.

   Server musí podporovat protokol MQTT version 3.1 přes SSL. Všechny servery MQTT z IBM to dělají, včetně IBM WebSphere MQ a IBM MessageSight. Viz “Začínáme se servery MQTT” na stránce 135.
2. Spusťte ukázkovou aplikaci klienta MQTT for Android "`MQTTExerciser`" na nezabezpečeném kanálu MQTT . Viz “Začínáme s produktem Klient MQTT pro produkt Java on Android” na stránce 17.

   Použijte aplikaci znovu k testování zabezpečeného kanálu.

   Pokud jste spustili virtuální zařízení Android , nechte jej běžet.
3. Volitelné: Nainstalujte vývojovou sadu produktu Java (JDK) ve verzi 7 nebo novější.

   verze 7 je vyžadován ke spuštění příkazu **keytool** pro certifikaci certifikátů. Pokud certifikáty certifikovat nechcete, není třeba mít sadu JDK verze 7.
4. Vytvořte a spusťte skripty pro generování dvojic klíčů a certifikátů a nakonfigurujte produkt IBM WebSphere MQ jako server MQTT .

   Postupujte podle kroků v části “Generování klíčů a certifikátů” na stránce 95 a vytvořte a spusťte skripty. Tyto skripty jsou také vypsány v “Ukázkové skripty pro konfiguraci certifikátů SSL pro produkt Windows” na stránce 66.

   Potřebujete veřejný certifikát vydavatele certifikátů a úložiště klíčů serveru. Nepotřebujete certifikáty klienta ani žádné certifikáty ve formátu `.pem` nebo `.p12` .
5. Zkontrolujte, zda jsou kanály zabezpečení SSL spuštěny a nastaveny dle očekávání.

   V IBM WebSphere MQ zadejte následující příkaz do okna příkazového řádku:

   - Linux

     ```
     echo 'DISPLAY CHSTATUS(SSL*) CHLTYPE(MQTT) ALL' | runmqsc MQXR_SAMPLE_QM
     echo 'DISPLAY CHANNEL(SSL*) CHLTYPE(MQTT) ALL' | runmqsc MQXR_SAMPLE_QM
     ```

   - Windows

     ```
     echo DISPLAY CHSTATUS(SSL*) CHLTYPE(MQTT) ALL | runmqsc MQXR_SAMPLE_QM
     echo DISPLAY CHANNEL(SSL*) CHLTYPE(MQTT) ALL | runmqsc MQXR_SAMPLE_QM
     ```
6. Nainstalujte certifikát certifikační autority do úložiště údajů o důvěryhodnosti produktu Android .

   Soubor certifikační autority v příkladu je `cacert.cer`.

   a) Přejmenujte certifikát na `cacert.crt` .

b) Zkopírujte certifikát do kořenového interního úložiště nebo na kartu SD.

Pro spuštěné virtuální zařízení otevřete produkt Eclipse nebo spusťte příkaz Android Debug Bridge (ADB) a zkopírujte certifikát na virtuální zařízení:

Eclipse

i) Spusťte produkt Eclipsea otevřete perspektivu DDMS.

ii) V hlavním pohledu otevřete okno **Průzkumník souborů** .

iii) Otevřete adresář `mnt/sdcard` .

iv) Přetáhněte soubor `cacert.crt` do adresáře `mnt/sdcard` .

ADB

i) Otevřete příkazové okno a nastavte jeho aktuální adresář na `android-sdk\platform-tools` v instalačním adresáři android, např. `C:\Program Files\Android\android-sdk\platform-tools`.

ii) Zkopírujte certifikát do adresáře `mnt/sdcard` :

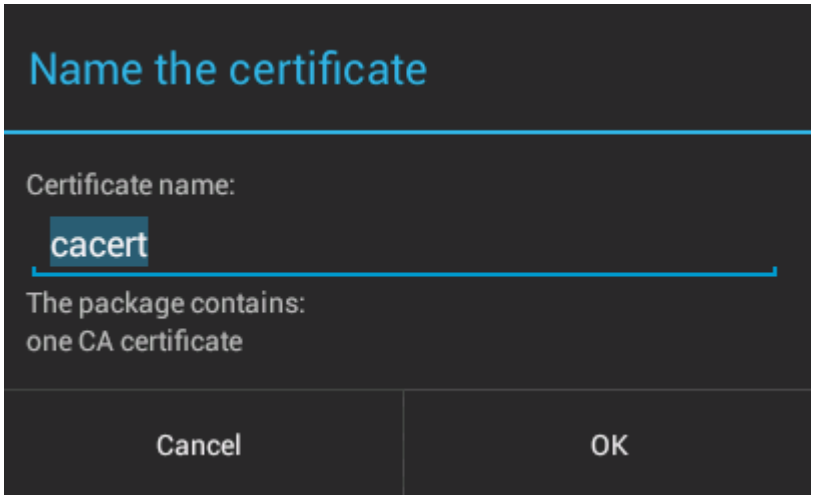
```
adb push %cacert% /mnt/sdcard/cacert.crt
```

7. Nainstalujte certifikát v úložišti údajů o důvěryhodnosti certifikátů na zařízeníAndroid .

Certifikát musí mít klauzuli Basic Constraints s hodnotou `Subject Type=CA`.

a) Odemkněte zařízení a klepněte na tlačítko **widgety** .

b) Klepněte na nabídku **Nastavení** > **Zabezpečení** > **Úložiště pověření** > **Instalovat z karty SD**.

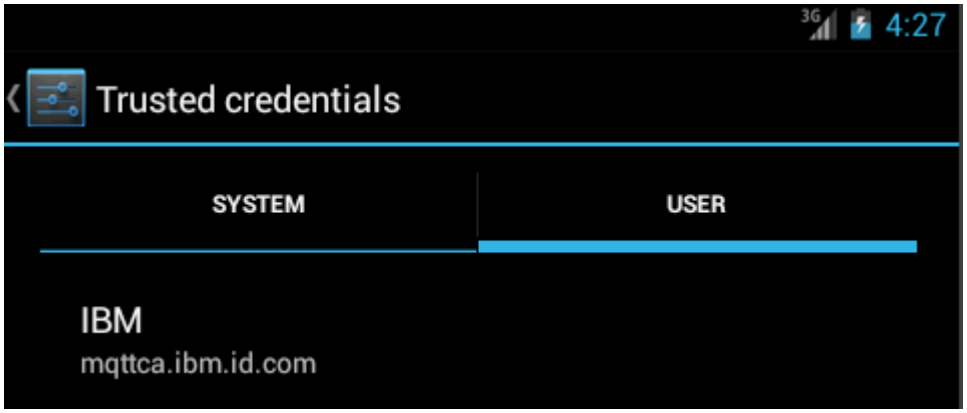


c) Potvrďte, že je název souboru certifikátu správný, a klepněte na tlačítko **OK**.

Poznámka: Pokud jste pro zařízení nedefinovali zámek, jste nyní vyzváni Android k nastavení uzamčeného zámku.

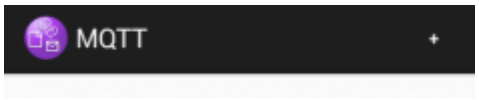
8. Potvrďte, že je certifikát instalován na zařízení.

a) Klepněte na volbu **Důvěryhodná pověření** > **Uživatel** a počkejte několik minut, nebo tak, aby se váš certifikát zobrazil v seznamu uživatelských certifikátů.



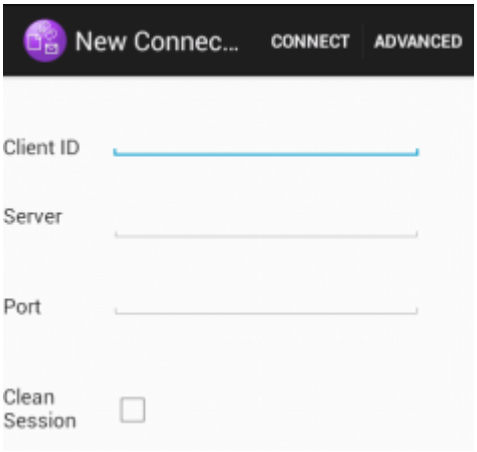
9. Znovu spusťte aplikaci `MQTTExerciser` a připojte se k kanálu produktu MQTT , který jste nakonfigurovali pro anonymní klienty SSL.
   a) Otevřete produkt Ukázková aplikace klienta MQTT Java pro produkt Android.

      Toto okno je otevřeno ve vašem zařízení Android :

   b) Připojte se k serveru MQTT .

      i) Klepnutím na symbol **+** otevřete nové připojení MQTT .

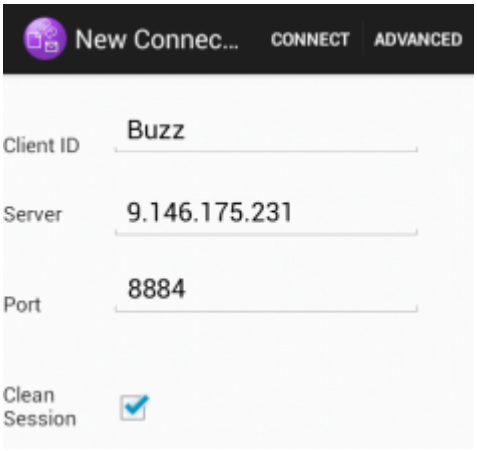
      ii) Do pole **ID klienta** zadejte libovolný jedinečný identifikátor. Buďte trpěliví, úhozy mohou být pomalé.

      iii) Zadejte do pole **Server** adresu IP vašeho serveru MQTT .

      Jedná se o server, který jste vybrali v prvním hlavním kroku. Adresa IP nesmí být `127.0.0.1`

      iv) Zadejte číslo portu pro připojení MQTT .

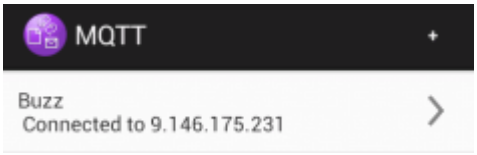
      Nastavte číslo portu na hodnotu 8884, které je nastaveno proměnnou `%sslportopt%` v ukázkových skriptech. Toto číslo portu je číslo portu kanálu produktu MQTT , který jste nakonfigurovali pro anonymní klienty SSL spuštěním ukázkových skriptů v kroku “4” na stránce 63.



v) Klepněte na kartu **Rozšířené** a vyberte volbu **SSL** . Klepněte na tlačítko **Uložit**.

vi) Klepněte na tlačítko **Připojit**.

Je-li připojení úspěšné, zobrazí se zpráva "Connecting" následované tímto oknem:



Výsledky

Aplikace `MQTTExerciser` trvá o něco déle, než se připojí a vymění zprávy, ale jinak se chová jinak, než se připojuje k nezabezpečenému připojení.

Související pojmy

“Zabezpečení produktu MQTT” na stránce 51
Tři koncepty jsou základní pro zabezpečení produktu MQTT : identita, ověřování a autorizace. Identita je o pojmenování klienta, který je autorizován a kterému je uděleno oprávnění. Ověřování je o prokazování identity klienta a autorizace se týká správy práv, která jsou klientovi udělena.

Související úlohy

“Generování klíčů a certifikátů” na stránce 95
Postupujte podle této procedury, chcete-li generovat klíče a certifikáty pro klienty Java a C, včetně aplikací Android a iOS , a serverů IBM WebSphere MQ a IBM MessageSight .

“Sestavení a spuštění zabezpečeného produktu Ukázková aplikace klienta MQTT Java” na stránce 54
Na základě příkladu Windows můžete začít pracovat se zabezpečenou ukázkovou aplikací Java na serveru buď IBM MessageSight , nebo IBM WebSphere MQ jako server MQTT. Aplikaci Klient MQTT pro produkt Java můžete spustit na libovolné platformě s JSE 1.5 nebo vyšším, které je "Java Kompatibilní"

“Authenticating an MQTT client Java app with JAAS” na stránce 72
Naučte se, jak ověřit klienta pomocí produktu JAAS. Complete the steps in this task to modify the sample program `JAASLoginModule.java` and configure IBM WebSphere MQ to authenticate an MQTT client Java app with JAAS.

Ukázkové skripty pro konfiguraci certifikátů SSL pro produkt Windows

** **

Ukázkové příkazové soubory vytvoří certifikáty a úložiště certifikátů, jak je popsáno v krocích v úloze. Kromě toho tento příklad nastaví správce front klienta MQTT pro použití úložiště certifikátů serveru. Příklad odstraní a znovu vytvoří správce front voláním skriptu `SampleMQM.bat`, který je poskytnut s IBM WebSphere MQ.

initcert.bat

initcert.bat nastavuje názvy a cesty k certifikátům a jiným parametrům, které jsou vyžadovány příkazy **keytool** a **openSSL**. Nastavení jsou popsána v komentářích ve skriptu.

```
@echo off
@rem Set the path where you installed the MQTT SDK
@rem and short cuts to the samples directories.
set SDKRoot=C:\MQTT
set jsamppath=%SDKRoot%\sdk\clients\java\samples
set csamppath=%SDKRoot%\sdk\clients\c\samples
```

```
@rem Set the paths to Version 7 of the JDK
@rem and to the directory where you built the openSSL package.
@rem Set short cuts to the tools.
set javapath=C:\Program Files\IBM\Java70
set keytool="%javapath%\jre\bin\keytool.exe"
set ikeyman="%javapath%\jre\bin\ikeyman.exe"
set openssl=%SDKRoot%\openSSL
set runopenssl="%openssl%\bin\openssl"
```

```
@rem Set the path to where certificates are to be stored,
@rem and set global security parameters.
@rem Omit set password, and the security tools prompt you for passwords.
@rem Validity is the expiry time of the certificates in days.
set certpath=%SDKRoot%\Certificates
set password=password
set validity=5000
set algorithm=RSA
```

```
@rem Set the certificate authority (CA) jks keystore and certificate parameters.
@rem Omit this step, unless you are defining your own certificate authority.
@rem The CA keystore contains the key-pair for your own certificate authority.
@rem You must protect the CA keystore.
@rem The CA certificate is the self-signed certificate authority public certificate.
@rem It is commonly known as the CA root certificate.
set caalias=caalias
set cadname="CN=mqttca.ibm.id.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
set cakeypass=%password%
@rem ca key store
set cajkskeystore=%certpath%\cakeystore.jks
set cajkskeystorepass=%password%
@rem ca certificate (root certificate)
set cacert=%certpath%\cacert.cer
```

```
@rem Set the server jks keystore and certificate parameters.
@rem The server keystore contains the key-pair for the server.
@rem You must protect the server keystore.
@rem If you then export the server certificate it is self-signed.
@rem Alternatively, if you export a certificate signing request (CSR)
@rem from the server keystore for the server key,
@rem and import the signed certificate back into the same keystore,
@rem it forms a certificate chain.
@rem The certificate chain links the server certificate to the CA.
@rem When you now export the server certificate,
@rem the exported certificate includes the certificate chain.
set srvalias=srvalias
set srvdname="CN=mqttserver.ibm.id.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
set srvkeypass=%password%
@rem server key stores
set srvjkskeystore=%certpath%\srvkeystore.jks
set srvjkskeystorepass=%password%
@rem server certificates
set srvcertreq=%certpath%\srvcertreq.csr
set srvcertcasigned=%certpath%\srvcertcasigned.cer
set srvcertselfsigned=%certpath%\srvcertselfsigned.cer
```

```
@rem Set the client jks keystore and certificate parameters
@rem Omit this step, unless you are authenticating clients.
@rem The client keystore contains the key-pair for the client.
@rem You must protect the client keystore.
@rem If you then export the client certificate it is self-signed.
@rem Alternatively, if you export a certificate signing request (CSR)
@rem from the client keystore for the client key,
@rem and import the signed certificate back into the same keystore,
@rem it forms a certificate chain.
```

```
@rem The certificate chain links the client certificate to the CA.
@rem When you now export the client certificate,
@rem the exported certificate includes the certificate chain.
set cltalias=cltalias
set cltdname="CN=mqttclient.ibm.id.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
set cltkeypass=%password%
@rem client key stores
set cltjkskeystore=%certpath%\cltkeystore.jks
set cltjkskeystorepass=%password%
set cltcertreq=%certpath%\cltcertreq.csr
set cltcertcasigned=%certpath%\cltcacertsigned.cer
set cltcertselfsigned=%certpath%\cltcertselfsigned.cer
```

```
@rem Set the paths to the client truststores signed by CA and signed by server key.
@rem You only need to define one of the trust stores.
@rem A trust store holds certificates that you trust,
@rem which are used to authenticate untrusted certificates.
@rem In this example, when the client authenticates the MQTT server it connects to,
@rem it authenticates the certificate it is sent by the server
@rem with the certificates in its trust store.
@rem For example, the MQTT server sends its server certificate,
@rem and the client authenticates it with either the same server certificate
@rem that you have stored in the cltsrvtruststore.jks trust store,
@rem or against the CA certificate, if the server certificate is signed by the CA.
set cltcajkstruststore=%certpath%\cltcatruststore.jks
set cltcajkstruststorepass=%password%
set cltsrvjkstruststore=%certpath%\cltsrvtruststore.jks
set cltsrvjkstruststorepass=%password%
```

```
@rem Set the paths to the client PKCS12 and PEM key and trust stores.
@rem Omit this step, unless you are configuring a C or iOS client.
@rem You only need to define either one of the trust stores for storing CA
@rem or server signed server certificates.
set cltp12keystore=%certpath%\cltkeystore.p12
set cltp12keystorepass=%password%
set cltpemkeystore=%certpath%\cltkeystore.pem
set cltpemkeystorepass=%password%
set cltcap12truststore=%certpath%\cltcatruststore.p12
set cltcap12truststorepass=%password%
set cltcapemtruststore=%certpath%\cltcatruststore.pem
set cltcapemtruststorepass=%password%
set cltsrvp12truststore=%certpath%\cltsrvtruststore.p12
set cltsrvp12truststorepass=%password%
set cltsrvpemtruststore=%certpath%\cltsrvtruststore.pem
set cltsrvpemtruststorepass=%password%
```

```
@rem set WMQ Variables
set authopt=NEVER
set authreq=REQUIRED
set qm=MQXR_SAMPLE_QM
set host=localhost
set mcauser='Guest'
set portsslopt=8884
set chlopt=SSLOPT
set portsslreq=8885
set chlreq=SSLREQ
V7.5.0.1 set portws=1886
set chlws=PLAINWS
set chlssloptws=SSLOPTWS
set portssloptws=8886
set chlsslreqws=SSLREQWS
set portsslreqws=8887
set mqlog=%certpath%\wmq.log
```

cleancert.bat

Příkazy ve skriptu `cleancert.bat` odstraní správce front klienta MQTT, aby se zajistilo, že úložiště certifikátů serveru není uzamčeno, a potom odstraní všechna úložiště klíčů a certifikáty, které jsou vytvořeny ukázkovými skripty zabezpečení.

```
@rem Delete the MQTT sample queue manager, MQXR_SAMPLE_QM
call "%MQ_FILE_PATH%\bin\setmqenv" -s
endmqm -i %qm%
dltmqm %qm%
```

```
@rem Erase all the certificates and key stores created by the sample scripts.
erase %cajkskeystore%
erase %cacert%
erase %srvjkskeystore%
erase %srvcertreq%
erase %srvcertcasigned%
erase %srvcertselfsigned%
erase %cltjkskeystore%
erase %cltp12keystore%
erase %cltpemkeystore%
erase %cltcertreq%
erase %cltcertcasigned%
erase %cltcertselfsigned%
erase %cltcajkstruststore%
erase %cltcap12truststore%
erase %cltcapemtruststore%
erase %cltsrvjkstruststore%
erase %cltsrvp12truststore%
erase %cltsrvpemtruststore%
erase %mqlog%
@echo Cleared all certificates
dir %certpath%\*.* /b
```

genkeys.bat

Příkazy ve skriptu `genkeys.bat` vytvářejí dvojice klíčů pro soukromé certifikační autority, server a klienta.

```
@rem
@echo ____________________________________________________________________________
@echo Generate %caalias%, %srvalias%, and %cltalias% key-pairs in %cajkskeystore%,
%srvjkskeystore%, and %cltjkskeystore%
@rem
@rem -- Generate a client certificate and a private key pair
@rem Omit this step, unless you are authenticating clients.
%keytool% -genkeypair -noprompt -alias %cltalias% -dname %cltdname% -keystore
%cltjkskeystore% -storepass %cltjkskeystorepass% -keypass %cltkeypass% -keyalg %algorithm%
-validity %validity%
```

```
@rem -- Generate a server certificate and private key pair
%keytool% -genkeypair -noprompt -alias %srvalias% -dname %srvdname% -keystore
%srvjkskeystore% -storepass %srvjkskeystorepass% -keypass %srvkeypass% -keyalg %algorithm%
-validity %validity%
```

```
@rem Create CA, client and server key-pairs
@rem  -- Generate a CA certificate and private key pair - The extension asserts this is a
certificate authority certificate, which is required to import into firefox
%keytool% -genkeypair -noprompt -ext bc=ca:true -alias %caalias%  -dname %cadname%
-keystore %cajkskeystore% -storepass %cajkskeystorepass%  -keypass %cakeypass% -keyalg
%algorithm% -validity %validity%
```

sscerts.bat

Příkazy ve skriptu `sscerts.bat` exportují certifikáty podepsané držitelem klienta a serveru z vlastních úložišť klíčů a importují certifikát serveru do úložiště údajů o důvěryhodnosti klienta a certifikát klienta do úložiště klíčů serveru. Server nemá úložiště údajů o důvěryhodnosti. Příkazy vytvoří úložiště údajů o důvěryhodnosti klienta ve formátu PEM z úložiště údajů o důvěryhodnosti JKS klienta.

```
@rem
@echo ____________________________________________________________________________
@echo Export self-signed certificates:  %srvcertselfsigned% and %cltcertselfsigned%
@rem Export Server public certificate
%keytool% -exportcert -noprompt -rfc -alias %srvalias% -keystore %srvjkskeystore%
-storepass %srvjkskeystorepass% -file %srvcertselfsigned%
@rem Export Client public certificate
@rem Omit this step, unless you are authenticating clients.
%keytool% -exportcert -noprompt -rfc -alias %cltalias% -keystore %cltjkskeystore%
-storepass %cltjkskeystorepass% -file %cltcertselfsigned%
```

```
@rem
@echo ____________________________________________________________________________
@echo Add selfsigned server certificate %srvcertselfsigned% to client trust store:
%cltsrvjkstruststore%
@rem Import the server certificate into the client-server trust store (for server self-
```

```
signed authentication)
%keytool% -import -noprompt -alias %srvalias% -file %srvcertselfsigned% -keystore
%cltsrvjkstruststore% -storepass %cltsrvjkstruststorepass%
```

```
@rem
@echo ------------------------------------------------------------------------------
@echo Add selfsigned client certificate %cltcertselfsigned% to server trust store:
%srvjkskeystore%
@rem Import the client certificate into the server trust store (for client self-signed
authentication)
@rem Omit this step, unless you are authenticating clients.
%keytool% -import -noprompt -alias %cltalias% -file %cltcertselfsigned% -keystore
%srvjkskeystore% -storepass %srvjkskeystorepass%
```

```
@rem
@echo ------------------------------------------------------------------------------
@echo Create a pem client-server trust store from the jks client-server trust store:
%cltsrvpemtruststore%
@rem
%keytool% -importkeystore -noprompt -srckeystore %cltsrvjkstruststore% -destkeystore
%cltsrvp12truststore% -srcstoretype jks -deststoretype pkcs12 -srcstorepass
%cltsrvjkstruststorepass% -deststorepass %cltsrvp12truststorepass%
%openssl%\bin\openssl pkcs12 -in %cltsrvp12truststore% -out %cltsrvpemtruststore% -passin
pass:%cltsrvp12truststorepass% -passout pass:%cltsrvpemtruststorepass%@rem
@rem
@echo ------------------------------------------------------------------------------
@echo Create a pem client key store from the jks client keystore
@rem Omit this step, unless you are configuring a C or iOS client.
@rem
%keytool% -importkeystore -noprompt -srckeystore %cltjkskeystore%      -destkeystore
%cltp12keystore%     -srcstoretype jks -deststoretype pkcs12 -srcstorepass
%cltjkskeystorepass%     -deststorepass %cltp12keystorepass%
%openssl%\bin\openssl pkcs12 -in %cltp12keystore%     -out %cltpemkeystore%      -passin
pass:%cltp12keystorepass% -passout pass:%cltpemkeystorepass%
```

cacerts.bat

Skript importuje kořenový certifikát certifikační autority soukromých úložišť klíčů. Kořenový certifikát CA je potřebný k vytvoření svazku klíčů mezi kořenovým certifikátem a podepsaným certifikátem. Skript `cacerts.bat` exportuje požadavky na certifikáty klienta a serveru ze svých úložišť klíčů. Skript podepíše požadavky na certifikáty s klíčem soukromé certifikační autority v úložišti klíčů `cajkskeystore.jks` a poté naimportuje podepsané certifikáty zpět do stejných úložišť klíčů, ze kterých přišly požadavky. Import vytvoří řetěz certifikátů s kořenovým certifikátem CA. Skript vytvoří úložiště údajů o důvěryhodnosti klienta ve formátu PEM z úložiště údajů o důvěryhodnosti JKS klienta.

```
@rem
@echo ------------------------------------------------------------------------------
@echo Export self-signed certificates: %cacert%
@rem
@rem Export CA public certificate
%keytool% -exportcert -noprompt -rfc -alias %caalias%  -keystore %cajkskeystore% -storepass
%cajkskeystorepass%  -file %cacert%
```

```
@rem
@echo ------------------------------------------------------------------------------
@echo Add CA to server key and client key and trust stores: %srvjkskeystore%,
%cltjkskeystore%, %cltcajkstruststore%,
@rem The CA certificate is necessary to create key chains in the client and server key
stores,
@rem and to certify key chains in the server key store and the client trust store
@rem
@rem Import the CA root certificate into the server key store
%keytool% -import -noprompt -alias %caalias% -file %cacert% -keystore %srvjkskeystore%
-storepass %srvjkskeystorepass%
@rem Import the CA root certificate into the client key store
@rem Omit this step, unless you are authenticating clients.
%keytool% -import -noprompt -alias %caalias% -file %cacert% -keystore %cltjkskeystore%
-storepass %cltjkskeystorepass%
@rem Import the CA root certificate into the client ca-trust store (for ca chained
authentication)
%keytool% -import -noprompt -alias %caalias% -file %cacert% -keystore %cltcajkstruststore%
-storepass %cltcajkstruststorepass%
```

```
@rem
@echo ____________________________________________________________________________
@echo Create certificate signing requests: %srvcertreq% and %cltcertreq%
@rem
@rem Create a certificate signing request (CSR) for the server key
%keytool% -certreq -alias %srvalias% -file %srvcertreq% -keypass %srvkeypass% -keystore
%srvjkskeystore% -storepass %srvjkskeystorepass%
@rem Create a certificate signing request (CSR) for the client key
%keytool% -certreq -alias %cltalias% -file %cltcertreq% -keypass %cltkeypass% -keystore
%cltjkskeystore% -storepass %cltjkskeystorepass%
```

```
@rem
@echo ____________________________________________________________________________
@echo Sign certificate requests: %srvcertcasigned% and %cltcertcasigned%
@rem The requests are signed with the ca key in the cajkskeystore.jks keystore
@rem
@rem Sign server certificate request
%keytool% -gencert -infile %srvcertreq% -outfile %srvcertcasigned% -alias %caalias%
-keystore %cajkskeystore% -storepass %cajkskeystorepass% -keypass %cakeypass%
@rem Sign client certificate request
@rem Omit this step, unless you are authenticating clients.
%keytool% -gencert -infile %cltcertreq% -outfile %cltcertcasigned% -alias %caalias%
-keystore %cajkskeystore% -storepass %cajkskeystorepass% -keypass %cakeypass%
```

```
@rem
@echo ____________________________________________________________________________
@echo Import the signed certificates back into the key stores to create the key chain:
%srvjkskeystore% and %cltjkskeystore%
@rem
@rem Import the signed server certificate
%keytool% -import -noprompt -alias %srvalias% -file %srvcertcasigned% -keypass %srvkeypass%
-keystore %srvjkskeystore% -storepass %srvjkskeystorepass%
@rem Import the signed client certificate and key chain back into the client keystore
%keytool% -import -noprompt -alias %cltalias% -file %cltcertcasigned% -keypass %cltkeypass%
-keystore %cltjkskeystore% -storepass %cltjkskeystorepass%
@rem
@rem The CA certificate is needed in the server key store, and the client trust store
@rem to verify the key chain sent from the client or server
@echo Delete the CA certificate from %cltjkskeystore%: it causes a problem in converting
keystore to pem
@rem Omit this step, unless you are authenticating clients.
%keytool% -delete -alias %caalias% -keystore %cltjkskeystore% -storepass
%cltjkskeystorepass%
```

```
@rem
@echo ____________________________________________________________________________
@echo Create a pem client-ca trust store from the jks client-ca trust store:
%cltcapemtruststore%
@rem Omit this step, unless you are configuring a C or iOS client.
@rem
%keytool% -importkeystore -noprompt -srckeystore %cltcajkstruststore%  -destkeystore
%cltcap12truststore%  -srcstoretype jks -deststoretype pkcs12 -srcstorepass
%cltcajkstruststorepass%  -deststorepass %cltcap12truststorepass%
%openssl%\bin\openssl pkcs12 -in %cltcap12truststore%  -out %cltcapemtruststore%  -passin
pass:%cltcap12truststorepass%  -passout pass:%cltpemtruststorepass%
```

```
@rem
@echo ____________________________________________________________________________
@echo Create a pem client key store from the jks client keystore
@rem Omit this step, unless you are configuring a C or iOS client.
@rem
%keytool% -importkeystore -noprompt -srckeystore %cltjkskeystore%      -destkeystore
%cltp12keystore%     -srcstoretype jks -deststoretype pkcs12 -srcstorepass
%cltjkskeystorepass%     -deststorepass %cltp12keystorepass%
%openssl%\bin\openssl pkcs12 -in %cltp12keystore%     -out %cltpemkeystore%      -passin
pass:%cltp12keystorepass% -passout pass:%cltpemkeystorepass%
```

mqcerts.bat

Skript uvádí úložiště klíčů a certifikáty v adresáři certifikátů. Poté vytvoří ukázkového správce front MQTT a nakonfiguruje zabezpečené kanály telemetrie.

```
@echo ____________________________________________________________________________
@echo List keystores and certificates
dir %certpath%\*.* /b
```

```
@rem
@echo Create queue manager and define mqtt channels and certificate stores
call "%MQ_FILE_PATH%\mqxr\Samples\SampleMQM" >> %mqlog%
echo DEFINE CHANNEL(%chlreq%)      CHLTYPE(MQTT) TRPTYPE(TCP) PORT(%portsslreq%)
SSLCAUTH(%authreq%) SSLKEYR('%srvjkskeystore%') SSLKEYP('%srvjkskeystorepass%')
MCAUSER(%mcauser%) | runmqsc %qm% >> %mqlog%
echo DEFINE CHANNEL(%chlopt%)      CHLTYPE(MQTT) TRPTYPE(TCP) PORT(%portsslopt%)
SSLCAUTH(%authopt%) SSLKEYR('%srvjkskeystore%') SSLKEYP('%srvjkskeystorepass%')
MCAUSER(%mcauser%) | runmqsc %qm% >> %mqlog%
```
V 7.5.0.1
```
echo DEFINE CHANNEL(%chlsslreqws%) CHLTYPE(MQTT) TRPTYPE(TCP) PORT(%portsslreqws%)
SSLCAUTH(%authreq%) SSLKEYR('%srvjkskeystore%') SSLKEYP('%srvjkskeystorepass%')
MCAUSER(%mcauser%) PROTOCOL(HTTP) | runmqsc %qm% >> %mqlog%
echo DEFINE CHANNEL(%chlssloptws%) CHLTYPE(MQTT) TRPTYPE(TCP) PORT(%portssloptws%)
SSLCAUTH(%authopt%) SSLKEYR('%srvjkskeystore%') SSLKEYP('%srvjkskeystorepass%')
MCAUSER(%mcauser%) PROTOCOL(HTTP) | runmqsc %qm% >> %mqlog%
echo DEFINE CHANNEL(%chlws%)       CHLTYPE(MQTT) TRPTYPE(TCP) PORT(%portws%)
MCAUSER(%mcauser%) PROTOCOL(HTTP) | runmqsc %qm% >> %mqlog%
@echo MQ logs saved in %mqlog%echo
```

Authenticating an MQTT client Java app with JAAS

Naučte se, jak ověřit klienta pomocí produktu JAAS. Complete the steps in this task to modify the sample program `JAASLoginModule.java` and configure IBM WebSphere MQ to authenticate an MQTT client Java app with JAAS.

Než začnete

1. Aplikaci Klient MQTT pro produkt Java můžete spustit na libovolné platformě s JSE 1.5 nebo vyšším, které je "Java Kompatibilní". Viz téma Systémové požadavky pro IBM Mobile Messaging and M2M Client Pack.
2. Je-li mezi vaším klientem a serverem brána firewall, zkontrolujte, zda neblokuje provoz produktu MQTT .
3. Musíte mít přístup k ukázkám produktu MQXR `JAASLoginModule` a `JAASPrincipal` Java v instalaci produktu IBM WebSphere MQ . Ukázky jsou v cestě `%MQ_FILE_PATH%\mqxr\samples.`
4. Proveďte kroky uvedené v Windows nebo Linux; příklady jsou převzaty z Windows.
5. Chcete-li dokončit krok “1” na stránce 72, musíte mít autorizaci k vytvoření správce front `MQXR_SAMPLE_QM` v systému IBM WebSphere MQ.

Informace o této úloze

V úloze získáte výstup identifikačních parametrů klienta MQTT `Sample` z vaší verze produktu `JAASLoginModule`. Při zápisu parametrů klienta dochází k úpravě ukázkového programu `JAASLoginModule` a konfiguraci produktu IBM WebSphere MQ za účelem načtení vaší verze produktu `JAASLoginModule`.

Postup

1. Proveďte kroky uvedené v “Kompilujte a spusťte všechny ukázkové aplikace Java klienta produktu MQTT z produktu Eclipse” na stránce 15 a spusťte klienta Paho MQTT `Sample` .

   Vaším cílem je připravit vývojové prostředí pro vývoj a testování ověření produktu JAAS . Chcete-li přizpůsobit modul ověření produktu JAAS , musíte prostředí pro vývoj produktu Java upravit. V tomto příkladu jste spustili ukázkového klienta Paho pro produkt Java za účelem testování konfigurace produktu JAAS . Pro zjednodušení použijte stejné vývojové prostředí, abyste upravili jak ukázkového klienta, tak ukázkového přihlašovacího modulu JAAS . Případně otestujete váš přihlašovací modul JAAS s klientem MQTT pro C nebo jiného klienta MQTT .

2. Volitelné: Přidejte parametr jména uživatele a hesla do ukázky MQTT Paho .

Poznámka: Pokud váš klient Paho for Java obsahuje parametry jména uživatele a hesla, tento krok není nutný. Zkontrolujte web pro stažení pro aktualizaci. Informace naleznete v tématu Soubory ke stažení komunity společnostiIBM, v opačném případě změňte kopii produktu `Sample.java`.

a) Otevřete průzkumník balíků v balíku produktu `org.eclipse.paho.sample.mqttv3app` v ukázkovým projektu produktu Paho .

b) Pravým tlačítkem myši klepněte na položku `Sample.java` **Kopírovat** > **Vložit**. V okně **Konflikt názvů** zadejte název `SampleForJAAS`.

c) Přidejte následující řádky kódu do metody `main` .

i) Po řádku "`boolean ssl = false;`"deklarujte proměnné `userName` a `password` :

```
String password = null;
String userName = null;
```

Pro kompatibilitu se staršími servery MQTT standardně nenastavujte heslo a parametry jména uživatele.

ii) Po řádku, "`case 'v': ssl = Boolean.valueOf(args[++i]).booleanValue(); break;`", analyzujte dva nové vstupní parametry:

```
case 'u': userName = args[++i]; break;
case 'z': password = args[++i]; break;
```

iii) Před řádkem, "`if (action.equals("publish")) {`", přidejte `userName` a `password` do argumentů konstruktoru `Sample` :

```
Sample sampleClient = new Sample(url, clientId, cleanSession, quietMode, userName,
password);
```

d) Přidejte `userName` a `password` do konstruktoru `Sample`.

Změňte:

```
public Sample(String brokerUrl, String clientId, boolean cleanSession,
       boolean quietMode) throws MqttException {
```

Do:

```
public Sample(String brokerUrl, String clientId, boolean cleanSession,
       boolean quietMode, String userName, char[] password) throws MqttException {
```

e) Přidejte následující řádky kódu do konstruktoru `Sample` .

Po řádku "`conOpt.setCleanSession(clean);`"nastavte proměnné `userName` a `password` v objektu `conOpt` v metodě `Sample` :

```
if(password != null ) {
   conOpt.setPassword(this.password.toCharArray());
}
if(userName != null) {
   conOpt.setUserName(this.userName);
}
```

f) V metodách `publish` a `subscribe` upravte následující řádky kódu:

Změňte řádek "`client.connect();`" na

```
client.connect(conOpt);
```

3. Vytvořte projekt Java , `JAASSample`, pro váš příklad JAAS .

a) Ve svém pracovním prostoru produktu Eclipse otevřete průvodce **Nový projekt Java** : Klepněte na volbu **Soubor** > **Nový** > **Projekt Java**.

b) Do pole **Název projektu** zadejte `JAASSample`.

c) Ve volbách prostředí JRE vyberte volbu `J2SE-1.5` jako prostředí implementace prostředí JRE a klepněte na tlačítko **Další**.

Prostředí JRE se musí shodovat s prostředím JRE, které váš server IBM WebSphere MQ spouští. IBM WebSphere MQ Version 7.5 spustí `J2SE-1.5`.

d) V okně **Nastavení prostředí Java** klepněte na kartu **Knihovny** a poté klepněte na volbu **Přidat externí soubory JAR ...**. Procházet a hledatAdresář `%MQ_FILE_PATH%\mqxr\lib` a vyberte volbu **MQXR.jar**; klepněte na tlačítko **Dokončit**.

4. Naimportujte ukázky JAAS `JAASLoginModule` a `JAASPrincipal` .

a) Klepněte pravým tlačítkem myši na projekt produktu `JAASSample` v Průzkumníku balíků **Importovat ...** > **Obecné** > **Systém souborů** a klepněte na tlačítko **Další**.

b) Přejděte na `%MQ_FILE_PATH%\mqxr\samples` a zkontrolujte `JAASLoginModule.java` a `JAASPrincipal.java`; klepněte na **Dokončit**.

c) Vyberte a klepněte pravým tlačítkem myši na oba soubory Java v Průzkumníku balíků, **Refaktorovat ...** > **Přesunout**.

d) V okně **Přesunout** ověřte, že je jako cíl pro oba prvky vybráno `JAASSample` a klepněte na volbu **Vytvořit balík ...**

e) Zadejte `samples` do pole **Název** v průvodci **Nový balík Java** ; klepněte na **Dokončit** > **OK**

Platforma Eclipse sestaví importované třídy produktu Java s počtem varování o nepoužitých hodnotách.

5. Přejmenovat třídu `JAASLoginModule`

Přejmenujte třídu tak, aby ji bylo snazší odlišit od ukázkové třídy `JAASLoginModule` , která je dodávána s produktem IBM WebSphere MQ.

a) Klepněte pravým tlačítkem myši na položku `JAASloginModule.java` v průzkumníku balíků **Refaktorovat ...** > **Přejmenovat**.

b) V okně **Přejmenovat kompilační jednotku** změňte pole **Nový název** z `JAASloginModule` na `MyJAASloginModule`; klepněte na **Dokončit**.

6. Upravte třídu `MyJAASloginModule` pro výstup obsahu polí zpětného volání.

a) Přidejte následující řádek kódu do `MyJAASloginModule.java`.

```
System.out.println("Username=" + username
                 + "\nPassword=" + new String(password)
                 + "\nClientId=" + clientId
                 + "\nNetwork address=" + networkAddress);
```

Umístěte řádky těsně před příkaz:`"if (true) loggedIn = true;"`

b) Stiskněte tlačítko `CTRL+Shift+O` , chcete-li reorganizovat importy a uložit soubor.

7. Přejmenujte `JAASPrincipal` na `MyJAASPrincipal`.

Přejmenujte třídu tak, aby nedošlo k záměně s ukázkovou třídou `JAASPrincipal` . V tomto příkladu ponechte obsah třídy `MyJAASPrincipal` nezměněný.

8. Zadejte ID uživatele, který spouští správce front, zpracovává oprávnění `read` a `execute` pro vaše třídy JAAS .

a) V Průzkumníku Windows otevřete adresář pracovního prostoru produktu Eclipse . V tomto příkladě je umístění pracovního prostoru Eclipse představováno proměnnou Eclipse , *`workspace_loc`*.

b) Přejděte do adresáře, který obsahuje vaše třídy `MyJAASLoginModule` a `MyJAASPrincipal` .

Cesta k adresáři je *`workspace_loc`*`\JAASSample\bin\samples`

c) Vyberte a poté klepněte pravým tlačítkem myši na obě třídy a poté klepněte na volbu **Vlastnosti**; klepněte na kartu **Zabezpečení** v okně **Vlastnosti** .

d) Klepněte na tlačítko **Přidat ...**. Zadejte název objektu `mqma` klepněte na volbu **Zkontrolovat názvy** a ověřte jej. Klepněte na tlačítko **OK**.

e) Vyberte položku **mqm** v seznamu **Názvy skupin nebo uživatelů** a zaškrtněte políčko **číst a provádět** a **číst** v seznamu oprávnění pro produkt `mqm`; klepněte na tlačítko OK.

9. Nakonfigurujte produkt IBM WebSphere MQ pro spuštění třídy `MyJAASLoginModule` .

a) Přidejte do konfigurace produktu IBM WebSphere MQ soubor `service.env` , abyste nadefinovali cesty ke třídě, abyste načetli třídu `MyJAASLoginModule` .

Vytvořte soubor `service.env` s následujícím příkazem pro cestu ke třídám ve svém adresáři *WMQ_DATA_PATH* :

```
CLASSPATH=user.dir\JAASSample\bin
```

Kde *user.dir* je kořenový adresář adresáře pro soubory tříd, které jsou kompilovány ve vašem pracovním prostoruEclipse . Adresář *WMQ_DATA_PATH* obsahuje adresář `qmgrs` . Viz Další proměnné prostředí.

Tip: `CLASSPATH=`*user.dir*`\JAASSample\bin` může být jediným příkazem v souboru `service.env`

b) Přidejte sekci `MyJAASStanza`do souboru `jaas.config` , abyste označili vaši třídu `MyJAASLoginModule` vzhledem k cestám tříd v souboru `service.env` .

`jaas.config` je v adresáři `mqxr` správce front; *WMQ_DATA_PATH*`\Qmgrs\MQXR_SAMPLE_QM\mqxr`.

Oddíl je:

```
MyJAASStanza {
  samples.MyJAASLoginModule required debug=true;
};
```

c) Konfigurujte kanál IBM WebSphere MQ Telemetry s názvem sekce konfigurace produktu JAAS .

V příkazovém okně spusťte následující příkaz:

```
echo DEFINE CHANNEL('MyJAAS') CHLTYPE(MQTT) TRPTYPE(TCP) PORT(1890)
JAASCFG('MyJAASStanza') | runmqsc MQXR_SAMPLE_QM
```

Příkaz spojí kanál `MyJAAS` se souborem `MyJAASStanza` v souboru `jaas.config` . Neuvedení volby `MCAUSER` nebo zadání volby `USECLTID` na definici kanálu umožňuje kanál autorizovat přístup k prostředkům správce front s uživatelským jménem dodávaným klientským programem MQTT . V tomto příkladu je jméno uživatele zadané klientem nastaveno na hodnotu "`Guest`". V tomto příkladu se také používají existující autorizace pro `Guest` nastavené příkazovým souborem `SampleMQM` .

10. Restartujte službu IBM WebSphere MQ Telemetry , abyste si přečetli nová konfigurační data. Chcete-li znovu spustit službu IBM WebSphere MQ Telemetry , spustit správce front nebo službu z produktu IBM WebSphere MQ Explorer, nebo spustit následující příkazy pro ukázkovou konfiguraci:

```
echo stop  service(SYSTEM.MQXR.SERVICE) | runmqsc  MQXR_SAMPLE_QM
echo start service(SYSTEM.MQXR.SERVICE) | runmqsc  MQXR_SAMPLE_QM
```

11. Spusťte program `Sample` .

Chcete-li nakonfigurovat konfiguraci spuštění pro produkt `SampleForJAAS` , postupujte podle stejné procedury jako v kroku “1” na stránce 72s následujícími úpravami:

a) Nastavte číslo portu na `1890` tak, aby odpovídalo konfiguraci kanálu produktu MQTT .

b) Přidejte parametry `-u Guest -z password` do hesel na kartě **(x) = Argumenty** pro konfigurace odběratele a vydavatele, které jste vytvořili pro program `Sample` .

Ukázkové programy se spouštějí bez jakékoli změny ve výstupu, kromě čísla portu je nyní `1890` spíše než `1883`.

V adresáři *WMQ_DATA_PATH*`\Qmgrs\MQXR_SAMPLE_QM` otevřete soubor `mqxr.stdout` . Výstup z `MyJAASLoginModule` je zapsán do `mqxr.stdout`:

```
Username=Guest
Password=passord
```

```
ClientId=SampleJavaV3_subscribe
Network address=/127.0.0.1
```

Jak pokračovat dále

Pokud váš příklad nefunguje, přečtěte si téma odstraňování problémů pro JAAS; “Vyřešení problému: přihlašovací modul JAAS , který není volán službou telemetrie” na stránce 183a zkuste tyto rady pro ladění programu.

1. Přidejte `-verbose` do parametrů v *WMQ_DATA_PATH*`\Qmgrs\MQXR_SAMPLE_QM\mqxr\java.properties`. V tomto protokolu můžete zjistit, zda byla třída úspěšně načtena.

   Výstup se zapisuje do *WMQ_DATA_PATH*`\Qmgrs\MQXR_SAMPLE_QM\mqxr.stderr`.
2. V produktu *WMQ_DATA_PATH*`\Qmgrs\MQXR_SAMPLE_QM\errors\mqxr.log` vyhledejte výjimky, které jsou generovány v produktu `MyJAASLoginModule`. Pokusíte-li se například o výstup s hodnotou null `password`, což je znakové pole a nikoli řetězec, dojde k výjimce.
3. Prohlédni si *WMQ_DATA_PATH*`\Qmgrs\MQXR_SAMPLE_QM\errors\AMQERR01.log`. Není-li jméno uživatele v produktu `userName` autorizováno pro přístup k prostředkům správce front a kanál je konfigurován bez volby `MCAUSER` nebo `USECLTID` , jsou zde nahlášeny jakékoli chyby.
4. Zkontrolujte, zda název oddílu v souboru *WMQ_DATA_PATH*`\Qmgrs\MQXR_SAMPLE_QM\mqxr\jaas.config` je stejný jako název v kanálu produktu MQTT , který je konfigurován pro port, ke kterému se klient `Sample` pokouší připojit.
5. Zkontrolujte, zda cesta v sekci odpovídá cestě ke třídě `MyJAASLoginModule` v souboru Eclipse; například:

   ```
   MyJAASStanza {
     samples.MyJAASLoginModule required debug=true;
   };
   ```
6. Chcete-li vyloučit, zda je chyba v cestě ke třídě v souboru `service.env` v *WMQ_DATA_PATH* správně vyzvedne, změňte řádek "`set CLASSPATH=%MQXRCLASSPATH%;%CLASSPATH%`" v souboru `%MQ_FILE_PATH%\mqxr\bin\controlMQXR.BAT` tak, aby obsahoval vaši cestu ke třídám. Cestu ke třídám lze také opakovat. Cesta ke třídám však neobsahuje cestu ke třídám, která je nastavena v produktu `service.env`, takže tato cesta bude fungovat pouze v případě, že upravíte soubor `controlMQXR.BAT` .

Související pojmy

“Zabezpečení produktu MQTT” na stránce 51
Tři koncepty jsou základní pro zabezpečení produktu MQTT : identita, ověřování a autorizace. Identita je o pojmenování klienta, který je autorizován a kterému je uděleno oprávnění. Ověřování je o prokazování identity klienta a autorizace se týká správy práv, která jsou klientovi udělena.

“Konfigurace kanálu JAAS kanálu telemetrie” na stránce 114
Nakonfigurujte službu JAAS tak, aby ověřoval identitu `Jméno uživatele` odeslané klientem.

Související úlohy

“Vyřešení problému: přihlašovací modul JAAS , který není volán službou telemetrie” na stránce 183
Zjistěte, zda váš přihlašovací modul JAAS není volán službou telemetrie (MQXR), a nakonfigurujte službu JAAS , chcete-li problém opravit.

“Sestavení a spuštění zabezpečeného produktu Ukázková aplikace klienta MQTT Java” na stránce 54
Na základě příkladu Windows můžete začít pracovat se zabezpečenou ukázkovou aplikací Java na serveru buď IBM MessageSight , nebo IBM WebSphere MQ jako server MQTT. Aplikaci Klient MQTT pro produkt Java můžete spustit na libovolné platformě s JSE 1.5 nebo vyšším, které je "Java Kompatibilní"

“Připojení ukázkové aplikace Java klienta MQTT k produktu Android přes SSL” na stránce 62
Vstupte a vyběhněte s ukázkovým klientem Android MQTT připojeným k produktu IBM WebSphere MQ přes SSL.

Související informace
Další proměnné prostředí

Připojení Klient systému zpráv MQTT pro produkt JavaScript přes SSL a WebSockets

Webovou aplikaci bezpečně připojte k produktu IBM WebSphere MQ pomocí ukázkových HTML stránek Klient systému zpráv MQTT pro produkt JavaScript s SSL a WebSocket protocol.

Než začnete

1. Musíte mít přístup k serveru MQTT version 3 , který podporuje MQTT protocol přes WebSockets.
2. Prohlížeč musí podporovat SSL a WebSocket protocol. Viz “Omezení v podpoře prohlížečů pro webové aplikace mobilního systému zpráv přes SSL” na stránce 172.
3. Kanály SSL musí být spuštěny.

Informace o této úloze

Prostřednictvím této úlohy lze spustit klienta systému zpráv produktu MQTT pro ukázkové stránky produktu JavaScript prostřednictvím protokolu SSL. Úloha vás přesměruje na “Generování klíčů a certifikátů” na stránce 95 , abyste vytvořili certifikáty a nakonfigurovali IBM WebSphere MQ.

Zabezpečte kanál SSL buď pomocí klíčů podepsanými certifikační autoritou, nebo pomocí klíčů podepsaným držitelem.

Postup

1. Zvolte server MQTT , ke kterému se můžete připojit k aplikaci klienta.

   Server musí podporovat MQTT protocol přes zabezpečení WebSockets.

   - IBM MessageSighta vydání produktu IBM WebSphere MQ verze 7.5.0.1 a novější, toto proveďte.
2. Volitelné: Nainstalujte sadu pro vývoj Java (JDK) verze 7 nebo novější.

   If you are setting up a test system, and you want to use self-signed certificates, you need to use the JDK Version 7 **keytool** command to certify your certificates. Pokud nastavujete produkční systém a odesílání požadavků na podpis certifikátu (CSR) na externí certifikační autoritu, nepotřebujete sadu JDK verze 7.
3. Vytvořte a spusťte skripty pro generování dvojic klíčů a certifikátů a nakonfigurujte produkt IBM WebSphere MQ jako server MQTT .

   Postupujte podle kroků v části “Generování klíčů a certifikátů” na stránce 95 a vytvořte a spusťte skripty. Tyto skripty jsou také vypsány v “Ukázkové skripty pro konfiguraci certifikátů SSL pro produkt Windows” na stránce 79.

   Obecný název certifikátu serveru se musí shodovat s názvem DNS kanálu serveru. Některé prohlížeče přijímají certifikáty, které obsahují seznam běžných názvů; například:

   ```
   "CN=localhost, CN=*.example.com"
   ```

   Ostatní prohlížeče přijímají pouze jedno obecné jméno. Například, Firefoxaž do verze 18 přijímá pouze jeden obecný název. Pozdější verze se mohou lišit.
4. Zkontrolujte, zda jsou kanály zabezpečení SSL spuštěny a nastaveny dle očekávání.

   V IBM WebSphere MQ zadejte následující příkaz do okna příkazového řádku:

   - Linux

   ```
   echo 'DISPLAY CHSTATUS(SSL*) CHLTYPE(MQTT) ALL' | runmqsc MQXR_SAMPLE_QM
   echo 'DISPLAY CHANNEL(SSL*) CHLTYPE(MQTT) ALL' | runmqsc MQXR_SAMPLE_QM
   ```

- Windows

```
echo DISPLAY CHSTATUS(SSL*) CHLTYPE(MQTT) ALL | runmqsc MQXR_SAMPLE_QM
echo DISPLAY CHANNEL(SSL*) CHLTYPE(MQTT) ALL | runmqsc MQXR_SAMPLE_QM
```

5. Nainstalujte certifikáty do úložiště certifikátů prohlížeče.

   Jako příklad zvolte jeden z následujících certifikátů serveru:

   a. Pro certifikát serveru s automatickým podpisem je certifikát `srvcertselfsigned.cer`.

   b. Pro certifikát serveru podepsaný vaší soukromou certifikační autoritou je certifikát `cacert.cer`.

   c. Pro certifikát serveru podepsaný externí certifikační autoritou, zkontrolujte kořenový certifikát certifikační autority, který je již instalován v úložišti certifikátů.

   V závislosti na podpoře dostupné ve vašem prohlížeči nainstalujte produkt `cacert.cer` do seznamu důvěryhodných kořenových certifikačních autorit. Viz “Omezení v podpoře prohlížečů pro webové aplikace mobilního systému zpráv přes SSL” na stránce 172.

6. Volitelné: Ověřte klienta.

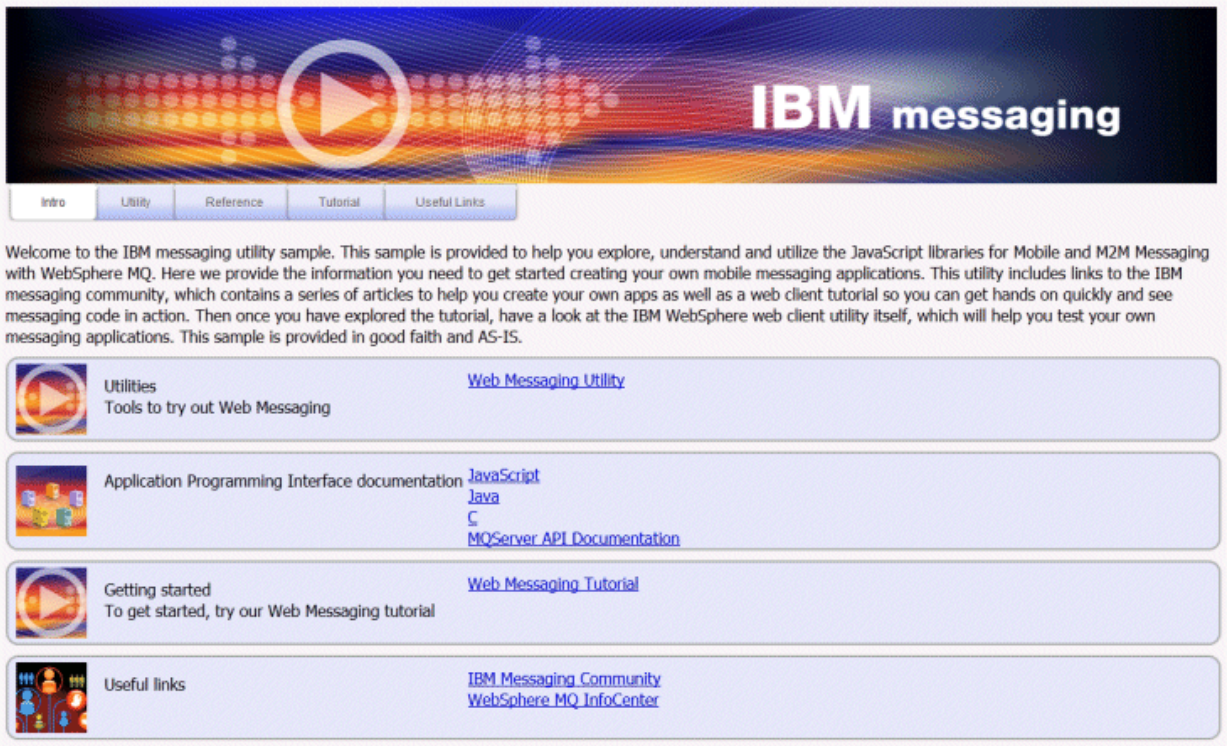
   V příkladu nainstalujete produkt `cacert.cer` k ověření serveru a zašifrujete kanál, ale ne k ověření klienta. Chcete-li ověřit klienta, musíte instalovat úložiště klíčů klienta, `cltkeystore.p12`, v prohlížeči. Ne všechny prohlížeče podporují ověření klientů. Viz “Omezení v podpoře prohlížečů pro webové aplikace mobilního systému zpráv přes SSL” na stránce 172.

7. Připojte se k zabezpečenému kanálu produktu WebSockets .

   Otevřete prohlížeč a zadejte adresu URL kanálu produktu WebSockets do adresního řádku; v tomto příkladu:

```
https://localhost:8886
```

   IBM WebSphere MQ reaguje s první stránkou Ukázkové stránky klienta systému zpráv MQTT JavaScript.



Dojde-li k selhání připojení a vy jste spustili ukázkové skripty pro nastavení ukázkového správce front MQTT , zkuste se připojit k normálnímu kanálu produktu WebSockets na portu 1886. Úspěch na portu 1886 izoluje selhání připojení přes SSL.

```
https://localhost:1886
```

Související pojmy

"Aplikace Klient systému zpráv MQTT pro produkt JavaScript a webové aplikace" na stránce 116

"Jak naprogramovat aplikace systému zpráv v produktu JavaScript" na stránce 120

Související úlohy

"Generování klíčů a certifikátů" na stránce 95

Postupujte podle této procedury, chcete-li generovat klíče a certifikáty pro klienty Java a C, včetně aplikací Android a iOS , a serverů IBM WebSphere MQ a IBM MessageSight .

"Začínáme s produktem Klient systému zpráv MQTT pro produkt JavaScript" na stránce 23

Můžete začít pracovat se serverem Klient systému zpráv MQTT pro produkt JavaScript zobrazením ukázkové domovské stránky klienta systému zpráv a procházením prostředků, na které se odkazuje. Chcete-li zobrazit tuto domovskou stránku, nakonfigurujte server MQTT tak, aby přijímal připojení ze serveru Ukázkové stránky klienta systému zpráv MQTT JavaScript, pak napíšete adresu URL, kterou jste nakonfigurovali na serveru do webového prohlížeče. Produkt Klient systému zpráv MQTT pro produkt JavaScript se automaticky spustí na vašem zařízení a zobrazí se ukázková domovská stránka klienta systému zpráv. Tato stránka obsahuje odkazy na obslužné programy, dokumentaci k programovacímu rozhraní, výukový program a další užitečné informace.

Související odkazy

"Omezení v podpoře prohlížečů pro webové aplikace mobilního systému zpráv přes SSL" na stránce 172

Mezi různými prohlížeči existují rozdíly ve schopnosti, na různých platformách. Porozumění těmto rozdílům pomáhá konfigurovat vaše aplikace, vydavatele certifikátů (CA) a certifikáty klientů pro připojení pomocí Klient systému zpráv MQTT pro produkt JavaScript přes SSL a WebSockets.

Ukázkové skripty pro konfiguraci certifikátů SSL pro produkt Windows

** **

Ukázkové příkazové soubory vytvoří certifikáty a úložiště certifikátů, jak je popsáno v krocích v úloze. Kromě toho tento příklad nastaví správce front klienta MQTT pro použití úložiště certifikátů serveru. Příklad odstraní a znovu vytvoří správce front voláním skriptu `SampleMQM.bat`, který je poskytnut s IBM WebSphere MQ.

`initcert.bat`

`initcert.bat` nastavuje názvy a cesty k certifikátům a jiným parametrům, které jsou vyžadovány příkazy **keytool** a **openSSL**. Nastavení jsou popsána v komentářích ve skriptu.

```
@echo off
@rem Set the path where you installed the MQTT SDK
@rem and short cuts to the samples directories.
set SDKRoot=C:\MQTT
set jsamppath=%SDKRoot%\sdk\clients\java\samples
set csamppath=%SDKRoot%\sdk\clients\c\samples
```

```
@rem Set the paths to Version 7 of the JDK
@rem and to the directory where you built the openSSL package.
@rem Set short cuts to the tools.
set javapath=C:\Program Files\IBM\Java70
set keytool="%javapath%\jre\bin\keytool.exe"
set ikeyman="%javapath%\jre\bin\ikeyman.exe"
set openssl=%SDKRoot%\openSSL
set runopenssl="%openssl%\bin\openssl"
```

```
@rem Set the path to where certificates are to be stored,
@rem and set global security parameters.
@rem Omit set password, and the security tools prompt you for passwords.
@rem Validity is the expiry time of the certificates in days.
set certpath=%SDKRoot%\Certificates
```

```
set password=password
set validity=5000
set algorithm=RSA
```

```
@rem Set the certificate authority (CA) jks keystore and certificate parameters.
@rem Omit this step, unless you are defining your own certificate authority.
@rem The CA keystore contains the key-pair for your own certificate authority.
@rem You must protect the CA keystore.
@rem The CA certificate is the self-signed certificate authority public certificate.
@rem It is commonly known as the CA root certificate.
set caalias=caalias
set cadname="CN=mqttca.ibm.id.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
set cakeypass=%password%
@rem ca key store
set cajkskeystore=%certpath%\cakeystore.jks
set cajkskeystorepass=%password%
@rem ca certificate (root certificate)
set cacert=%certpath%\cacert.cer
```

```
@rem Set the server jks keystore and certificate parameters.
@rem The server keystore contains the key-pair for the server.
@rem You must protect the server keystore.
@rem If you then export the server certificate it is self-signed.
@rem Alternatively, if you export a certificate signing request (CSR)
@rem from the server keystore for the server key,
@rem and import the signed certificate back into the same keystore,
@rem it forms a certificate chain.
@rem The certificate chain links the server certificate to the CA.
@rem When you now export the server certificate,
@rem the exported certificate includes the certificate chain.
set srvalias=srvalias
set srvdname="CN=mqttserver.ibm.id.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
set srvkeypass=%password%
@rem server key stores
set srvjkskeystore=%certpath%\srvkeystore.jks
set srvjkskeystorepass=%password%
@rem server certificates
set srvcertreq=%certpath%\srvcertreq.csr
set srvcertcasigned=%certpath%\srvcertcasigned.cer
set srvcertselfsigned=%certpath%\srvcertselfsigned.cer
```

```
@rem Set the client jks keystore and certificate parameters
@rem Omit this step, unless you are authenticating clients.
@rem The client keystore contains the key-pair for the client.
@rem You must protect the client keystore.
@rem If you then export the client certificate it is self-signed.
@rem Alternatively, if you export a certificate signing request (CSR)
@rem from the client keystore for the client key,
@rem and import the signed certificate back into the same keystore,
@rem it forms a certificate chain.
@rem The certificate chain links the client certificate to the CA.
@rem When you now export the client certificate,
@rem the exported certificate includes the certificate chain.
set cltalias=cltalias
set cltdname="CN=mqttclient.ibm.id.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
set cltkeypass=%password%
@rem client key stores
set cltjkskeystore=%certpath%\cltkeystore.jks
set cltjkskeystorepass=%password%
set cltcertreq=%certpath%\cltcertreq.csr
set cltcertcasigned=%certpath%\cltcacertsigned.cer
set cltcertselfsigned=%certpath%\cltcertselfsigned.cer
```

```
@rem Set the paths to the client truststores signed by CA and signed by server key.
@rem You only need to define one of the trust stores.
@rem A trust store holds certificates that you trust,
@rem which are used to authenticate untrusted certificates.
@rem In this example, when the client authenticates the MQTT server it connects to,
@rem it authenticates the certificate it is sent by the server
@rem with the certificates in its trust store.
@rem For example, the MQTT server sends its server certificate,
@rem and the client authenticates it with either the same server certificate
@rem that you have stored in the cltsrvtruststore.jks trust store,
@rem or against the CA certificate, if the server certificate is signed by the CA.
set cltcajkstruststore=%certpath%\cltcatruststore.jks
set cltcajkstruststorepass=%password%
```

```
set cltsrvjkstruststore=%certpath%\cltsrvtruststore.jks
set cltsrvjkstruststorepass=%password%
```

```
@rem Set the paths to the client PKCS12 and PEM key and trust stores.
@rem Omit this step, unless you are configuring a C or iOS client.
@rem You only need to define either one of the trust stores for storing CA
@rem or server signed server certificates.
set cltp12keystore=%certpath%\cltkeystore.p12
set cltp12keystorepass=%password%
set cltpemkeystore=%certpath%\cltkeystore.pem
set cltpemkeystorepass=%password%
set cltcap12truststore=%certpath%\cltcatruststore.p12
set cltcap12truststorepass=%password%
set cltcapemtruststore=%certpath%\cltcatruststore.pem
set cltcapemtruststorepass=%password%
set cltsrvp12truststore=%certpath%\cltsrvtruststore.p12
set cltsrvp12truststorepass=%password%
set cltsrvpemtruststore=%certpath%\cltsrvtruststore.pem
set cltsrvpemtruststorepass=%password%
```

```
@rem set WMQ Variables
set authopt=NEVER
set authreq=REQUIRED
set qm=MQXR_SAMPLE_QM
set host=localhost
set mcauser='Guest'
set portsslopt=8884
set chlopt=SSLOPT
set portsslreq=8885
set chlreq=SSLREQ
V 7.5.0.1 set portws=1886
set chlws=PLAINWS
set chlssloptws=SSLOPTWS
set portssloptws=8886
set chlsslreqws=SSLREQWS
set portsslreqws=8887
set mqlog=%certpath%\wmq.log
```

cleancert.bat

Příkazy ve skriptu `cleancert.bat` odstraní správce front klienta MQTT, aby se zajistilo, že úložiště certifikátů serveru není uzamčeno, a potom odstraní všechna úložiště klíčů a certifikáty, které jsou vytvořeny ukázkovými skripty zabezpečení.

```
@rem Delete the MQTT sample queue manager, MQXR_SAMPLE_QM
call "%MQ_FILE_PATH%\bin\setmqenv" -s
endmqm -i %qm%
dltmqm %qm%
```

```
@rem Erase all the certificates and key stores created by the sample scripts.
erase %cajkskeystore%
erase %cacert%
erase %srvjkskeystore%
erase %srvcertreq%
erase %srvcertcasigned%
erase %srvcertselfsigned%
erase %cltjkskeystore%
erase %cltp12keystore%
erase %cltpemkeystore%
erase %cltcertreq%
erase %cltcertcasigned%
erase %cltcertselfsigned%
erase %cltcajkstruststore%
erase %cltcap12truststore%
erase %cltcapemtruststore%
erase %cltsrvjkstruststore%
erase %cltsrvp12truststore%
erase %cltsrvpemtruststore%
erase %mqlog%
@echo Cleared all certificates
dir %certpath%\*.* /b
```

genkeys.bat

Příkazy ve skriptu `genkeys.bat` vytvářejí dvojice klíčů pro soukromé certifikační autority, server a klienta.

```
@rem
@echo ------------------------------------------------------------------------
@echo Generate %caalias%, %srvalias%, and %cltalias% key-pairs in %cajkskeystore%,
%srvjkskeystore%, and %cltjkskeystore%
@rem
@rem -- Generate a client certificate and a private key pair
@rem Omit this step, unless you are authenticating clients.
%keytool% -genkeypair -noprompt -alias %cltalias% -dname %cltdname% -keystore
%cltjkskeystore% -storepass %cltjkskeystorepass% -keypass %cltkeypass% -keyalg %algorithm%
-validity %validity%
```

```
@rem -- Generate a server certificate and private key pair
%keytool% -genkeypair -noprompt -alias %srvalias% -dname %srvdname% -keystore
%srvjkskeystore% -storepass %srvjkskeystorepass% -keypass %srvkeypass% -keyalg %algorithm%
-validity %validity%
```

```
@rem Create CA, client and server key-pairs
@rem  -- Generate a CA certificate and private key pair - The extension asserts this is a
certificate authority certificate, which is required to import into firefox
%keytool% -genkeypair -noprompt -ext bc=ca:true -alias %caalias%  -dname %cadname%
-keystore %cajkskeystore% -storepass %cajkskeystorepass%  -keypass %cakeypass% -keyalg
%algorithm% -validity %validity%
```

sscerts.bat

Příkazy ve skriptu `sscerts.bat` exportují certifikáty podepsané držitelem klienta a serveru z vlastních úložišť klíčů a importují certifikát serveru do úložiště údajů o důvěryhodnosti klienta a certifikát klienta do úložiště klíčů serveru. Server nemá úložiště údajů o důvěryhodnosti. Příkazy vytvoří úložiště údajů o důvěryhodnosti klienta ve formátu PEM z úložiště údajů o důvěryhodnosti JKS klienta.

```
@rem
@echo ------------------------------------------------------------------------
@echo Export self-signed certificates:  %srvcertselfsigned% and %cltcertselfsigned%
@rem Export Server public certificate
%keytool% -exportcert -noprompt -rfc -alias %srvalias% -keystore %srvjkskeystore%
-storepass %srvjkskeystorepass% -file %srvcertselfsigned%
@rem Export Client public certificate
@rem Omit this step, unless you are authenticating clients.
%keytool% -exportcert -noprompt -rfc -alias %cltalias% -keystore %cltjkskeystore%
-storepass %cltjkskeystorepass% -file %cltcertselfsigned%
```

```
@rem
@echo ------------------------------------------------------------------------
@echo Add selfsigned server certificate %srvcertselfsigned% to client trust store:
%cltsrvjkstruststore%
@rem Import the server certificate into the client-server trust store (for server self-
signed authentication)
%keytool% -import -noprompt -alias %srvalias% -file %srvcertselfsigned% -keystore
%cltsrvjkstruststore% -storepass %cltsrvjkstruststorepass%
```

```
@rem
@echo ------------------------------------------------------------------------
@echo Add selfsigned client certificate %cltcertselfsigned% to server trust store:
%srvjkskeystore%
@rem Import the client certificate into the server trust store (for client self-signed
authentication)
@rem Omit this step, unless you are authenticating clients.
%keytool% -import -noprompt -alias %cltalias% -file %cltcertselfsigned% -keystore
%srvjkskeystore% -storepass %srvjkskeystorepass%
```

```
@rem
@echo ------------------------------------------------------------------------
@echo Create a pem client-server trust store from the jks client-server trust store:
%cltsrvpemtruststore%
@rem
%keytool% -importkeystore -noprompt -srckeystore %cltsrvjkstruststore% -destkeystore
```

```
%cltsrvp12truststore% -srcstoretype jks -deststoretype pkcs12 -srcstorepass
%cltsrvjkstruststorepass% -deststorepass %cltsrvp12truststorepass%
%openssl%\bin\openssl pkcs12 -in %cltsrvp12truststore% -out %cltsrvpemtruststore% -passin
pass:%cltsrvp12truststorepass% -passout pass:%cltsrvpemtruststorepass%@rem
@rem
@echo ____________________________________________________________________________
@echo Create a pem client key store from the jks client keystore
@rem Omit this step, unless you are configuring a C or iOS client.
@rem
%keytool% -importkeystore -noprompt -srckeystore %cltjkskeystore%      -destkeystore
%cltp12keystore%     -srcstoretype jks -deststoretype pkcs12 -srcstorepass
%cltjkskeystorepass%     -deststorepass %cltp12keystorepass%
%openssl%\bin\openssl pkcs12 -in %cltp12keystore%     -out %cltpemkeystore%      -passin
pass:%cltp12keystorepass% -passout pass:%cltpemkeystorepass%
```

cacerts.bat

Skript importuje kořenový certifikát certifikační autority soukromých úložišť klíčů. Kořenový certifikát CA je potřebný k vytvoření svazku klíčů mezi kořenovým certifikátem a podepsaným certifikátem. Skript `cacerts.bat` exportuje požadavky na certifikáty klienta a serveru ze svých úložišť klíčů. Skript podepíše požadavky na certifikáty s klíčem soukromé certifikační autority v úložišti klíčů `cajkskeystore.jks` a poté naimportuje podepsané certifikáty zpět do stejných úložišť klíčů, ze kterých přišly požadavky. Import vytvoří řetěz certifikátů s kořenovým certifikátem CA. Skript vytvoří úložiště údajů o důvěryhodnosti klienta ve formátu PEM z úložiště údajů o důvěryhodnosti JKS klienta.

```
@rem
@echo ____________________________________________________________________________
@echo Export self-signed certificates: %cacert%
@rem
@rem Export CA public certificate
%keytool% -exportcert -noprompt -rfc -alias %caalias%  -keystore %cajkskeystore% -storepass
%cajkskeystorepass%  -file %cacert%
```

```
@rem
@echo ____________________________________________________________________________
@echo Add CA to server key and client key and trust stores: %srvjkskeystore%,
%cltjkskeystore%, %cltcajkstruststore%,
@rem The CA certificate is necessary to create key chains in the client and server key
stores,
@rem and to certify key chains in the server key store and the client trust store
@rem
@rem Import the CA root certificate into the server key store
%keytool% -import -noprompt -alias %caalias% -file %cacert% -keystore %srvjkskeystore%
-storepass %srvjkskeystorepass%
@rem Import the CA root certificate into the client key store
@rem Omit this step, unless you are authenticating clients.
%keytool% -import -noprompt -alias %caalias% -file %cacert% -keystore %cltjkskeystore%
-storepass %cltjkskeystorepass%
@rem Import the CA root certificate into the client ca-trust store (for ca chained
authentication)
%keytool% -import -noprompt -alias %caalias% -file %cacert% -keystore %cltcajkstruststore%
-storepass %cltcajkstruststorepass%
```

```
@rem
@echo ____________________________________________________________________________
@echo Create certificate signing requests: %srvcertreq% and %cltcertreq%
@rem
@rem Create a certificate signing request (CSR) for the server key
%keytool% -certreq -alias %srvalias% -file %srvcertreq% -keypass %srvkeypass% -keystore
%srvjkskeystore% -storepass %srvjkskeystorepass%
@rem Create a certificate signing request (CSR) for the client key
%keytool% -certreq -alias %cltalias% -file %cltcertreq% -keypass %cltkeypass% -keystore
%cltjkskeystore% -storepass %cltjkskeystorepass%
```

```
@rem
@echo ____________________________________________________________________________
@echo Sign certificate requests: %srvcertcasigned% and %cltcertcasigned%
@rem The requests are signed with the ca key in the cajkskeystore.jks keystore
@rem
@rem Sign server certificate request
%keytool% -gencert -infile %srvcertreq% -outfile %srvcertcasigned% -alias %caalias%
-keystore %cajkskeystore% -storepass %cajkskeystorepass% -keypass %cakeypass%
@rem Sign client certificate request
@rem Omit this step, unless you are authenticating clients.
```

```
%keytool% -gencert -infile %cltcertreq% -outfile %cltcertcasigned% -alias %caalias%
-keystore %cajkskeystore% -storepass %cajkskeystorepass% -keypass %cakeypass%
```

```
@rem
@echo ________________________________________________________________________________
@echo Import the signed certificates back into the key stores to create the key chain:
%srvjkskeystore% and %cltjkskeystore%
@rem
@rem Import the signed server certificate
%keytool% -import -noprompt -alias %srvalias% -file %srvcertcasigned% -keypass %srvkeypass%
-keystore %srvjkskeystore% -storepass %srvjkskeystorepass%
@rem Import the signed client certificate and key chain back into the client keystore
%keytool% -import -noprompt -alias %cltalias% -file %cltcertcasigned% -keypass %cltkeypass%
-keystore %cltjkskeystore% -storepass %cltjkskeystorepass%
@rem
@rem The CA certificate is needed in the server key store, and the client trust store
@rem to verify the key chain sent from the client or server
@echo Delete the CA certificate from %cltjkskeystore%: it causes a problem in converting
keystore to pem
@rem Omit this step, unless you are authenticating clients.
%keytool% -delete -alias %caalias% -keystore %cltjkskeystore% -storepass
%cltjkskeystorepass%
```

```
@rem
@echo ________________________________________________________________________________
@echo Create a pem client-ca trust store from the jks client-ca trust store:
%cltcapemtruststore%
@rem Omit this step, unless you are configuring a C or iOS client.
@rem
%keytool% -importkeystore -noprompt -srckeystore %cltcajkstruststore%  -destkeystore
%cltcap12truststore%  -srcstoretype jks -deststoretype pkcs12 -srcstorepass
%cltcajkstruststorepass%  -deststorepass %cltcap12truststorepass%
%openssl%\bin\openssl pkcs12 -in %cltcap12truststore%  -out %cltcapemtruststore%  -passin
pass:%cltcap12truststorepass%  -passout pass:%cltpemtruststorepass%
```

```
@rem
@echo ________________________________________________________________________________
@echo Create a pem client key store from the jks client keystore
@rem Omit this step, unless you are configuring a C or iOS client.
@rem
%keytool% -importkeystore -noprompt -srckeystore %cltjkskeystore%      -destkeystore
%cltp12keystore%     -srcstoretype jks -deststoretype pkcs12 -srcstorepass
%cltjkskeystorepass%     -deststorepass %cltp12keystorepass%
%openssl%\bin\openssl pkcs12 -in %cltp12keystore%     -out %cltpemkeystore%      -passin
pass:%cltp12keystorepass% -passout pass:%cltpemkeystorepass%
```

mqcerts.bat

Skript uvádí úložiště klíčů a certifikáty v adresáři certifikátů. Poté vytvoří ukázkového správce front MQTT a nakonfiguruje zabezpečené kanály telemetrie.

```
@echo ________________________________________________________________________________
@echo List keystores and certificates
dir %certpath%\*.* /b
```

```
@rem
@echo Create queue manager and define mqtt channels and certificate stores
call "%MQ_FILE_PATH%\mqxr\Samples\SampleMQM" >> %mqlog%
echo DEFINE CHANNEL(%chlreq%)      CHLTYPE(MQTT) TRPTYPE(TCP) PORT(%portsslreq%)
SSLCAUTH(%authreq%) SSLKEYR('%srvjkskeystore%') SSLKEYP('%srvjkskeystorepass%')
MCAUSER(%mcauser%) | runmqsc %qm% >> %mqlog%
echo DEFINE CHANNEL(%chlopt%)      CHLTYPE(MQTT) TRPTYPE(TCP) PORT(%portsslopt%)
SSLCAUTH(%authopt%) SSLKEYR('%srvjkskeystore%') SSLKEYP('%srvjkskeystorepass%')
MCAUSER(%mcauser%) | runmqsc %qm% >> %mqlog%
V7.5.0.1
echo DEFINE CHANNEL(%chlsslreqws%) CHLTYPE(MQTT) TRPTYPE(TCP) PORT(%portsslreqws%)
SSLCAUTH(%authreq%) SSLKEYR('%srvjkskeystore%') SSLKEYP('%srvjkskeystorepass%')
MCAUSER(%mcauser%) PROTOCOL(HTTP) | runmqsc %qm% >> %mqlog%
echo DEFINE CHANNEL(%chlssloptws%) CHLTYPE(MQTT) TRPTYPE(TCP) PORT(%portssloptws%)
SSLCAUTH(%authopt%) SSLKEYR('%srvjkskeystore%') SSLKEYP('%srvjkskeystorepass%')
MCAUSER(%mcauser%) PROTOCOL(HTTP) | runmqsc %qm% >> %mqlog%
echo DEFINE CHANNEL(%chlws%)       CHLTYPE(MQTT) TRPTYPE(TCP) PORT(%portws%)
MCAUSER(%mcauser%) PROTOCOL(HTTP) | runmqsc %qm% >> %mqlog%
@echo MQ logs saved in %mqlog%echo
```

Sestavení a spuštění zabezpečeného produktu Ukázková aplikace C klienta MQTT

Na základě příkladu Windows můžete začít pracovat se zabezpečeným vzorovým aplikací C na libovolném operačním systému, pro který můžete kompilovat zdroj C. Ověřte, že můžete spustit ukázkovou aplikaci C na serveru buď IBM MessageSight , nebo IBM WebSphere MQ jako server MQTT.

Než začnete

1. Musíte mít přístup k serveru MQTT version 3.1 , který podporuje protokol MQTT protocol přes SSL.
2. Je-li mezi vaším klientem a serverem brána firewall, zkontrolujte, zda neblokuje provoz produktu MQTT .
3. Binární verze klienta pro knihovny C jsou poskytnuty pro řadu operačních systémů. Pro některé z těchto operačních systémů není zabezpečená verze klienta poskytována jako binární soubor. V případě těchto operačních systémů musíte postupovat podle pokynů v příručce “Sestavení klienta MQTT pro knihovny C” na stránce 30.
4. V případě řešení problémů může podpora IBM vyžadovat spuštění klienta produktu MQTT pro platformu C na referenční platformě.
5. Kanály SSL musí být spuštěny.

Přehled podporovaných a referenčních platforem viz Systémové požadavky pro IBM Mobile Messaging and M2M Client Pack. Podrobnosti o tom, co je podporováno pro klienta jazyka C, naleznete v příslušných oddílech Systémové požadavky pro produkt WebSphere MQ V7.5 Telemetry.

Informace o této úloze

Tento článek vám názorně ukazuje, jak kompilovat a spustit zabezpečení Ukázková aplikace C klienta MQTT v systému Windows z příkazového řádku. Na obrázku je produkt Microsoft Visual Studio 2010 použit ke kompilaci klienta. Skripty příkazového řádku můžete upravit ke kompilaci a spuštění ukázkové aplikace v jiných operačních systémech, jako je například Linux a iOS.

Poznámka:

Skripty produktu Windows uvedené v tomto článku předpokládají, že sestavujete celý balík OpenSSL ze zdroje. Rozhodnete-li se použít předkompilované knihovny, které poskytuje společnost IBM , můžete raději získat předkompilované binární vydání OpenSSL. Předkompilované knihovny nejsou k dispozici pro použití s produktem iOS.

Zabezpečte kanál SSL buď pomocí klíčů podepsanými certifikační autoritou, nebo pomocí klíčů podepsaným držitelem.

Postup

1. Zvolte server MQTT , ke kterému se můžete připojit k aplikaci klienta.

   Server musí podporovat protokol MQTT version 3.1 přes SSL. Všechny servery MQTT z IBM to dělají, včetně IBM WebSphere MQ a IBM MessageSight. Viz “Začínáme se servery MQTT” na stránce 135.
2. Volitelné: Nainstalujte vývojovou sadu produktu Java (JDK) ve verzi 7 nebo novější.

   verze 7 je vyžadován ke spuštění příkazu **keytool** pro certifikaci certifikátů. Pokud certifikáty certifikovat nechcete, není třeba mít sadu JDK verze 7.
3. Nainstalujte vývojové prostředí C na platformě, na které sestavujete.

   Soubory Makefile v příkladech v tomto tématu se zaměřují na následující nástroje:

   - iOS Pro iOS, v Apple Mac s OS X 10.8.2 s vývojovými nástroji iOS z Xcode.
   - Linux Pro Linux, gcc verze 4.4.6 z Red Hat Enterprise Linux verze 6.2.

     Minimální podporovaná úroveň knihovny `glibc` C je 2.12a jádro Linux je 2.6.32.

- Windows Pro Microsoft Windows, Visual Studio verze 10.0.

4. Stáhněte si produkt Mobile Messaging and M2M Client Pack a nainstalujte sadu SDK produktu MQTT .

   K dispozici není žádný instalační program, stačí jen rozbalit stažený soubor.

   a. Stáhněte si Mobile Messaging and M2M Client Pack.

   b. Vytvořte složku, do které budete instalovat sadu SDK.

      Je možné, že budete chtít pojmenovat složku MQTT. Cesta k této služce je zde označována jako *sdkroot*.

   c. Rozbalte komprimovaný obsah souboru Mobile Messaging and M2M Client Pack do *sdkroot*. Rozšíření vytváří adresářový strom, který začíná v *sdkroot*\SDK.

5. Volitelné: Postupujte podle kroků uvedených v tématu “Sestavení klienta MQTT pro knihovny C” na stránce 30.

   Tento krok proveďte pouze v případě, že sada MQTT SDK neobsahuje zabezpečenou knihovnu klienta jazyka C pro váš cílový operační systém.

   - Windows Knihovny jsou `mqttv3cs.lib` pro kompilaci a `mqttv3cs.dll` pro spuštění.
   - Linux Knihovna je `libmqttv3cs.so`
   - iOS Knihovna je `libmqttv3cs.a`

6. Vytvořte a spusťte skripty pro generování dvojic klíčů a certifikátů a nakonfigurujte produkt IBM WebSphere MQ jako server MQTT .

   Postupujte podle kroků v části “Generování klíčů a certifikátů” na stránce 95 a vytvořte a spusťte skripty. Tyto skripty jsou také vypsány v “Ukázkové skripty pro konfiguraci certifikátů SSL pro produkt Windows” na stránce 89.

7. Zkontrolujte, zda jsou kanály zabezpečení SSL spuštěny a nastaveny dle očekávání.

   V IBM WebSphere MQ zadejte následující příkaz do okna příkazového řádku:

   - Linux

     ```
     echo 'DISPLAY CHSTATUS(SSL*) CHLTYPE(MQTT) ALL' | runmqsc MQXR_SAMPLE_QM
     echo 'DISPLAY CHANNEL(SSL*) CHLTYPE(MQTT) ALL' | runmqsc MQXR_SAMPLE_QM
     ```

   - Windows

     ```
     echo DISPLAY CHSTATUS(SSL*) CHLTYPE(MQTT) ALL | runmqsc MQXR_SAMPLE_QM
     echo DISPLAY CHANNEL(SSL*) CHLTYPE(MQTT) ALL | runmqsc MQXR_SAMPLE_QM
     ```

8. Vytvořte skripty pro sestavení a spuštění zabezpečeného produktu Ukázková aplikace C klienta MQTT.

   a) Vytvořte a spusťte `sscclient.bat` , abyste otestoval kanál SSL, který je zabezpečen s certifikáty s vlastním podpisem.

   b) Vytvořte a spusťte `cacclient.bat` , abyste otestoval kanál SSL, který je zabezpečen pomocí certifikátů podepsaných certifikační autoritou.

Výsledky

Výsledky jsou podobné spuštění nezabezpečeného klienta.

```
Connected to ssl://localhost:8884
Subscribing to topic "MQTTV3SSample/#" qos 2
Topic:          MQTTV3SSample/C/v3
Message:        Message from MQTTv3 SSL C client
QoS:            2
Connected to ssl://localhost:8885
Subscribing to topic "MQTTV3SSample/#" qos 2
Topic:          MQTTV3SSample/C/v3
Message:        Message from MQTTv3 SSL C client
QoS:            2
```

Obrázek 16. Zabezpečený odběratel

```
Setting environment for using Microsoft Visual Studio 2010 x86 tools.
MQTTV3SSample.c
Connected to ssl://localhost:8884
Publishing to topic "MQTTV3SSample/C/v3" qos 2
Disconnected
Press any key to continue . . .
Connected to ssl://localhost:8885
Publishing to topic "MQTTV3SSample/C/v3" qos 2
Disconnected
Press any key to continue . . .
```

Obrázek 17. Zabezpečený vydavatel

Skripty pro spuštění zabezpečeného produktu Ukázková aplikace C klienta MQTT

Spusťte skripty v produktu “Ukázkové skripty pro konfiguraci certifikátů SSL pro produkt Windows” na stránce 89 před spuštěním těchto skriptů.

Zabezpečte Ukázková aplikace C klienta MQTT certifikáty s vlastním podpisem.
Spusťte tento skript s certifikáty s vlastním podpisem, které jste vytvořili spuštěním skriptu `sscerts.bat` .

```
@echo off
setlocal
cd %csamppath%
erase MQTTV3SSample.obj
erase MQTTV3SSample.exe
call "C:\Program Files\Microsoft Visual Studio 10.0\VC\vcvarsall.bat" x86
cl  /nologo /D "WIN32" /I "..\include" "MQTTV3SSample.c" /link /
nologo ..\windows_ia32\mqttv3cs.lib
set path=%path%;%csamppath%\..\windows_ia32
ping -n 2 127.0.0.1 > NUL 2>&1
@echo start "MQTT Subscriber" MQTTV3SSample -a subscribe -b %host% -p %sslportopt% -k
%cltpemkeystore% -w %cltpemkeystorepass% -r %cltsrvpemtruststore% -v 1
start "MQTT Subscriber" MQTTV3SSample -a subscribe -b %host% -p %sslportopt% -k
%cltpemkeystore% -w %cltpemkeystorepass% -r %cltsrvpemtruststore% -v 1
@rem Sleep for 2 seconds
ping -n 2 127.0.0.1 > NUL 2>&1
@echo MQTTV3SSample -b %host% -p %sslportopt% -k %cltpemkeystore% -w %cltpemkeystorepass%
-r %cltsrvpemtruststore% -v 1
MQTTV3SSample -b %host% -p %sslportopt% -k %cltpemkeystore% -w %cltpemkeystorepass% -r
%cltsrvpemtruststore% -v 1
pause
ping -n 2 127.0.0.1 > NUL 2>&1
@echo start "MQTT Subscriber" MQTTV3SSample -a subscribe -b %host% -p %sslportreq% -k
%cltpemkeystore% -w %cltpemkeystorepass% -r %cltsrvpemtruststore% -v 1
start "MQTT Subscriber" MQTTV3SSample -a subscribe -b %host% -p %sslportreq% -k
%cltpemkeystore% -w %cltpemkeystorepass% -r %cltsrvpemtruststore% -v 1
@rem Sleep for 2 seconds
ping -n 2 127.0.0.1 > NUL 2>&1
@echo MQTTV3SSample -b %host% -p %sslportreq% -k %cltpemkeystore% -w %cltpemkeystorepass%
-r %cltsrvpemtruststore% -v 1
MQTTV3SSample -b %host% -p %sslportreq% -k %cltpemkeystore% -w %cltpemkeystorepass% -r
%cltsrvpemtruststore% -v 1
pause
ping -n 2 127.0.0.1 > NUL 2>&1
endlocal
```

Obrázek 18. `sscclient.bat`

Spusťte ukázku klienta jazyka C zabezpečeného klienta produktu MQTT s certifikáty podepsanými certifikační autoritou.

Spusťte tento skript spolu s certifikáty podepsanými certifikační autoritou, které jste vytvořili spuštěním skriptu `cacerts.bat` .

```
@echo off
setlocal
cd %csamppath%
erase MQTTV3SSample.obj
erase MQTTV3SSample.exe
call "C:\Program Files\Microsoft Visual Studio 10.0\VC\vcvarsall.bat" x86
cl  /nologo /D "WIN32" /I "..\include" "MQTTV3SSample.c" /link /
nologo ..\windows_ia32\mqttv3cs.lib
set path=%path%;%csamppath%\..\windows_ia32
ping -n 2 127.0.0.1 > NUL 2>&1
@echo start "MQTT Subscriber" MQTTV3SSample -a subscribe -b %host% -p %sslportopt% -k
%cltpemkeystore% -w %cltpemkeystorepass% -r %cltsrvpemtruststore% -v 1
start "MQTT Subscriber" MQTTV3SSample -a subscribe -b %host% -p %sslportopt% -k
%cltpemkeystore% -w %cltpemkeystorepass% -r %cltsrvpemtruststore% -v 1
@rem Sleep for 2 seconds
ping -n 2 127.0.0.1 > NUL 2>&1
@echo MQTTV3SSample -b %host% -p %sslportopt% -k %cltpemkeystore% -w %cltpemkeystorepass%
-r %cltsrvpemtruststore% -v 1
MQTTV3SSample -b %host% -p %sslportopt% -k %cltpemkeystore% -w %cltpemkeystorepass% -r
%cltsrvpemtruststore% -v 1
pause
ping -n 2 127.0.0.1 > NUL 2>&1
@echo start "MQTT Subscriber" MQTTV3SSample -a subscribe -b %host% -p %sslportreq% -k
%cltpemkeystore% -w %cltpemkeystorepass% -r %cltsrvpemtruststore% -v 1
start "MQTT Subscriber" MQTTV3SSample -a subscribe -b %host% -p %sslportreq% -k
%cltpemkeystore% -w %cltpemkeystorepass% -r %cltsrvpemtruststore% -v 1
@rem Sleep for 2 seconds
ping -n 2 127.0.0.1 > NUL 2>&1
@echo MQTTV3SSample -b %host% -p %sslportreq% -k %cltpemkeystore% -w %cltpemkeystorepass%
-r %cltsrvpemtruststore% -v 1
MQTTV3SSample -b %host% -p %sslportreq% -k %cltpemkeystore% -w %cltpemkeystorepass% -r
%cltsrvpemtruststore% -v 1
pause
ping -n 2 127.0.0.1 > NUL 2>&1
endlocal
```

Obrázek 19. `cacclient.bat`

Související pojmy

“Zabezpečení produktu MQTT” na stránce 51

Tři koncepty jsou základní pro zabezpečení produktu MQTT : identita, ověřování a autorizace. Identita je o pojmenování klienta, který je autorizován a kterému je uděleno oprávnění. Ověřování je o prokazování identity klienta a autorizace se týká správy práv, která jsou klientovi udělena.

Související úlohy

“Generování klíčů a certifikátů” na stránce 95

Postupujte podle této procedury, chcete-li generovat klíče a certifikáty pro klienty Java a C, včetně aplikací Android a iOS , a serverů IBM WebSphere MQ a IBM MessageSight .

Ukázkové skripty pro konfiguraci certifikátů SSL pro produkt Windows

Příklad

Ukázkové příkazové soubory vytvoří certifikáty a úložiště certifikátů, jak je popsáno v krocích v úloze. Kromě toho tento příklad nastaví správce front klienta MQTT pro použití úložiště certifikátů serveru. Příklad odstraní a znovu vytvoří správce front voláním skriptu `SampleMQM.bat`, který je poskytnut s IBM WebSphere MQ.

`initcert.bat`

`initcert.bat` nastavuje názvy a cesty k certifikátům a jiným parametrům, které jsou vyžadovány příkazy **keytool** a **openSSL**. Nastavení jsou popsána v komentářích ve skriptu.

```
@echo off
@rem Set the path where you installed the MQTT SDK
@rem and short cuts to the samples directories.
set SDKRoot=C:\MQTT
set jsamppath=%SDKRoot%\sdk\clients\java\samples
set csamppath=%SDKRoot%\sdk\clients\c\samples
```

```
@rem Set the paths to Version 7 of the JDK
@rem and to the directory where you built the openSSL package.
@rem Set short cuts to the tools.
set javapath=C:\Program Files\IBM\Java70
set keytool="%javapath%\jre\bin\keytool.exe"
set ikeyman="%javapath%\jre\bin\ikeyman.exe"
set openssl=%SDKRoot%\openSSL
set runopenssl="%openssl%\bin\openssl"
```

```
@rem Set the path to where certificates are to be stored,
@rem and set global security parameters.
@rem Omit set password, and the security tools prompt you for passwords.
@rem Validity is the expiry time of the certificates in days.
set certpath=%SDKRoot%\Certificates
set password=password
set validity=5000
set algorithm=RSA
```

```
@rem Set the certificate authority (CA) jks keystore and certificate parameters.
@rem Omit this step, unless you are defining your own certificate authority.
@rem The CA keystore contains the key-pair for your own certificate authority.
@rem You must protect the CA keystore.
@rem The CA certificate is the self-signed certificate authority public certificate.
@rem It is commonly known as the CA root certificate.
set caalias=caalias
set cadname="CN=mqttca.ibm.id.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
set cakeypass=%password%
@rem ca key store
set cajkskeystore=%certpath%\cakeystore.jks
set cajkskeystorepass=%password%
@rem ca certificate (root certificate)
set cacert=%certpath%\cacert.cer
```

```
@rem Set the server jks keystore and certificate parameters.
@rem The server keystore contains the key-pair for the server.
@rem You must protect the server keystore.
@rem If you then export the server certificate it is self-signed.
@rem Alternatively, if you export a certificate signing request (CSR)
@rem from the server keystore for the server key,
@rem and import the signed certificate back into the same keystore,
@rem it forms a certificate chain.
@rem The certificate chain links the server certificate to the CA.
@rem When you now export the server certificate,
@rem the exported certificate includes the certificate chain.
set srvalias=srvalias
set srvdname="CN=mqttserver.ibm.id.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
set srvkeypass=%password%
@rem server key stores
set srvjkskeystore=%certpath%\srvkeystore.jks
set srvjkskeystorepass=%password%
@rem server certificates
set srvcertreq=%certpath%\srvcertreq.csr
set srvcertcasigned=%certpath%\srvcertcasigned.cer
set srvcertselfsigned=%certpath%\srvcertselfsigned.cer
```

```
@rem Set the client jks keystore and certificate parameters
@rem Omit this step, unless you are authenticating clients.
@rem The client keystore contains the key-pair for the client.
@rem You must protect the client keystore.
@rem If you then export the client certificate it is self-signed.
@rem Alternatively, if you export a certificate signing request (CSR)
@rem from the client keystore for the client key,
@rem and import the signed certificate back into the same keystore,
@rem it forms a certificate chain.
@rem The certificate chain links the client certificate to the CA.
@rem When you now export the client certificate,
@rem the exported certificate includes the certificate chain.
set cltalias=cltalias
set cltdname="CN=mqttclient.ibm.id.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
set cltkeypass=%password%
@rem client key stores
set cltjkskeystore=%certpath%\cltkeystore.jks
set cltjkskeystorepass=%password%
set cltcertreq=%certpath%\cltcertreq.csr
set cltcertcasigned=%certpath%\cltcacertsigned.cer
set cltcertselfsigned=%certpath%\cltcertselfsigned.cer
```

```
@rem Set the paths to the client truststores signed by CA and signed by server key.
@rem You only need to define one of the trust stores.
@rem A trust store holds certificates that you trust,
@rem which are used to authenticate untrusted certificates.
@rem In this example, when the client authenticates the MQTT server it connects to,
@rem it authenticates the certificate it is sent by the server
@rem with the certificates in its trust store.
@rem For example, the MQTT server sends its server certificate,
@rem and the client authenticates it with either the same server certificate
@rem that you have stored in the cltsrvtruststore.jks trust store,
@rem or against the CA certificate, if the server certificate is signed by the CA.
set cltcajkstruststore=%certpath%\cltcatruststore.jks
set cltcajkstruststorepass=%password%
set cltsrvjkstruststore=%certpath%\cltsrvtruststore.jks
set cltsrvjkstruststorepass=%password%
```

```
@rem Set the paths to the client PKCS12 and PEM key and trust stores.
@rem Omit this step, unless you are configuring a C or iOS client.
@rem You only need to define either one of the trust stores for storing CA
@rem or server signed server certificates.
set cltp12keystore=%certpath%\cltkeystore.p12
set cltp12keystorepass=%password%
set cltpemkeystore=%certpath%\cltkeystore.pem
set cltpemkeystorepass=%password%
set cltcap12truststore=%certpath%\cltcatruststore.p12
set cltcap12truststorepass=%password%
set cltcapemtruststore=%certpath%\cltcatruststore.pem
set cltcapemtruststorepass=%password%
set cltsrvp12truststore=%certpath%\cltsrvtruststore.p12
set cltsrvp12truststorepass=%password%
set cltsrvpemtruststore=%certpath%\cltsrvtruststore.pem
set cltsrvpemtruststorepass=%password%
```

```
@rem set WMQ Variables
set authopt=NEVER
set authreq=REQUIRED
set qm=MQXR_SAMPLE_QM
set host=localhost
set mcauser='Guest'
set portsslopt=8884
set chlopt=SSLOPT
set portsslreq=8885
set chlreq=SSLREQ
V 7.5.0.1 set portws=1886
set chlws=PLAINWS
set chlssloptws=SSLOPTWS
set portssloptws=8886
set chlsslreqws=SSLREQWS
set portsslreqws=8887
set mqlog=%certpath%\wmq.log
```

cleancert.bat

Příkazy ve skriptu `cleancert.bat` odstraní správce front klienta MQTT, aby se zajistilo, že úložiště certifikátů serveru není uzamčeno, a potom odstraní všechna úložiště klíčů a certifikáty, které jsou vytvořeny ukázkovými skripty zabezpečení.

```
@rem Delete the MQTT sample queue manager, MQXR_SAMPLE_QM
call "%MQ_FILE_PATH%\bin\setmqenv" -s
endmqm -i %qm%
dltmqm %qm%
```

```
@rem Erase all the certificates and key stores created by the sample scripts.
erase %cajkskeystore%
erase %cacert%
erase %srvjkskeystore%
erase %srvcertreq%
erase %srvcertcasigned%
erase %srvcertselfsigned%
erase %cltjkskeystore%
erase %cltp12keystore%
erase %cltpemkeystore%
erase %cltcertreq%
erase %cltcertcasigned%
erase %cltcertselfsigned%
erase %cltcajkstruststore%
```

```
erase %cltcap12truststore%
erase %cltcapemtruststore%
erase %cltsrvjkstruststore%
erase %cltsrvp12truststore%
erase %cltsrvpemtruststore%
erase %mqlog%
@echo Cleared all certificates
dir %certpath%\*.* /b
```

genkeys.bat

Příkazy ve skriptu `genkeys.bat` vytvářejí dvojice klíčů pro soukromé certifikační autority, server a klienta.

```
@rem
@echo ____________________________________________________________________________
@echo Generate %caalias%, %srvalias%, and %cltalias% key-pairs in %cajkskeystore%,
%srvjkskeystore%, and %cltjkskeystore%
@rem
@rem -- Generate a client certificate and a private key pair
@rem Omit this step, unless you are authenticating clients.
%keytool% -genkeypair -noprompt -alias %cltalias% -dname %cltdname% -keystore
%cltjkskeystore% -storepass %cltjkskeystorepass% -keypass %cltkeypass% -keyalg %algorithm%
-validity %validity%
```

```
@rem -- Generate a server certificate and private key pair
%keytool% -genkeypair -noprompt -alias %srvalias% -dname %srvdname% -keystore
%srvjkskeystore% -storepass %srvjkskeystorepass% -keypass %srvkeypass% -keyalg %algorithm%
-validity %validity%
```

```
@rem Create CA, client and server key-pairs
@rem  -- Generate a CA certificate and private key pair - The extension asserts this is a
certificate authority certificate, which is required to import into firefox
%keytool% -genkeypair -noprompt -ext bc=ca:true -alias %caalias%  -dname %cadname%
-keystore %cajkskeystore% -storepass %cajkskeystorepass%  -keypass %cakeypass% -keyalg
%algorithm% -validity %validity%
```

sscerts.bat

Příkazy ve skriptu `sscerts.bat` exportují certifikáty podepsané držitelem klienta a serveru z vlastních úložišť klíčů a importují certifikát serveru do úložiště údajů o důvěryhodnosti klienta a certifikát klienta do úložiště klíčů serveru. Server nemá úložiště údajů o důvěryhodnosti. Příkazy vytvoří úložiště údajů o důvěryhodnosti klienta ve formátu PEM z úložiště údajů o důvěryhodnosti JKS klienta.

```
@rem
@echo ____________________________________________________________________________
@echo Export self-signed certificates:  %srvcertselfsigned% and %cltcertselfsigned%
@rem Export Server public certificate
%keytool% -exportcert -noprompt -rfc -alias %srvalias% -keystore %srvjkskeystore%
-storepass %srvjkskeystorepass% -file %srvcertselfsigned%
@rem Export Client public certificate
@rem Omit this step, unless you are authenticating clients.
%keytool% -exportcert -noprompt -rfc -alias %cltalias% -keystore %cltjkskeystore%
-storepass %cltjkskeystorepass% -file %cltcertselfsigned%
```

```
@rem
@echo ____________________________________________________________________________
@echo Add selfsigned server certificate %srvcertselfsigned% to client trust store:
%cltsrvjkstruststore%
@rem Import the server certificate into the client-server trust store (for server self-
signed authentication)
%keytool% -import -noprompt -alias %srvalias% -file %srvcertselfsigned% -keystore
%cltsrvjkstruststore% -storepass %cltsrvjkstruststorepass%
```

```
@rem
@echo ____________________________________________________________________________
@echo Add selfsigned client certificate %cltcertselfsigned% to server trust store:
%srvjkskeystore%
@rem Import the client certificate into the server trust store (for client self-signed
authentication)
@rem Omit this step, unless you are authenticating clients.
%keytool% -import -noprompt -alias %cltalias% -file %cltcertselfsigned% -keystore
```

```
%srvjkskeystore% -storepass %srvjkskeystorepass%
```

```
@rem
@echo ------------------------------------------------------------------------------
@echo Create a pem client-server trust store from the jks client-server trust store:
%cltsrvpemtruststore%
@rem
%keytool% -importkeystore -noprompt -srckeystore %cltsrvjkstruststore% -destkeystore
%cltsrvp12truststore% -srcstoretype jks -deststoretype pkcs12 -srcstorepass
%cltsrvjkstruststorepass% -deststorepass %cltsrvp12truststorepass%
%openssl%\bin\openssl pkcs12 -in %cltsrvp12truststore% -out %cltsrvpemtruststore% -passin
pass:%cltsrvp12truststorepass% -passout pass:%cltsrvpemtruststorepass%@rem
@rem
@echo ------------------------------------------------------------------------------
@echo Create a pem client key store from the jks client keystore
@rem Omit this step, unless you are configuring a C or iOS client.
@rem
%keytool% -importkeystore -noprompt -srckeystore %cltjkskeystore%      -destkeystore
%cltp12keystore%     -srcstoretype jks -deststoretype pkcs12 -srcstorepass
%cltjkskeystorepass%     -deststorepass %cltp12keystorepass%
%openssl%\bin\openssl pkcs12 -in %cltp12keystore%     -out %cltpemkeystore%      -passin
pass:%cltp12keystorepass% -passout pass:%cltpemkeystorepass%
```

cacerts.bat

Skript importuje kořenový certifikát certifikační autority soukromých úložišť klíčů. Kořenový certifikát CA je potřebný k vytvoření svazku klíčů mezi kořenovým certifikátem a podepsaným certifikátem. Skript `cacerts.bat` exportuje požadavky na certifikáty klienta a serveru ze svých úložišť klíčů. Skript podepíše požadavky na certifikáty s klíčem soukromé certifikační autority v úložišti klíčů `cajkskeystore.jks` a poté naimportuje podepsané certifikáty zpět do stejných úložišť klíčů, ze kterých přišly požadavky. Import vytvoří řetěz certifikátů s kořenovým certifikátem CA. Skript vytvoří úložiště údajů o důvěryhodnosti klienta ve formátu PEM z úložiště údajů o důvěryhodnosti JKS klienta.

```
@rem
@echo ------------------------------------------------------------------------------
@echo Export self-signed certificates: %cacert%
@rem
@rem Export CA public certificate
%keytool% -exportcert -noprompt -rfc -alias %caalias%  -keystore %cajkskeystore% -storepass
%cajkskeystorepass%  -file %cacert%
```

```
@rem
@echo ------------------------------------------------------------------------------
@echo Add CA to server key and client key and trust stores: %srvjkskeystore%,
%cltjkskeystore%, %cltcajkstruststore%,
@rem The CA certificate is necessary to create key chains in the client and server key
stores,
@rem and to certify key chains in the server key store and the client trust store
@rem
@rem Import the CA root certificate into the server key store
%keytool% -import -noprompt -alias %caalias% -file %cacert% -keystore %srvjkskeystore%
-storepass %srvjkskeystorepass%
@rem Import the CA root certificate into the client key store
@rem Omit this step, unless you are authenticating clients.
%keytool% -import -noprompt -alias %caalias% -file %cacert% -keystore %cltjkskeystore%
-storepass %cltjkskeystorepass%
@rem Import the CA root certificate into the client ca-trust store (for ca chained
authentication)
%keytool% -import -noprompt -alias %caalias% -file %cacert% -keystore %cltcajkstruststore%
-storepass %cltcajkstruststorepass%
```

```
@rem
@echo ------------------------------------------------------------------------------
@echo Create certificate signing requests: %srvcertreq% and %cltcertreq%
@rem
@rem Create a certificate signing request (CSR) for the server key
%keytool% -certreq -alias %srvalias% -file %srvcertreq% -keypass %srvkeypass% -keystore
%srvjkskeystore% -storepass %srvjkskeystorepass%
@rem Create a certificate signing request (CSR) for the client key
%keytool% -certreq -alias %cltalias% -file %cltcertreq% -keypass %cltkeypass% -keystore
%cltjkskeystore% -storepass %cltjkskeystorepass%
```

```
@rem
@echo ------------------------------------------------------------------------------
```

```
@echo Sign certificate requests: %srvcertcasigned% and %cltcertcasigned%
@rem The requests are signed with the ca key in the cajkskeystore.jks keystore
@rem
@rem Sign server certificate request
%keytool% -gencert -infile %srvcertreq% -outfile %srvcertcasigned% -alias %caalias%
-keystore %cajkskeystore% -storepass %cajkskeystorepass% -keypass %cakeypass%
@rem Sign client certificate request
@rem Omit this step, unless you are authenticating clients.
%keytool% -gencert -infile %cltcertreq% -outfile %cltcertcasigned% -alias %caalias%
-keystore %cajkskeystore% -storepass %cajkskeystorepass% -keypass %cakeypass%
```

```
@rem
@echo ____________________________________________________________________________
@echo Import the signed certificates back into the key stores to create the key chain:
%srvjkskeystore% and %cltjkskeystore%
@rem
@rem Import the signed server certificate
%keytool% -import -noprompt -alias %srvalias% -file %srvcertcasigned% -keypass %srvkeypass%
-keystore %srvjkskeystore% -storepass %srvjkskeystorepass%
@rem Import the signed client certificate and key chain back into the client keystore
%keytool% -import -noprompt -alias %cltalias% -file %cltcertcasigned% -keypass %cltkeypass%
-keystore %cltjkskeystore% -storepass %cltjkskeystorepass%
@rem
@rem The CA certificate is needed in the server key store, and the client trust store
@rem to verify the key chain sent from the client or server
@echo Delete the CA certificate from %cltjkskeystore%: it causes a problem in converting
keystore to pem
@rem Omit this step, unless you are authenticating clients.
%keytool% -delete -alias %caalias% -keystore %cltjkskeystore% -storepass
%cltjkskeystorepass%
```

```
@rem
@echo ____________________________________________________________________________
@echo Create a pem client-ca trust store from the jks client-ca trust store:
%cltcapemtruststore%
@rem Omit this step, unless you are configuring a C or iOS client.
@rem
%keytool% -importkeystore -noprompt -srckeystore %cltcajkstruststore%  -destkeystore
%cltcap12truststore%  -srcstoretype jks -deststoretype pkcs12 -srcstorepass
%cltcajkstruststorepass%  -deststorepass %cltcap12truststorepass%
%openssl%\bin\openssl pkcs12 -in %cltcap12truststore%  -out %cltcapemtruststore%  -passin
pass:%cltcap12truststorepass%  -passout pass:%cltpemtruststorepass%
```

```
@rem
@echo ____________________________________________________________________________
@echo Create a pem client key store from the jks client keystore
@rem Omit this step, unless you are configuring a C or iOS client.
@rem
%keytool% -importkeystore -noprompt -srckeystore %cltjkskeystore%      -destkeystore
%cltp12keystore%     -srcstoretype jks -deststoretype pkcs12 -srcstorepass
%cltjkskeystorepass%     -deststorepass %cltp12keystorepass%
%openssl%\bin\openssl pkcs12 -in %cltp12keystore%     -out %cltpemkeystore%      -passin
pass:%cltp12keystorepass% -passout pass:%cltpemkeystorepass%
```

mqcerts.bat

Skript uvádí úložiště klíčů a certifikáty v adresáři certifikátů. Poté vytvoří ukázkového správce front MQTT a nakonfiguruje zabezpečené kanály telemetrie.

```
@echo ____________________________________________________________________________
@echo List keystores and certificates
dir %certpath%\*.* /b
```

```
@rem
@echo Create queue manager and define mqtt channels and certificate stores
call "%MQ_FILE_PATH%\mqxr\Samples\SampleMQM" >> %mqlog%
echo DEFINE CHANNEL(%chlreq%)      CHLTYPE(MQTT) TRPTYPE(TCP) PORT(%portsslreq%)
SSLCAUTH(%authreq%) SSLKEYR('%srvjkskeystore%') SSLKEYP('%srvjkskeystorepass%')
MCAUSER(%mcauser%) | runmqsc %qm% >> %mqlog%
echo DEFINE CHANNEL(%chlopt%)      CHLTYPE(MQTT) TRPTYPE(TCP) PORT(%portsslopt%)
SSLCAUTH(%authopt%) SSLKEYR('%srvjkskeystore%') SSLKEYP('%srvjkskeystorepass%')
MCAUSER(%mcauser%) | runmqsc %qm% >> %mqlog%
V7.5.0.1
echo DEFINE CHANNEL(%chlsslreqws%) CHLTYPE(MQTT) TRPTYPE(TCP) PORT(%portsslreqws%)
SSLCAUTH(%authreq%) SSLKEYR('%srvjkskeystore%') SSLKEYP('%srvjkskeystorepass%')
MCAUSER(%mcauser%) PROTOCOL(HTTP) | runmqsc %qm% >> %mqlog%
```

```
echo DEFINE CHANNEL(%chlssloptws%) CHLTYPE(MQTT) TRPTYPE(TCP) PORT(%portssloptws%)
SSLCAUTH(%authopt%) SSLKEYR('%srvjkskeystore%') SSLKEYP('%srvjkskeystorepass%')
MCAUSER(%mcauser%) PROTOCOL(HTTP) | runmqsc %qm% >> %mqlog%
echo DEFINE CHANNEL(%chlws%)       CHLTYPE(MQTT) TRPTYPE(TCP) PORT(%portws%)
MCAUSER(%mcauser%) PROTOCOL(HTTP) | runmqsc %qm% >> %mqlog%
@echo MQ logs saved in %mqlog%echo
```

Generování klíčů a certifikátů

Postupujte podle této procedury, chcete-li generovat klíče a certifikáty pro klienty Java a C, včetně aplikací Android a iOS , a serverů IBM WebSphere MQ a IBM MessageSight .

Než začnete

1. Musíte mít kopii příkazu **keytool** . Ne všechny verze produktu **keytool** podporují převod úložiště klíčů z úložiště klíčů Java (JKS) na PKCS (Public Key Cryptographic System) nebo k podepisování požadavků na certifikáty. Tento příklad používá příkaz **keytool** v produktu JDK verze 7.0, který podporuje obě tyto schopnosti.
2. Hodláte-li generovat klíče a certifikáty pro klienta pro jazyk C, které jsou ve formátu PEM (Privacy-Enhanced Mail), je nutné mít kopii příkazu **openSSL** . Postupujte podle kroků uvedených v části “Sestavení klienta MQTT pro knihovny C” na stránce 30 a sestavte balík produktu openSSL .
3. Změňte hodnoty parametrů ve skriptu `initcert.bat` tak, aby odpovídaly vašim potřebám. Zejména se můžete rozhodnout, že vynecháte parametry hesla, abyste se vyvarovali zápisu hesel. Příkaz **keytool** vás vyzve k zadání chybějících hesel.

Informace o této úloze

Potřebujete klíče a certifikáty pro vytváření zabezpečených připojení SSL mezi klienty MQTT a servery. Tato úloha ukazuje dva různé způsoby, jak vytvořit klíče a certifikáty, které požadujete: podepsané a podepsané vaším vlastním vydavatelem certifikátů. Způsob, který sledujete, závisí na tom, jak plánujete spravovat úložiště klíčů a certifikáty.

Chcete-li použít certifikáty podepsané externím vydavatelem certifikátů, nahraďte podpisový krok v `cacerts.bat`tím, že odešlete požadavky na certifikát externím vydavatelem certifikátů. Certifikační autorita může navíc k podepsaným certifikátům vrátit přechodný a kořenový certifikát. Postupujte podle pokynů poskytnutých externím CA, kam se mají instalovat vrácené certifikáty.

Server IBM WebSphere MQ vyhledá certifikáty pouze v úložišti certifikátů, které jste zadali v konfiguračních parametrech kanálu telemetrie. Neprovádí navíc vyhledávání v úložišti produktu JSE `cacerts` . Klient produktu Java vyhledá certifikáty v úložišti údajů o důvěryhodnosti, které zadáte. Pokud neuvedete úložiště údajů o důvěryhodnosti, bude hledat v úložišti klíčů produktu `cacerts` v adresáři JSE `jre\lib\security` . Klienti Android vyhledávají certifikáty v předdefinovaném úložišti certifikátů na zařízení Android . Aplikace klienta jazyka C a aplikace iOS prohledávají pouze v úložištích certifikátů, které aplikace uvádí.

Klienti Android a Java prohledávají předkonfigurované úložiště údajů o důvěryhodnosti pro důvěryhodné certifikáty. Kořenové certifikáty CA jsou uloženy v úložišti důvěryhodných certifikátů produktu Android a v úložišti produktu JSE `jre\lib\security\cacerts` . Pokud je kořenový certifikát CA, který certifikován certifikát serveru, již instalován v předkonfigurovaném úložišti údajů o důvěryhodnosti, nedefinujte úložiště údajů o důvěryhodnosti klienta. Jediná konfigurace, která se požaduje, je nastavit port TCP/IP pro zabezpečený kanál serveru MQTT .

Nástroje k vytvoření klíčů a certifikátů a správa všech různých formátů, nejsou jednoduché k použití. Mají mnoho parametrů pro správu a **openSSL** vyžaduje konfigurační soubor, `openssl.cnf`a parametry příkazového řádku. Žádný nástroj poskytuje všechny funkce, které jsou vyžadovány pro správu klíčů a certifikátů pro aplikace, které jsou spuštěny na C a Java. Kanály telemetrie v produktu IBM WebSphere MQ vyžadují úložiště klíčů JKS a tyto příklady používají hlavně nástroje certifikátu produktu Java , **ikeyman** a **keytool**. Nástroje produktu Java však nepodporují formát PEM, který je požadován pro aplikace klienta jazyka C. Chcete-li vytvořit úložiště klíčů ve formátu PEM, spusťte nástroj **openSSL** . Nástroj **openSSL** převádí úložiště klíčů z formátu PKCS12 na formát PEM a produkt **keytool** převádí

úložiště klíčů mezi formátem JKS a formátem PKCS12 . Pro převod úložiště klíčů není vyžadován žádný soubor `openssl.cnf` . Produkt **openSSL** vyžaduje pouze v případě, že plánujete sestavit aplikace klienta jazyka C nebo aplikace iOS . Dáváte-li přednost práci s produktem **openSSL**, můžete jej použít k podepisování certifikátů namísto podepisování certifikátů s produktem **keytool**.

Postup

1. Otevřete příkazové okno a spusťte následující skripty.
2. Vytvořte a spusťte skript `initcert.bat` , abyste nastavili parametry, které jsou nezbytné ke spuštění důvěryhodných ukázkových klientů produktu MQTT .
3. Vytvořte a spusťte skript `cleancert.bat` , abyste vyčistěte prostředí připravené k vytvoření nových úložišť klíčů a certifikátů.
4. Vytvořte a spusťte skript `genkeys.bat` a vygenerujte dvojice klíčů, které požadujete.
5. Proveďte jednu z následujících možností:
   - Vytvořte a spusťte skript `sscerts.bat` , který vytvoří certifikáty podepsané sebou samým.
   - Vytvořte a spusťte skript `cacerts.bat` a vytvořte podpisové řetězce certifikátů certifikačních autorit.
6. Vytvořte a spusťte skript `mqcerts.bat` a vytvořte správce front produktu `MQXR_SAMPLE_QM` a nakonfigurujte jeho kanály telemetrie.

Související úlohy

“Sestavení a spuštění zabezpečeného produktu Ukázková aplikace C klienta MQTT” na stránce 85
Na základě příkladu Windows můžete začít pracovat se zabezpečeným vzorovým aplikací C na libovolném operačním systému, pro který můžete kompilovat zdroj C. Ověřte, že můžete spustit ukázkovou aplikaci C na serveru buď IBM MessageSight , nebo IBM WebSphere MQ jako server MQTT.

“Sestavení a spuštění zabezpečeného produktu Ukázková aplikace klienta MQTT Java” na stránce 54
Na základě příkladu Windows můžete začít pracovat se zabezpečenou ukázkovou aplikací Java na serveru buď IBM MessageSight , nebo IBM WebSphere MQ jako server MQTT. Aplikaci Klient MQTT pro produkt Java můžete spustit na libovolné platformě s JSE 1.5 nebo vyšším, které je "Java Kompatibilní"

Ukázkové skripty pro konfiguraci certifikátů SSL pro produkt Windows

Příklad

Ukázkové příkazové soubory vytvoří certifikáty a úložiště certifikátů, jak je popsáno v krocích v úloze. Kromě toho tento příklad nastaví správce front klienta MQTT pro použití úložiště certifikátů serveru. Příklad odstraní a znovu vytvoří správce front voláním skriptu `SampleMQM.bat`, který je poskytnut s IBM WebSphere MQ.

`initcert.bat`

`initcert.bat` nastavuje názvy a cesty k certifikátům a jiným parametrům, které jsou vyžadovány příkazy **keytool** a **openSSL**. Nastavení jsou popsána v komentářích ve skriptu.

```
@echo off
@rem Set the path where you installed the MQTT SDK
@rem and short cuts to the samples directories.
set SDKRoot=C:\MQTT
set jsamppath=%SDKRoot%\sdk\clients\java\samples
set csamppath=%SDKRoot%\sdk\clients\c\samples
```

```
@rem Set the paths to Version 7 of the JDK
@rem and to the directory where you built the openSSL package.
@rem Set short cuts to the tools.
set javapath=C:\Program Files\IBM\Java70
set keytool="%javapath%\jre\bin\keytool.exe"
set ikeyman="%javapath%\jre\bin\ikeyman.exe"
set openssl=%SDKRoot%\openSSL
set runopenssl="%openssl%\bin\openssl"
```

```
@rem Set the path to where certificates are to be stored,
@rem and set global security parameters.
@rem Omit set password, and the security tools prompt you for passwords.
@rem Validity is the expiry time of the certificates in days.
set certpath=%SDKRoot%\Certificates
set password=password
set validity=5000
set algorithm=RSA
```

```
@rem Set the certificate authority (CA) jks keystore and certificate parameters.
@rem Omit this step, unless you are defining your own certificate authority.
@rem The CA keystore contains the key-pair for your own certificate authority.
@rem You must protect the CA keystore.
@rem The CA certificate is the self-signed certificate authority public certificate.
@rem It is commonly known as the CA root certificate.
set caalias=caalias
set cadname="CN=mqttca.ibm.id.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
set cakeypass=%password%
@rem ca key store
set cajkskeystore=%certpath%\cakeystore.jks
set cajkskeystorepass=%password%
@rem ca certificate (root certificate)
set cacert=%certpath%\cacert.cer
```

```
@rem Set the server jks keystore and certificate parameters.
@rem The server keystore contains the key-pair for the server.
@rem You must protect the server keystore.
@rem If you then export the server certificate it is self-signed.
@rem Alternatively, if you export a certificate signing request (CSR)
@rem from the server keystore for the server key,
@rem and import the signed certificate back into the same keystore,
@rem it forms a certificate chain.
@rem The certificate chain links the server certificate to the CA.
@rem When you now export the server certificate,
@rem the exported certificate includes the certificate chain.
set srvalias=srvalias
set srvdname="CN=mqttserver.ibm.id.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
set srvkeypass=%password%
@rem server key stores
set srvjkskeystore=%certpath%\srvkeystore.jks
set srvjkskeystorepass=%password%
@rem server certificates
set srvcertreq=%certpath%\srvcertreq.csr
set srvcertcasigned=%certpath%\srvcertcasigned.cer
set srvcertselfsigned=%certpath%\srvcertselfsigned.cer
```

```
@rem Set the client jks keystore and certificate parameters
@rem Omit this step, unless you are authenticating clients.
@rem The client keystore contains the key-pair for the client.
@rem You must protect the client keystore.
@rem If you then export the client certificate it is self-signed.
@rem Alternatively, if you export a certificate signing request (CSR)
@rem from the client keystore for the client key,
@rem and import the signed certificate back into the same keystore,
@rem it forms a certificate chain.
@rem The certificate chain links the client certificate to the CA.
@rem When you now export the client certificate,
@rem the exported certificate includes the certificate chain.
set cltalias=cltalias
set cltdname="CN=mqttclient.ibm.id.com, OU=ID, O=IBM, L=Hursley, S=Hants, C=GB"
set cltkeypass=%password%
@rem client key stores
set cltjkskeystore=%certpath%\cltkeystore.jks
set cltjkskeystorepass=%password%
set cltcertreq=%certpath%\cltcertreq.csr
set cltcertcasigned=%certpath%\cltcacertsigned.cer
set cltcertselfsigned=%certpath%\cltcertselfsigned.cer
```

```
@rem Set the paths to the client truststores signed by CA and signed by server key.
@rem You only need to define one of the trust stores.
@rem A trust store holds certificates that you trust,
@rem which are used to authenticate untrusted certificates.
@rem In this example, when the client authenticates the MQTT server it connects to,
@rem it authenticates the certificate it is sent by the server
@rem with the certificates in its trust store.
@rem For example, the MQTT server sends its server certificate,
@rem and the client authenticates it with either the same server certificate
```

```
@rem that you have stored in the cltsrvtruststore.jks trust store,
@rem or against the CA certificate, if the server certificate is signed by the CA.
set cltcajkstruststore=%certpath%\cltcatruststore.jks
set cltcajkstruststorepass=%password%
set cltsrvjkstruststore=%certpath%\cltsrvtruststore.jks
set cltsrvjkstruststorepass=%password%
```

```
@rem Set the paths to the client PKCS12 and PEM key and trust stores.
@rem Omit this step, unless you are configuring a C or iOS client.
@rem You only need to define either one of the trust stores for storing CA
@rem or server signed server certificates.
set cltp12keystore=%certpath%\cltkeystore.p12
set cltp12keystorepass=%password%
set cltpemkeystore=%certpath%\cltkeystore.pem
set cltpemkeystorepass=%password%
set cltcap12truststore=%certpath%\cltcatruststore.p12
set cltcap12truststorepass=%password%
set cltcapemtruststore=%certpath%\cltcatruststore.pem
set cltcapemtruststorepass=%password%
set cltsrvp12truststore=%certpath%\cltsrvtruststore.p12
set cltsrvp12truststorepass=%password%
set cltsrvpemtruststore=%certpath%\cltsrvtruststore.pem
set cltsrvpemtruststorepass=%password%
```

```
@rem set WMQ Variables
set authopt=NEVER
set authreq=REQUIRED
set qm=MQXR_SAMPLE_QM
set host=localhost
set mcauser='Guest'
set portsslopt=8884
set chlopt=SSLOPT
set portsslreq=8885
set chlreq=SSLREQ
V7.5.0.1 set portws=1886
set chlws=PLAINWS
set chlssloptws=SSLOPTWS
set portssloptws=8886
set chlsslreqws=SSLREQWS
set portsslreqws=8887
set mqlog=%certpath%\wmq.log
```

cleancert.bat

Příkazy ve skriptu `cleancert.bat` odstraní správce front klienta MQTT, aby se zajistilo, že úložiště certifikátů serveru není uzamčeno, a potom odstraní všechna úložiště klíčů a certifikáty, které jsou vytvořeny ukázkovými skripty zabezpečení.

```
@rem Delete the MQTT sample queue manager, MQXR_SAMPLE_QM
call "%MQ_FILE_PATH%\bin\setmqenv" -s
endmqm -i %qm%
dltmqm %qm%
```

```
@rem Erase all the certificates and key stores created by the sample scripts.
erase %cajkskeystore%
erase %cacert%
erase %srvjkskeystore%
erase %srvcertreq%
erase %srvcertcasigned%
erase %srvcertselfsigned%
erase %cltjkskeystore%
erase %cltp12keystore%
erase %cltpemkeystore%
erase %cltcertreq%
erase %cltcertcasigned%
erase %cltcertselfsigned%
erase %cltcajkstruststore%
erase %cltcap12truststore%
erase %cltcapemtruststore%
erase %cltsrvjkstruststore%
erase %cltsrvp12truststore%
erase %cltsrvpemtruststore%
erase %mqlog%
@echo Cleared all certificates
dir %certpath%\*.* /b
```

genkeys.bat

Příkazy ve skriptu `genkeys.bat` vytvářejí dvojice klíčů pro soukromé certifikační autority, server a klienta.

```
@rem
@echo ------------------------------------------------------------------------------
@echo Generate %caalias%, %srvalias%, and %cltalias% key-pairs in %cajkskeystore%,
%srvjkskeystore%, and %cltjkskeystore%
@rem
@rem -- Generate a client certificate and a private key pair
@rem Omit this step, unless you are authenticating clients.
%keytool% -genkeypair -noprompt -alias %cltalias% -dname %cltdname% -keystore
%cltjkskeystore% -storepass %cltjkskeystorepass% -keypass %cltkeypass% -keyalg %algorithm%
-validity %validity%
```

```
@rem -- Generate a server certificate and private key pair
%keytool% -genkeypair -noprompt -alias %srvalias% -dname %srvdname% -keystore
%srvjkskeystore% -storepass %srvjkskeystorepass% -keypass %srvkeypass% -keyalg %algorithm%
-validity %validity%
```

```
@rem Create CA, client and server key-pairs
@rem  -- Generate a CA certificate and private key pair - The extension asserts this is a
certificate authority certificate, which is required to import into firefox
%keytool% -genkeypair -noprompt -ext bc=ca:true -alias %caalias%  -dname %cadname%
-keystore %cajkskeystore% -storepass %cajkskeystorepass%  -keypass %cakeypass% -keyalg
%algorithm% -validity %validity%
```

sscerts.bat

Příkazy ve skriptu `sscerts.bat` exportují certifikáty podepsané držitelem klienta a serveru z vlastních úložišť klíčů a importují certifikát serveru do úložiště údajů o důvěryhodnosti klienta a certifikát klienta do úložiště klíčů serveru. Server nemá úložiště údajů o důvěryhodnosti. Příkazy vytvoří úložiště údajů o důvěryhodnosti klienta ve formátu PEM z úložiště údajů o důvěryhodnosti JKS klienta.

```
@rem
@echo ------------------------------------------------------------------------------
@echo Export self-signed certificates:  %srvcertselfsigned% and %cltcertselfsigned%
@rem Export Server public certificate
%keytool% -exportcert -noprompt -rfc -alias %srvalias% -keystore %srvjkskeystore%
-storepass %srvjkskeystorepass% -file %srvcertselfsigned%
@rem Export Client public certificate
@rem Omit this step, unless you are authenticating clients.
%keytool% -exportcert -noprompt -rfc -alias %cltalias% -keystore %cltjkskeystore%
-storepass %cltjkskeystorepass% -file %cltcertselfsigned%
```

```
@rem
@echo ------------------------------------------------------------------------------
@echo Add selfsigned server certificate %srvcertselfsigned% to client trust store:
%cltsrvjkstruststore%
@rem Import the server certificate into the client-server trust store (for server self-
signed authentication)
%keytool% -import -noprompt -alias %srvalias% -file %srvcertselfsigned% -keystore
%cltsrvjkstruststore% -storepass %cltsrvjkstruststorepass%
```

```
@rem
@echo ------------------------------------------------------------------------------
@echo Add selfsigned client certificate %cltcertselfsigned% to server trust store:
%srvjkskeystore%
@rem Import the client certificate into the server trust store (for client self-signed
authentication)
@rem Omit this step, unless you are authenticating clients.
%keytool% -import -noprompt -alias %cltalias% -file %cltcertselfsigned% -keystore
%srvjkskeystore% -storepass %srvjkskeystorepass%
```

```
@rem
@echo ------------------------------------------------------------------------------
@echo Create a pem client-server trust store from the jks client-server trust store:
%cltsrvpemtruststore%
@rem
%keytool% -importkeystore -noprompt -srckeystore %cltsrvjkstruststore% -destkeystore
```

```
%cltsrvp12truststore% -srcstoretype jks -deststoretype pkcs12 -srcstorepass
%cltsrvjkstruststorepass% -deststorepass %cltsrvp12truststorepass%
%openssl%\bin\openssl pkcs12 -in %cltsrvp12truststore% -out %cltsrvpemtruststore% -passin
pass:%cltsrvp12truststorepass% -passout pass:%cltsrvpemtruststorepass%@rem
@rem
@echo ------------------------------------------------------------------------------
@echo Create a pem client key store from the jks client keystore
@rem Omit this step, unless you are configuring a C or iOS client.
@rem
%keytool% -importkeystore -noprompt -srckeystore %cltjkskeystore%      -destkeystore
%cltp12keystore%     -srcstoretype jks -deststoretype pkcs12 -srcstorepass
%cltjkskeystorepass%     -deststorepass %cltp12keystorepass%
%openssl%\bin\openssl pkcs12 -in %cltp12keystore%     -out %cltpemkeystore%      -passin
pass:%cltp12keystorepass% -passout pass:%cltpemkeystorepass%
```

cacerts.bat

Skript importuje kořenový certifikát certifikační autority soukromých úložišť klíčů. Kořenový certifikát CA je potřebný k vytvoření svazku klíčů mezi kořenovým certifikátem a podepsaným certifikátem. Skript `cacerts.bat` exportuje požadavky na certifikáty klienta a serveru ze svých úložišť klíčů. Skript podepíše požadavky na certifikáty s klíčem soukromé certifikační autority v úložišti klíčů `cajkskeystore.jks` a poté naimportuje podepsané certifikáty zpět do stejných úložišť klíčů, ze kterých přišly požadavky. Import vytvoří řetěz certifikátů s kořenovým certifikátem CA. Skript vytvoří úložiště údajů o důvěryhodnosti klienta ve formátu PEM z úložiště údajů o důvěryhodnosti JKS klienta.

```
@rem
@echo ------------------------------------------------------------------------------
@echo Export self-signed certificates: %cacert%
@rem
@rem Export CA public certificate
%keytool% -exportcert -noprompt -rfc -alias %caalias%  -keystore %cajkskeystore% -storepass
%cajkskeystorepass%  -file %cacert%
```

```
@rem
@echo ------------------------------------------------------------------------------
@echo Add CA to server key and client key and trust stores: %srvjkskeystore%,
%cltjkskeystore%, %cltcajkstruststore%,
@rem The CA certificate is necessary to create key chains in the client and server key
stores,
@rem and to certify key chains in the server key store and the client trust store
@rem
@rem Import the CA root certificate into the server key store
%keytool% -import -noprompt -alias %caalias% -file %cacert% -keystore %srvjkskeystore%
-storepass %srvjkskeystorepass%
@rem Import the CA root certificate into the client key store
@rem Omit this step, unless you are authenticating clients.
%keytool% -import -noprompt -alias %caalias% -file %cacert% -keystore %cltjkskeystore%
-storepass %cltjkskeystorepass%
@rem Import the CA root certificate into the client ca-trust store (for ca chained
authentication)
%keytool% -import -noprompt -alias %caalias% -file %cacert% -keystore %cltcajkstruststore%
-storepass %cltcajkstruststorepass%
```

```
@rem
@echo ------------------------------------------------------------------------------
@echo Create certificate signing requests: %srvcertreq% and %cltcertreq%
@rem
@rem Create a certificate signing request (CSR) for the server key
%keytool% -certreq -alias %srvalias% -file %srvcertreq% -keypass %srvkeypass% -keystore
%srvjkskeystore% -storepass %srvjkskeystorepass%
@rem Create a certificate signing request (CSR) for the client key
%keytool% -certreq -alias %cltalias% -file %cltcertreq% -keypass %cltkeypass% -keystore
%cltjkskeystore% -storepass %cltjkskeystorepass%
```

```
@rem
@echo ------------------------------------------------------------------------------
@echo Sign certificate requests: %srvcertcasigned% and %cltcertcasigned%
@rem The requests are signed with the ca key in the cajkskeystore.jks keystore
@rem
@rem Sign server certificate request
%keytool% -gencert -infile %srvcertreq% -outfile %srvcertcasigned% -alias %caalias%
-keystore %cajkskeystore% -storepass %cajkskeystorepass% -keypass %cakeypass%
@rem Sign client certificate request
@rem Omit this step, unless you are authenticating clients.
```

```
%keytool% -gencert -infile %cltcertreq% -outfile %cltcertcasigned% -alias %caalias%
-keystore %cajkskeystore% -storepass %cajkskeystorepass% -keypass %cakeypass%
```

```
@rem
@echo ____________________________________________________________________________
@echo Import the signed certificates back into the key stores to create the key chain:
%srvjkskeystore% and %cltjkskeystore%
@rem
@rem Import the signed server certificate
%keytool% -import -noprompt -alias %srvalias% -file %srvcertcasigned% -keypass %srvkeypass%
-keystore %srvjkskeystore% -storepass %srvjkskeystorepass%
@rem Import the signed client certificate and key chain back into the client keystore
%keytool% -import -noprompt -alias %cltalias% -file %cltcertcasigned% -keypass %cltkeypass%
-keystore %cltjkskeystore% -storepass %cltjkskeystorepass%
@rem
@rem The CA certificate is needed in the server key store, and the client trust store
@rem to verify the key chain sent from the client or server
@echo Delete the CA certificate from %cltjkskeystore%: it causes a problem in converting
keystore to pem
@rem Omit this step, unless you are authenticating clients.
%keytool% -delete -alias %caalias% -keystore %cltjkskeystore% -storepass
%cltjkskeystorepass%
```

```
@rem
@echo ____________________________________________________________________________
@echo Create a pem client-ca trust store from the jks client-ca trust store:
%cltcapemtruststore%
@rem Omit this step, unless you are configuring a C or iOS client.
@rem
%keytool% -importkeystore -noprompt -srckeystore %cltcajkstruststore%  -destkeystore
%cltcap12truststore%  -srcstoretype jks -deststoretype pkcs12 -srcstorepass
%cltcajkstruststorepass%  -deststorepass %cltcap12truststorepass%
%openssl%\bin\openssl pkcs12 -in %cltcap12truststore%  -out %cltcapemtruststore%  -passin
pass:%cltcap12truststorepass%  -passout pass:%cltpemtruststorepass%
```

```
@rem
@echo ____________________________________________________________________________
@echo Create a pem client key store from the jks client keystore
@rem Omit this step, unless you are configuring a C or iOS client.
@rem
%keytool% -importkeystore -noprompt -srckeystore %cltjkskeystore%      -destkeystore
%cltp12keystore%     -srcstoretype jks -deststoretype pkcs12 -srcstorepass
%cltjkskeystorepass%     -deststorepass %cltp12keystorepass%
%openssl%\bin\openssl pkcs12 -in %cltp12keystore%     -out %cltpemkeystore%      -passin
pass:%cltp12keystorepass% -passout pass:%cltpemkeystorepass%
```

mqcerts.bat

Skript uvádí úložiště klíčů a certifikáty v adresáři certifikátů. Poté vytvoří ukázkového správce front MQTT a nakonfiguruje zabezpečené kanály telemetrie.

```
@echo ____________________________________________________________________________
@echo List keystores and certificates
dir %certpath%\*.* /b
```

```
@rem
@echo Create queue manager and define mqtt channels and certificate stores
call "%MQ_FILE_PATH%\mqxr\Samples\SampleMQM" >> %mqlog%
echo DEFINE CHANNEL(%chlreq%)      CHLTYPE(MQTT) TRPTYPE(TCP) PORT(%portsslreq%)
SSLCAUTH(%authreq%) SSLKEYR('%srvjkskeystore%') SSLKEYP('%srvjkskeystorepass%')
MCAUSER(%mcauser%) | runmqsc %qm% >> %mqlog%
echo DEFINE CHANNEL(%chlopt%)      CHLTYPE(MQTT) TRPTYPE(TCP) PORT(%portsslopt%)
SSLCAUTH(%authopt%) SSLKEYR('%srvjkskeystore%') SSLKEYP('%srvjkskeystorepass%')
MCAUSER(%mcauser%) | runmqsc %qm% >> %mqlog%
V7.5.0.1
echo DEFINE CHANNEL(%chlsslreqws%) CHLTYPE(MQTT) TRPTYPE(TCP) PORT(%portsslreqws%)
SSLCAUTH(%authreq%) SSLKEYR('%srvjkskeystore%') SSLKEYP('%srvjkskeystorepass%')
MCAUSER(%mcauser%) PROTOCOL(HTTP) | runmqsc %qm% >> %mqlog%
echo DEFINE CHANNEL(%chlssloptws%) CHLTYPE(MQTT) TRPTYPE(TCP) PORT(%portssloptws%)
SSLCAUTH(%authopt%) SSLKEYR('%srvjkskeystore%') SSLKEYP('%srvjkskeystorepass%')
MCAUSER(%mcauser%) PROTOCOL(HTTP) | runmqsc %qm% >> %mqlog%
echo DEFINE CHANNEL(%chlws%)       CHLTYPE(MQTT) TRPTYPE(TCP) PORT(%portws%)
MCAUSER(%mcauser%) PROTOCOL(HTTP) | runmqsc %qm% >> %mqlog%
@echo MQ logs saved in %mqlog%echo
```

identifikace klienta MQTT, autorizace a ověření

Služba telemetrie (MQXR) publikuje nebo odebírá témata WebSphere MQ v zastoupení klientů MQTT s použitím kanálů MQTT. Administrátor produktu WebSphere MQ konfiguruje identitu kanálu MQTT, která se používá pro autorizaci produktu WebSphere MQ . Administrátor může definovat společnou identitu pro kanál nebo použít `Jméno uživatele` nebo `ClientIdentifier` klienta připojeného k danému kanálu.

Služba telemetrie (MQXR) může ověřit identitu klienta pomocí tokenu `Username` dodaného klientem nebo pomocí certifikátu klienta. `Jméno uživatele` je ověřováno pomocí hesla poskytnutého klientem.

Shrnuto: Identifikace klienta je výběrem identity klienta. V závislosti na kontextu je klient identifikován pomocí `ClientIdentifier`, `Username`, obecné identity klienta vytvořené administrátorem nebo klientským certifikátem. Identifikátor klienta používaný pro kontrolu pravosti nemusí být stejný jako identifikátor, který se používá pro autorizaci.

Klientské programy MQTT nastavují `Jméno uživatele` a `Heslo` , které jsou odeslány na server pomocí kanálu MQTT. Mohou také nastavit vlastnosti zabezpečení SSL, které jsou vyžadovány pro šifrování a ověření připojení. Administrátor rozhodne, zda má být ověřen kanál MQTT a jak ověřit kanál.

Chcete-li autorizovat klienta MQTT pro přístup k objektům produktu IBM WebSphere MQ , autorizujte klienta `ClientIdentifier`nebo `Username` klienta nebo autorizujte společnou identitu klienta. Chcete-li povolit klientovi připojit se k produktu IBM WebSphere MQ, ověřit identitu `Jméno uživatele`nebo použít certifikát klienta. Nakonfigurujte službu JAAS pro ověření uživatele `Jméno uživatele`a nakonfigurujte zabezpečení SSL pro ověření certifikátu klienta.

Pokud na straně klienta nastavíte `Heslo` , buď zašifrujte připojení pomocí sítě VPN, nebo nakonfigurujte kanál MQTT tak, aby používal zabezpečení SSL, aby bylo heslo nastaveno jako soukromé.

Certifikáty klientů je obtížné spravovat. Z tohoto důvodu, pokud jsou rizika spojená s ověřením hesla přijatelná, ověření hesla se často používá k ověření klientů.

Pokud je k dispozici bezpečný způsob správy a ukládání certifikátu klienta, je možné spolehnout se na ověření certifikátu. Avšak zřídka je možné bezpečně spravovat certifikáty v typech prostředí, ve kterých se používá telemetrie. Místo toho je ověření zařízení používajících certifikáty klientů doplněno ověřením hesel klienta na serveru. Z důvodu další složitosti je použití certifikátů klienta omezeno na vysoce citlivé aplikace. Použití dvou forem ověření se nazývá dvoufaktorové ověření. Musíte znát jeden z faktorů, jako je heslo, a mít druhý takový certifikát, jako je například certifikát.

Ve vysoce citlivé aplikaci, jako je například čip a poziční zařízení, je toto zařízení během výroby zablokováno, aby se zabránilo manipulaci s vnitřním hardwarem a softwarem. Důvěryhodný, časově omezený, klientský certifikát je zkopírován do zařízení. Zařízení je implementováno do umístění, kde má být použito. Další ověřování se provádí pokaždé, když je zařízení používáno, buď pomocí hesla, nebo jiným certifikátem z čipové karty.

Identita a autorizace klienta MQTT

K získání přístupu k objektům WebSphere MQ použijte `ClientIdentifier`, `Username`nebo společnou identitu klienta.

Administrátor produktu IBM WebSphere MQ má tři volby pro výběr identity kanálu MQTT. Administrátor provede výběr při definování nebo úpravě kanálu MQTT používaného klientem. Identita se používá k autorizaci přístupu k tématům produktu IBM WebSphere MQ . K dispozici jsou tyto volby:

1. Identifikátor klienta.
2. Identita, kterou administrátor poskytuje pro kanál.
3. `Jméno uživatele` předané z klienta MQTT.

`Jméno uživatele` je atribut třídy voleb MqttConnect. Musí být nastavena dříve, než se klient připojí ke službě. Jeho výchozí hodnota je null.

Pomocí příkazu IBM WebSphere MQ **setmqaut** vyberte, které objekty a které akce mají být autorizovány k použití identitou přidruženou k kanálu MQTT. Chcete-li například autorizovat určitou identitu kanálu, `MQTTClient`, kterou poskytuje administrátor správce front, `QM1`:

```
setmqaut -m QM1 -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p MQTTClient -all +put
setmqaut -m QM1 -t topic -n SYSTEM.BASE.TOPIC -p MQTTClient -all +pub +sub
```

Související informace

Autorizování klientů MQTT pro přístup k objektům produktu WebSphere MQ

Ověření klienta MQTT pomocí hesla

Ověřte `Jméno uživatele` pomocí hesla klienta. Klienta můžete ověřit pomocí jiné identity k totožnosti, která se používá k autorizaci klienta pro publikování a odběr témat.

Služba telemetrie (MQXR) používá službu JAAS k ověření klienta `Username`. Volba JAAS používá `heslo` dodané klientem MQTT.

Administrátor produktu IBM WebSphere MQ rozhoduje, zda ověřit identitu `Jméno uživatele`, nebo ne provést ověření ve všech, konfigurací kanálu MQTT, ke kterému se klient připojuje. Klienti mohou být přiřazeni k různým kanálům a každý kanál lze nakonfigurovat tak, aby ověřoval klienty různými způsoby. Pomocí JAASmůžete nakonfigurovat, které metody musí ověřit klienta a které mohou volitelně autentizovat klienta.

Výběr identity pro ověření nemá vliv na volbu identity pro autorizaci. Možná budete chtít nastavit obecnou identitu pro autorizaci pro administrativní pohodlí, ale autentizovat každého uživatele k použití této identity. Následující postup popisuje kroky k ověření totožnosti jednotlivých uživatelů při používání společné identity:

1. Administrátor produktu IBM WebSphere MQ nastavuje identitu kanálu MQTT na libovolný název, jako např. `MQTTClientUser`, pomocí Průzkumníka IBM WebSphere MQ .
2. Administrátor produktu IBM WebSphere MQ autorizuje produkt `MQTTClient` k publikování a odběru libovolného tématu:

   ```
   setmqaut -m QM1 -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p MQTTClient -all +put
   setmqaut -m QM1 -t topic -n SYSTEM.BASE.TOPIC -p MQTTClient -all +pub +sub
   ```

3. Vývojář aplikací klienta MQTT vytvoří objekt `MqttConnectOptions` a nastaví před připojením k serveru `Jméno uživatele` a `Heslo` .
4. Vývojář zabezpečení vytvoří JAAS `LoginModule` , aby ověřil `Jméno uživatele` s heslem `Heslo` a zahrne jej do konfiguračního souboru JAAS .
5. Administrátor produktu IBM WebSphere MQ konfiguruje kanál MQTT k ověření identity uživatele `UserName` klienta pomocí JAAS.

Ověření klienta MQTT pomocí SSL

Připojení mezi klientem MQTT a správcem front jsou vždy iniciována klientem MQTT. Klient MQTT je vždy klientem SSL. Ověření klienta serveru a ověření serveru klienta MQTT jsou volitelná.

Poskytnutím klienta se soukromým podepsaným digitálním certifikátem můžete ověřit klienta MQTT na serveru IBM WebSphere MQ. Administrátor serveru IBM WebSphere MQ může přinutit klienty MQTT, aby se ověřovali ve správci front pomocí protokolu SSL. Ověření klienta lze požadovat pouze jako součást vzájemného ověření.

Jako alternativu k používání SSL ověřují některé druhy VPN (Virtual Private Network), jako je IPsec, koncové body připojení TCP/IP. VPN šifruje každý IP paket, který se posílá po síti. Jakmile je navázáno spojení s VPN, vytvořili jste důvěryhodnou síť. Klienty MQTT je možné připojit ke kanálům telemetrie pomocí protokolu TCP/IP v síti VPN.

Ověření klienta pomocí SSL spoléhá na to, že klient má tajný klíč. Tajný klíč je soukromý klíč klienta v případě certifikátu podepsaného držitelem nebo klíč poskytnutý certifikační autoritou. Klíč se používá k podepsání digitálního certifikátu klienta. Každý, kdo má k dispozici odpovídající veřejný klíč, může

ověřit digitální certifikát. Certifikáty mohou být důvěryhodné, nebo pokud jsou zřetězeny, jsou trasovány zpět prostřednictvím řetězu certifikátů s důvěryhodným kořenovým certifikátem. Ověření klienta odešle všechny certifikáty v řetězu certifikátů, které poskytuje klient na server. Server kontroluje řetěz certifikátů, dokud nenajde certifikát, kterému důvěřuje. Důvěryhodný certifikát je buď veřejný certifikát generovaný z certifikátu podepsaného držitelem, nebo kořenový certifikát, který je obvykle vydán certifikační autoritou. Jako finální, volitelný, krok lze důvěryhodný certifikát porovnat s "aktivním" seznamem odvolaných certifikátů.

Důvěryhodný certifikát může být vydán certifikační autoritou a je již obsažen v úložišti certifikátů JRE. Může se jednat o certifikát podepsaný držitelem nebo o certifikát, který byl přidán do úložiště klíčů kanálu telemetrie jako důvěryhodný certifikát.

Poznámka: Kanál telemetrie má kombinované úložiště klíčů/úložiště údajů o důvěryhodnosti, které drží soukromé klíče jednomu nebo více kanálům telemetrie, a všechny veřejné certifikáty potřebné k ověření klientů. Vzhledem k tomu, že kanál SSL musí mít úložiště klíčů a jedná se o stejný soubor jako úložiště údajů o důvěryhodnosti kanálu, není nikdy na úložiště certifikátů JRE odkazováno. Z toho vyplývá, že pokud ověření klienta vyžaduje kořenový certifikát CA, musíte umístit kořenový certifikát do úložiště klíčů pro kanál i v případě, že je již kořenový certifikát CA v úložišti certifikátů JRE. Na úložiště certifikátů JRE se nikdy neodkazuje.

Přemýšlejte o hrozbách, s nimiž má ověřování klienta bojovat, a rolích, které klient a server hrají při potlačování hrozeb. Ověřování samotného certifikátu klienta není dostatečné k zabránění neoprávněnému přístupu do systému. Pokud má zařízení klienta v držení nějaký jiný uživatel, nemusí zařízení klienta nutně jednat s oprávněním patřícím držiteli certifikátu. Nikdy se proti nechtěným útokům nespoléhejte na jedinou obranu. Přinejmenším použijte dvoufaktorovou metodu ověření a doplňte držení certifikátu znalostmi o soukromých informacích. Můžete například použít službu JAAS a ověřit klienta pomocí hesla vydaného serverem.

Primární hrozbou pro certifikát klienta je to, že se dostane do špatných rukou. Certifikát je uchováván v heslem chráněném úložišti klíčů u klienta. Jak se dostane do úložiště klíčů? Jakým způsobem klient MQTT získá heslo k úložišti klíčů? Jak je bezpečná je ochrana pomocí hesla? Telemetrická zařízení jsou často snadno odnímatelná, a pak mohou být někde v soukromí napadnuta. Musí být hardware zařízení odolný proti neoprávněné manipulaci? Obtíže s rozdělením a ochranou certifikátů na straně klienta jsou známy, označují se jako problém se správou klíčů.

Sekundární hrozbou je, že zařízení může být zneužito pro přístup k serverům nezamýšlenými způsoby. Například když je manipulováno s aplikací MQTT, lze využít slabinu v konfiguraci serveru s použitím identity ověřeného klienta.

Chcete-li ověřit klienta MQTT pomocí protokolu SSL, nakonfigurujte kanál telemetrie a klienta.

-
-

Konfigurace klienta MQTT pro ověření klienta pomocí SSL

Chcete-li ověřit klienta MQTT pomocí protokolu SSL, připojí se klient k kanálu telemetrie pomocí zabezpečení SSL. Musí určovat port TCP, který odpovídá kanálu telemetrie, který je konfigurován pro ověřování klientů SSL.

Například na straně klienta:

```
MQTTClient mqttClient = new MqttClient( "ssl://www.example.org:8884", "clientId1");
mqttClient.connect();
```

Prostředí JVM klienta musí používat standardní továrnu na sokety z prostředí JSSE. Pokud používáte prostředí Java ME, musíte se ujistit, že je načten balík JSSE. Pokud používáte prostředí Java SE, rozšíření JSSE bylo začleněno s prostředím JRE od verze jazyka Java 1.4.1.

Připojení SSL vyžaduje před připojením nastavit různé vlastnosti SSL. Vlastnosti můžete nastavit tak, že je předáte do prostředí JVM pomocí přepínače `-D` , nebo můžete nastavit vlastnosti pomocí metody `MqttConnectionOptions.setSSLProperties` .

Pokud zavedete nestandardní továrnu soketů voláním metody `MqttConnectOptions.setSocketFactory(javax.net.SocketFactory)`, pak způsob, jakým se nastavení SSL předávají do soketu sítě, je definován aplikací.

Přidejte digitální certifikát klienta, podepsaný buď pomocí soukromého klíče klienta, nebo pomocí CA, do úložiště klíčů chráněného heslem na klientovi. Pokud má certifikát klíčový řetězec, můžete přidat certifikáty z řetězce klíče do úložiště. Když server ověřuje certifikát klienta, používá certifikáty odeslané klientem, aby se shodovaly s certifikáty v úložišti klíčů. Hledá první shodu v klíčovém řetězci s certifikátem, který má. Zbytek řetězce klíčů je ignorován.

Klient MQTT odešle všechny certifikáty v úložišti klíčů na server. Pokud server ověří některý z řetězců klíčů, které klient odesílá, je klient ověřen.

Šifrovací sady zabezpečení SSL můžete také použít pro ověřování klienta. Zde je abecední seznam šifrovacích sad SSL, které jsou momentálně podporované:

- `SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA`
- `SSL_DH_anon_EXPORT_WITH_RC4_40_MD5`
- `SSL_DH_anon_WITH_3DES_EDE_CBC_SHA`
- `SSL_DH_anon_WITH_AES_128_CBC_SHA`
- `SSL_DH_anon_WITH_DES_CBC_SHA`
- `SSL_DH_anon_WITH_RC4_128_MD5`
- `SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA`
- `SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA`
- `SSL_DHE_DSS_WITH_AES_128_CBC_SHA`
- `SSL_DHE_DSS_WITH_DES_CBC_SHA`
- `SSL_DHE_DSS_WITH_RC4_128_SHA`
- `SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA`
- `SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA`
- `SSL_DHE_RSA_WITH_AES_128_CBC_SHA`
- `SSL_DHE_RSA_WITH_DES_CBC_SHA`
- `SSL_KRB5_EXPORT_WITH_DES_CBC_40_MD5`
- `SSL_KRB5_EXPORT_WITH_DES_CBC_40_SHA`
- `SSL_KRB5_EXPORT_WITH_RC4_40_MD5`
- `SSL_KRB5_EXPORT_WITH_RC4_40_SHA`
- `SSL_KRB5_WITH_3DES_EDE_CBC_MD5`
- `SSL_KRB5_WITH_3DES_EDE_CBC_SHA`
- `SSL_KRB5_WITH_DES_CBC_MD5`
- `SSL_KRB5_WITH_DES_CBC_SHA`
- `SSL_KRB5_WITH_RC4_128_MD5`
- `SSL_KRB5_WITH_RC4_128_SHA`
- `SSL_RSA_EXPORT_WITH_DES40_CBC_SHA`
- `SSL_RSA_EXPORT_WITH_RC4_40_MD5`
- `SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA`
- V 7.5.0.2 `SSL_RSA_FIPS_WITH_AES_128_CBC_SHA256`
- V 7.5.0.2 `SSL_RSA_FIPS_WITH_AES_256_CBC_SHA256`
- `SSL_RSA_FIPS_WITH_DES_CBC_SHA`
- `SSL_RSA_WITH_3DES_EDE_CBC_SHA`

- SSL_RSA_WITH_AES_128_CBC_SHA
- V7.5.0.2 SSL_RSA_WITH_AES_128_CBC_SHA256
- V7.5.0.2 SSL_RSA_WITH_AES_256_CBC_SHA256
- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_WITH_NULL_MD5
- SSL_RSA_WITH_NULL_SHA
- V7.5.0.2 SSL_RSA_WITH_NULL_SHA256
- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA

V7.5.0.2 Plánujete-li používat šifrovací sady SHA-2 , viz “Systémové požadavky pro použití šifrovacích sad SHA-2 s klienty MQTT” na stránce 171.

Související pojmy

“Konfigurace klienta MQTT pro ověření kanálu pomocí zabezpečení SSL” na stránce 107
Chcete-li ověřit kanál telemetrie pomocí zabezpečení SSL, musí se klient připojit ke kanálu telemetrie pomocí zabezpečení SSL. Musí určovat port, který odpovídá kanálu telemetrie konfigurovanému pro zabezpečení SSL. Konfigurace musí obsahovat úložiště klíčů chráněné heslem, které obsahuje soukromě podepsaný digitální certifikát serveru.

Ověření kanálu telemetrie pomocí zabezpečení SSL

Připojení mezi klientem MQTT a správcem front jsou vždy iniciována klientem MQTT. Klient MQTT je vždy klientem SSL. Ověření klienta serveru a ověření serveru klienta MQTT jsou volitelná.

Klient se vždy pokusí o ověření serveru, pokud není nakonfigurován pro použití specifikace CipherSpec, která podporuje anonymní připojení. Pokud se ověření nezdaří, připojení není navázáno.

Jako alternativu k používání SSL ověřují některé druhy VPN (Virtual Private Network), jako je IPsec, koncové body připojení TCP/IP. VPN šifruje každý IP paket, který se posílá po síti. Jakmile je navázáno spojení s VPN, vytvořili jste důvěryhodnou síť. Klienty MQTT je možné připojit ke kanálům telemetrie pomocí protokolu TCP/IP v síti VPN.

Ověření serveru pomocí SSL ověřuje server, na který chcete zasílat důvěrné informace. Klient provádí kontroly, které odpovídají certifikátům odeslaným ze serveru, proti certifikátům umístěným v úložišti údajů o důvěryhodnosti nebo ve svém úložišti prostředí JRE `cacerts` .

Úložiště certifikátů JRE je soubor JKS, `cacerts`. Nachází se v adresáři JRE `InstallPath\lib\security\`. Je nainstalován s výchozím heslem `changeit`. Můžete buď uložit certifikáty, kterým důvěřujete, do úložiště certifikátů JRE, nebo do úložiště údajů o důvěryhodnosti klienta. Nelze používat obě úložiště. Úložiště údajů o důvěryhodnosti klienta použijte v případě, že chcete zachovat veřejné certifikáty, kterým klient důvěřuje, od certifikátů, které používají jiné aplikace Java. Úložiště certifikátů JRE používejte v případě, že chcete používat společné úložiště certifikátů pro všechny aplikace Java spuštěné v rámci klienta. Rozhodnete-li se použít úložiště certifikátů JRE, zkontrolujte certifikáty, které obsahuje, abyste se ujistili, že jim důvěřujete.

Konfiguraci JSSE můžete upravit zadáním jiného poskytovatele důvěryhodnosti. Poskytovatele důvěryhodnosti můžete upravit tak, aby prováděl různé kontroly certifikátů. V některých prostředích OGSAN, které používají klienta MQTT, prostředí poskytuje jiného poskytovatele důvěryhodnosti.

Chcete-li ověřit kanál telemetrie pomocí SSL, nakonfigurujte server a klienta.

•

Související pojmy

“Konfigurace klienta MQTT pro ověření kanálu pomocí zabezpečení SSL” na stránce 107
Chcete-li ověřit kanál telemetrie pomocí zabezpečení SSL, musí se klient připojit ke kanálu telemetrie pomocí zabezpečení SSL. Musí určovat port, který odpovídá kanálu telemetrie konfigurovanému pro

zabezpečení SSL. Konfigurace musí obsahovat úložiště klíčů chráněné heslem, které obsahuje soukromě podepsaný digitální certifikát serveru.

Konfigurace klienta MQTT pro ověření kanálu pomocí zabezpečení SSL

Chcete-li ověřit kanál telemetrie pomocí zabezpečení SSL, musí se klient připojit ke kanálu telemetrie pomocí zabezpečení SSL. Musí určovat port, který odpovídá kanálu telemetrie konfigurovanému pro zabezpečení SSL. Konfigurace musí obsahovat úložiště klíčů chráněné heslem, které obsahuje soukromě podepsaný digitální certifikát serveru.

Například na straně klienta:

```
MQTTClient mqttClient = new MqttClient( "ssl://www.example.org:8884", "clientId1");
mqttClient.connect();
```

Prostředí JVM klienta musí používat standardní továrnu na sokety z prostředí JSSE. Pokud používáte prostředí Java ME, musíte se ujistit, že je načten balík JSSE. Pokud používáte prostředí Java SE, rozšíření JSSE bylo začleněno s prostředím JRE od verze jazyka Java 1.4.1.

Připojení SSL vyžaduje před připojením nastavit různé vlastnosti SSL. Vlastnosti můžete nastavit tak, že je předáte do prostředí JVM pomocí přepínače `-D` , nebo můžete nastavit vlastnosti pomocí metody `MqttConnectionOptions.setSSLProperties` .

Pokud zavedete nestandardní továrnu soketů voláním metody `MqttConnectOptions.setSocketFactory(javax.net.SocketFactory)`, pak způsob, jakým se nastavení SSL předávají do soketu sítě, je definován aplikací.

Zadejte kód klienta pro připojení k kanálu telemetrie pomocí zabezpečení SSL a konfiguraci klienta tak, aby důvěřoval certifikátu serveru jedním ze tří způsobů:

Použití certifikátu serveru, který je podepsán dobře známou certifikační autoritou v úložišti `cacerts` .

Pokud server odesílá všechny pomocné klíče v řetězu certifikátů, žádná další konfigurace. Doporučuje se, abyste přezkoumali certifikáty v úložišti `cacerts` klienta JRE a změnili heslo na úložiště `cacerts` .

Další certifikáty

Uložte certifikáty, které důvěřujete v úložišti údajů o důvěryhodnosti na klientovi. Musíte uložit alespoň jeden z certifikátů v řetězu certifikátů v úložišti údajů o důvěryhodnosti, nastavit parametry úložiště údajů o důvěryhodnosti v produktu `MqttConnectionOptions.SSLProperty`.

- `com.ibm.ssl.trustStore`
- `com.ibm.ssl.trustStorePassword`

Použití vlastního správce důvěry

Implementujte poskytovatele důvěryhodnosti a předejte mu název použitého algoritmu. Nastavte název třídy poskytovatele a algoritmus, který má být použit v produktu `MqttConnectionOptions.SSLProperty`.

- `com.ibm.ssl.trustStoreProvider`
- `com.ibm.ssl.trustStoreManager`

Šifrovací sady zabezpečení SSL můžete také použít pro ověření kanálu. Zde je abecední seznam šifrovacích sad SSL, které jsou momentálně podporované:

- `SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA`
- `SSL_DH_anon_EXPORT_WITH_RC4_40_MD5`
- `SSL_DH_anon_WITH_3DES_EDE_CBC_SHA`
- `SSL_DH_anon_WITH_AES_128_CBC_SHA`
- `SSL_DH_anon_WITH_DES_CBC_SHA`

- SSL_DH_anon_WITH_RC4_128_MD5
- SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
- SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA
- SSL_DHE_DSS_WITH_AES_128_CBC_SHA
- SSL_DHE_DSS_WITH_DES_CBC_SHA
- SSL_DHE_DSS_WITH_RC4_128_SHA
- SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_DHE_RSA_WITH_AES_128_CBC_SHA
- SSL_DHE_RSA_WITH_DES_CBC_SHA
- SSL_KRB5_EXPORT_WITH_DES_CBC_40_MD5
- SSL_KRB5_EXPORT_WITH_DES_CBC_40_SHA
- SSL_KRB5_EXPORT_WITH_RC4_40_MD5
- SSL_KRB5_EXPORT_WITH_RC4_40_SHA
- SSL_KRB5_WITH_3DES_EDE_CBC_MD5
- SSL_KRB5_WITH_3DES_EDE_CBC_SHA
- SSL_KRB5_WITH_DES_CBC_MD5
- SSL_KRB5_WITH_DES_CBC_SHA
- SSL_KRB5_WITH_RC4_128_MD5
- SSL_KRB5_WITH_RC4_128_SHA
- SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_RSA_EXPORT_WITH_RC4_40_MD5
- SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA
- V7.5.0.2 SSL_RSA_FIPS_WITH_AES_128_CBC_SHA256
- V7.5.0.2 SSL_RSA_FIPS_WITH_AES_256_CBC_SHA256
- SSL_RSA_FIPS_WITH_DES_CBC_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_AES_128_CBC_SHA
- V7.5.0.2 SSL_RSA_WITH_AES_128_CBC_SHA256
- V7.5.0.2 SSL_RSA_WITH_AES_256_CBC_SHA256
- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_WITH_NULL_MD5
- SSL_RSA_WITH_NULL_SHA
- V7.5.0.2 SSL_RSA_WITH_NULL_SHA256
- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA

V7.5.0.2 Plánujete-li používat šifrovací sady SHA-2 , viz “Systémové požadavky pro použití šifrovacích sad SHA-2 s klienty MQTT” na stránce 171.

Související pojmy

“Konfigurace klienta MQTT pro ověření klienta pomocí SSL” na stránce 104

Chcete-li ověřit klienta MQTT pomocí protokolu SSL, připojí se klient k kanálu telemetrie pomocí zabezpečení SSL. Musí určovat port TCP, který odpovídá kanálu telemetrie, který je konfigurován pro ověřování klientů SSL.

Soukromí zveřejňování kanálů telemetrie

Soukromí publikování protokolu MQTT odeslané v obou směrech přes kanály telemetrie jsou zabezpečeny pomocí SSL pro šifrování přenosů přes připojení.

Klienti MQTT, kteří se připojují k telemetrickým kanálům, používají zabezpečení SSL k zabezpečení soukromých informací přenášených na kanálu pomocí symetrického šifrování klíče. Vzhledem k tomu, že koncové body nejsou ověřeni, nelze samotné šifrování kanálu důvěřovat. Kombinuje zabezpečení ochrany soukromí se serverem nebo vzájemným autentizací.

Jako alternativu k používání SSL ověřují některé druhy VPN (Virtual Private Network), jako je IPsec, koncové body připojení TCP/IP. VPN šifruje každý IP paket, který se posílá po síti. Jakmile je navázáno spojení s VPN, vytvořili jste důvěryhodnou síť. Klienty MQTT je možné připojit ke kanálům telemetrie pomocí protokolu TCP/IP v síti VPN.

Pro typickou konfiguraci, která šifruje kanál a ověřuje server, nahlédněte do “Ověření kanálu telemetrie pomocí zabezpečení SSL” na stránce 106.

Šifrování připojení SSL bez ověření serveru odkrývá připojení k útokům typu "man-in-the-middle-middle". Ačkoli informace, které vyměňujete, jsou chráněny proti odposlouchávání, nevíte, s kým si ji vyměňujete. Pokud neovládnou síť, vystavujete se někomu, kdo zadrží vaše IP přenosy a maskuje se jako koncový bod.

Můžete vytvořit šifrované připojení SSL bez ověření serveru pomocí výměny klíčů Diffie-Hellman CipherSpec , která podporuje anonymní SSL. Hlavní utajený údaj, sdílený mezi klientem a serverem a používaný k šifrování přenosů SSL, je vytvořen bez výměny soukromě podepsaného certifikátu serveru.

Vzhledem k tomu, že anonymní připojení jsou nezabezpečená, většina implementací SSL nestandardně používá anonymní CipherSpecs. Pokud je požadavek klienta na připojení SSL přijat kanálem telemetrie, kanál musí mít úložiště klíčů chráněné heslem. Ve výchozím nastavení, protože implementace SSL nepoužívají anonymní CipherSpecs, musí úložiště klíčů obsahovat soukromě podepsaný certifikát, který může klient ověřit.

Pokud použijete anonymní CipherSpecs, úložiště klíčů serveru musí existovat, ale nemusí obsahovat žádné soukromě podepsané certifikáty.

Jiným způsobem, jak navázat šifrované spojení, je nahradit poskytovatele důvěryhodnosti na klientovi vlastním implementací. Váš poskytovatel důvěryhodnosti by neověřil certifikát serveru, ale připojení by bylo šifrováno.

Konfigurace zabezpečení SSL pro klienty MQTT a kanály telemetrie

Klienti MQTT a služba WebSphere MQ Telemetry (MQXR) používají rozšíření JSSE (Java Secure Socket Extension) pro připojení telemetrických kanálů pomocí zabezpečení SSL. Klienti MQTT C a démon produktu WebSphere MQ Telemetry pro zařízení nepodporují zabezpečení SSL.

Nakonfigurujte zabezpečení SSL pro ověření kanálu telemetrie a klienta MQTT a zašifrujte přenos zpráv mezi klienty a telemetrickým kanálem.

Jako alternativu k používání SSL ověřují některé druhy VPN (Virtual Private Network), jako je IPsec, koncové body připojení TCP/IP. VPN šifruje každý IP paket, který se posílá po síti. Jakmile je navázáno spojení s VPN, vytvořili jste důvěryhodnou síť. Klienty MQTT je možné připojit ke kanálům telemetrie pomocí protokolu TCP/IP v síti VPN.

Můžete nakonfigurovat připojení mezi klientem Java MQTT a kanálem telemetrie, chcete-li používat protokol SSL přes TCP/IP. To, co je zabezpečeno, závisí na tom, jak konfigurovat SSL pro použití JSSE. Počínaje nejbezpečnějšími konfiguracema můžete nakonfigurovat tři různé úrovně zabezpečení:

1. Povolte připojení pouze důvěryhodných klientů MQTT. Připojte klienta MQTT pouze k důvěryhodnému kanálu telemetrie. Šifrování zpráv mezi klientem a správcem front; viz “Ověření klienta MQTT pomocí SSL” na stránce 103.

2. Připojte klienta MQTT pouze k důvěryhodnému kanálu telemetrie. Šifrování zpráv mezi klientem a správcem front; viz “Ověření kanálu telemetrie pomocí zabezpečení SSL” na stránce 106.
3. Šifrování zpráv mezi klientem a správcem front; viz “Soukromí zveřejňování kanálů telemetrie” na stránce 109.

Konfigurační parametry JSSE

Upravte parametry JSSE, abyste změnili způsob, jakým je nakonfigurováno připojení SSL. Konfigurační parametry JSSE jsou uspořádány do tří sad:

1. Kanál telemetrieIBM WebSphere MQ
2. Klient MQTT Java
3. JRE

Konfigurujte parametry kanálu telemetrie pomocí Průzkumníka IBM WebSphere MQ . Nastavte parametry klienta protokolu MQTT Java v atributu `MqttConnectionOptions.SSLProperties` . Upravte parametry zabezpečení prostředí JRE úpravou souborů v adresáři zabezpečení prostředí JRE jak na klientu, tak na serveru.

IBM WebSphere MQ kanál telemetrie

Nastavte všechny parametry zabezpečení SSL kanálu telemetrie pomocí Průzkumníka produktu WebSphere MQ .

`ChannelName`

`ChannelName` je povinný parametr u všech kanálů.

Název kanálu identifikuje kanál přidružený k určitému číslu portu. Kanály názvů, které vám pomohou spravovat sady klientů MQTT.

`PortNumber`

`PortNumber` je volitelný parametr na všech kanálech. Výchozí hodnota je 1883 pro kanály TCP a 8883 pro kanály SSL.

Číslo portu TCP/IP přidružené k tomuto kanálu. Klienti MQTT jsou připojeni k kanálu zadáním portu definovaného pro kanál. Má-li kanál vlastnosti SSL, musí se klient připojit pomocí protokolu SSL, například:

```
MQTTClient mqttClient = new MqttClient( "ssl://www.example.org:8884", "clientId1");
mqttClient.connect();
```

`KeyFile`

Volba `KeyFileName` je požadovaným parametrem pro kanály SSL. Musí být vynechán pro kanály TCP.

`KeyFileName` je cesta k úložišti klíčů Java obsahující digitální certifikáty, které poskytujete. Použijte JKS, JCEKS nebo PKCS12 jako typ úložiště klíčů na serveru.

Identifikujte typ úložiště klíčů pomocí jednoho z následujících přípon souborů:

```
.jks
.jceks
.p12
.pkcs12
```

Předpokládá se, že úložiště klíčů JKS je úložištěm klíčů s jinou příponou souboru.

Na serveru můžete kombinovat jeden typ úložiště klíčů s jinými typy úložiště klíčů na straně klienta.

Uložte soukromý certifikát serveru do úložiště klíčů. Certifikát je znám jako certifikát serveru. Certifikát může být podepsán sám sebou nebo může být součástí řetězu certifikátů, který je podepsán podpisovým orgánem.

Používáte-li řetěz certifikátů, umístěte přidružené certifikáty do úložiště klíčů serveru.

Certifikát serveru a všechny certifikáty ve svém řetězu certifikátů jsou odeslány klientům za účelem ověření identity serveru.

Pokud jste nastavili `ClientAuth` na `Required`, úložiště klíčů musí obsahovat všechny certifikáty nezbytné k ověření klienta. Klient odešle certifikát podepsaný svým držitelem nebo řetěz certifikátů a klient je autentizován prvním ověřením tohoto materiálu proti certifikátu v úložišti klíčů. Pomocí řetězu certifikátů může jeden certifikát ověřit mnoho klientů, i když je vydáván s různými klientskými certifikáty.

PassPhrase

`PassPhrase` je povinný parametr pro kanály SSL. Musí být vynechán pro kanály TCP.

Přístupová fráze se používá k ochraně úložiště klíčů.

ClientAuth

Parametr `ClientAuth` je volitelným parametrem zabezpečení SSL. Výchozí hodnota je bez ověření klienta. Musí být vynechán pro kanály TCP.

Nastavte volbu `ClientAuth` , pokud chcete, aby služba telemetrie (MQXR) ověřila klienta, před tím, než povolíte klientovi připojení k telemetrickým kanálům.

Nastavíte-li `ClientAuth`, klient se musí připojit k serveru pomocí SSL a ověřit server. V odpovědi na nastavení `ClientAuth`odešle klient svůj digitální certifikát na server a všechny ostatní certifikáty ve svém úložišti klíčů. Jeho digitální certifikát je známý jako certifikát klienta. Tyto certifikáty jsou ověřovány proti certifikátům uchovávané v úložišti klíčů kanálu a v úložišti prostředí JRE `cacerts` .

CipherSuite

Hodnota `CipherSuite` je volitelným parametrem zabezpečení SSL. Výchozí nastavení je vyzkoušet všechny povolené specifikace CipherSpecs. Musí být vynechán pro kanály TCP.

Chcete-li použít konkrétní CipherSpec, nastavte `CipherSuite` na název CipherSpec , který musí být použit k ustanovení připojení SSL.

Služba telemetrie a klient MQTT vyjednává obecnou specifikaci CipherSpec ze všech specifikací CipherSpecs , které jsou povoleny na každém konci. Je-li na obou koncích připojení zadána specifická hodnota CipherSpec , musí se shodovat s položkou CipherSpec na druhém konci.

Nainstalujte další šifry přidáním dalších poskytovatelů do JSSE.

Federální standardy zpracování informací (FIPS)

Standard FIPS je volitelné nastavení. Ve výchozím nastavení není nastavena.

Buď na panelu vlastností správce front, nebo pomocí obslužného programu **runmqsc**nastavte SSLFIPS. Volba SSLFIPS určuje, zda mají být použity pouze algoritmy s certifikací FIPS.

Seznam názvů revokace

Seznam názvů revoze je volitelné nastavení. Ve výchozím nastavení není nastavena.

Buď na panelu vlastností správce front, nebo pomocí obslužného programu **runmqsc**nastavte SSLCRLNL. SSLCRLNL uvádí seznam názvů objektů ověřovacích informací, které se používají k poskytnutí umístění odvolaných certifikátů.

Nejsou použity žádné další parametry správce front, které nastavují vlastnosti zabezpečení SSL.

Klient MQTT Java

Nastavte vlastnosti SSL pro klienta Java v souboru `MqttConnectionOptions.SSLProperties`; například:

```
java.util.Properties sslClientProperties = new Properties();
sslClientProperties.setProperty("com.ibm.ssl.keyStoreType", "JKS");
com.ibm.micro.client.mqttv3.MqttConnectOptions conOptions = new MqttConnectOptions();
conOptions.setSSLProperties(sslClientProperties);
```

Názvy a hodnoty specifických vlastností jsou popsány v dokumentaci rozhraní API pro produkt `MqttConnectOptions`. Pro odkazy na dokumentaci ke klientským rozhraním API pro knihovny klienta MQTT viz Odkaz na programování klienta MQTT.

`Protokol`

`Protokol` je volitelný.

Protokol je vybrán při vyjednávání s telemetrickým serverem. Požadujete-li určitý protokol, můžete jej vybrat. Pokud telemetrický server nepodporuje protokol, připojení selže.

`ContextProvider`

`ContextProvider` je volitelný.

`KeyStore`

`KeyStore` je volitelné. Nakonfigurujte ji, pokud je na serveru nastaveno `ClientAuth` , aby se vynutila autentizace klienta.

Umístěte digitální certifikát klienta, podepsaný pomocí svého soukromého klíče do úložiště klíčů. Zadejte cestu k úložišti klíčů a heslo. Typ a poskytovatel jsou volitelné. JKS je výchozí typ a IBMJCE je výchozí poskytovatel.

Určete jiného poskytovatele úložiště klíčů, který bude odkazovat na třídu, která přidává nového poskytovatele úložiště klíčů. Chcete-li vytvořit instanci produktu `KeyManagerFactory` , nastavte název algoritmu používaného poskytovatelem úložiště klíčů tak, že nastavíte název správce klíčů.

`TrustStore`

Parametr `TrustStore` je volitelný. Všechny certifikáty, které důvěřujete, můžete umístit do úložiště JRE `cacerts` .

Konfigurujte úložiště údajů o důvěryhodnosti, chcete-li mít pro klienta jiné úložiště údajů o důvěryhodnosti. Úložiště údajů o důvěryhodnosti byste neměli konfigurovat, pokud server používá certifikát vydaný dobře známou CA, která již má svůj kořenový certifikát uložený v produktu `cacerts`.

Přidejte veřejně podepsaný certifikát serveru nebo kořenového certifikátu do úložiště údajů o důvěryhodnosti a uveďte cestu k úložišti údajů o důvěryhodnosti a heslo. JKS je výchozí typ a IBMJCE je výchozí poskytovatel.

Uveďte jiného poskytovatele úložiště údajů o důvěryhodnosti, který bude odkazovat na třídu, která přidává nového poskytovatele úložiště údajů o důvěryhodnosti. Předejte název algoritmu používaného poskytovatelem úložiště údajů o důvěryhodnosti pro vytvoření instance správce `TrustManagerFactory` nastavením názvu správce důvěryhodnosti.

JRE

Další aspekty zabezpečení produktu Java , které mají vliv na chování zabezpečení SSL na straně klienta i serveru, jsou v prostředí JRE konfigurovány. Konfigurační soubory na Windows jsou v *`Java Installation Directory`*`\jre\lib\security`. Pokud používáte prostředí JRE dodané s produktem IBM WebSphere MQ , cesta je uvedena v následující tabulce:

Tabulka 3. Cesty k souborům podle platformy pro konfigurační soubory zabezpečení SSL prostředí JRE	
Platforma	**Cesta k souboru**
Windows	*WMQ Installation Directory*\java\jre\lib\security
Linux pro System x 32 bitů	*WMQ Installation Directory*/java/jre/lib/security
Ostatní platformy UNIX and Linux	*WMQ Installation Directory*/java/jre64/jre/lib/security

Dobře známá certifikační autority

Soubor `cacerts` obsahuje kořenové certifikáty známých certifikačních autorit. `cacerts` se standardně použije, pokud neuvedete úložiště údajů o důvěryhodnosti. Pokud používáte úložiště `cacerts` nebo neposkytujete úložiště údajů o důvěryhodnosti, musíte přezkoumat a upravit seznam podepisujících subjektů v produktu `cacerts` tak, aby splňoval vaše požadavky na zabezpečení.

Produkt `cacerts` můžete otevřít pomocí příkazu WebSphere MQ `strmqikm`, který spustí obslužný program správy klíčů IBM . Otevřete soubor `cacerts` jako soubor JKS s použitím hesla `changeit`. Upravte heslo tak, aby byl soubor zabezpečen.

Konfigurace tříd zabezpečení

Použijte soubor `java.security` k registraci dalších poskytovatelů zabezpečení a dalších výchozích vlastností zabezpečení.

Oprávnění

Použijte soubor `java.policy` k úpravě oprávnění udělených prostředkům. `javaws.policy` uděluje oprávnění pro `javaws.jar`

Odolnost šifrování

Některá prostředí JRE se dodávají se sníženým šifrováním odolnosti. Pokud nemůžete importovat klíče do úložišť klíčů, může být příčinou snížení odolnosti šifrování. Buď zkuste spustit příkaz **ikeyman** pomocí příkazu **strmqikm** , nebo stáhněte silné, ale omezené soubory jurisdikcí z IBM Developer Kit, Security information.

Důležité: Vaše země původu by mohla mít omezení týkající se dovozu, držení, užívání nebo přeexportování do jiné země, šifrovacího softwaru. Před stažením nebo používáním neomezených souborů zásad je třeba zkontrolovat zákony vaší země. Zkontrolujte jeho předpisy a zásady týkající se dovozu, držení, užívání a zpětného exportu šifrovacího softwaru za účelem zjištění, zda je tento šifrovací software povolen.

Upravte poskytovatele důvěryhodnosti tak, aby se klient mohl připojit k libovolnému serveru.

Následující příklad ilustruje, jak přidat poskytovatele důvěryhodnosti a odkázat jej z kódu klienta MQTT. Tento příklad nezajišťuje žádné ověření klienta nebo serveru. Výsledné připojení SSL je šifrováno, aniž by bylo ověřováno.

Úsek kódu v produktu Obrázek 20 na stránce 113 nastavuje poskytovatele důvěry `AcceptAllProviders` a správce důvěryhodnosti pro klienta MQTT.

```
java.security.Security.addProvider(new AcceptAllProvider());
java.util.Properties sslClientProperties = new Properties();
sslClientProperties.setProperty("com.ibm.ssl.trustManager","TrustAllCertificates");
sslClientProperties.setProperty("com.ibm.ssl.trustStoreProvider","AcceptAllProvider");
conOptions.setSSLProperties(sslClientProperties);
```

Obrázek 20. Úsek kódu klienta MQTT

```
package com.ibm.mq.id;
public class AcceptAllProvider extends java.security.Provider {
    private static final long serialVersionUID = 1L;
    public AcceptAllProvider() {
        super("AcceptAllProvider", 1.0, "Trust all X509 certificates");
        put("TrustManagerFactory.TrustAllCertificates",
                AcceptAllTrustManagerFactory.class.getName());
    }
```

Obrázek 21. AcceptAllProvider.java

```
    protected static class AcceptAllTrustManagerFactory extends
            javax.net.ssl.TrustManagerFactorySpi {
        public AcceptAllTrustManagerFactory() {}
        protected void engineInit(java.security.KeyStore keystore) {}
        protected void engineInit(
                javax.net.ssl.ManagerFactoryParameters parameters) {}
        protected javax.net.ssl.TrustManager[] engineGetTrustManagers() {
            return new javax.net.ssl.TrustManager[] { new AcceptAllX509TrustManager() };
        }
```

Obrázek 22. AcceptAllTrustManagerFactory.java

```
    protected static class AcceptAllX509TrustManager implements
            javax.net.ssl.X509TrustManager {
        public void checkClientTrusted(
                java.security.cert.X509Certificate[] certificateChain,
                String authType) throws java.security.cert.CertificateException {
            report("Client authtype=" + authType);
            for (java.security.cert.X509Certificate certificate : certificateChain) {
                report("Accepting:" + certificate);
            }
        }
        public void checkServerTrusted(
                java.security.cert.X509Certificate[] certificateChain,
                String authType) throws java.security.cert.CertificateException {
            report("Server authtype=" + authType);
            for (java.security.cert.X509Certificate certificate : certificateChain) {
                report("Accepting:" + certificate);
            }
        }
        public java.security.cert.X509Certificate[] getAcceptedIssuers() {
            return new java.security.cert.X509Certificate[0];
        }
    private static void report(String string) {
           System.out.println(string);
      }
    }
```

Obrázek 23. AcceptAllX509TrustManager.java

Konfigurace kanálu JAAS kanálu telemetrie

Nakonfigurujte službu JAAS tak, aby ověřoval identitu `Jméno uživatele` odeslané klientem.

Administrátor produktu WebSphere MQ konfiguruje, které kanály MQTT vyžadují ověření klienta pomocí služby JAAS. Určete název konfigurace JAAS pro každý kanál, který má provést ověření JAAS . Kanály mohou používat stejnou konfiguraci JAAS nebo mohou používat různé konfigurace služby JAAS . Konfigurace jsou definovány v produktu *WMQData directory*`\qmgrs\`*qMgrName*`\mqxr\jaas.config`.

Soubor `jaas.config` je uspořádán podle názvu konfigurace JAAS . Pod každým názvem konfigurace je uveden seznam konfigurací přihlášení, viz Obrázek 24 na stránce 115.

JAAS poskytuje čtyři standardní přihlašovací moduly. Standardní přihlašovací moduly NT a UNIX mají omezenou hodnotu.

Modul JndiLogin
Ověřuje se proti adresářové službě konfigurované pod rozhraním JNDI (Java Naming and Directory Interface).

Krb5LoginModule
Ověřuje použití protokolů Kerberos .

NTLoginModule
Provádí ověření pomocí informací o zabezpečení NT pro aktuálního uživatele.

Modul UnixLogin
Provádí ověření pomocí informací o zabezpečení systému UNIX pro aktuálního uživatele.

Problém s používáním produktu `NTLoginModule` nebo `UnixLoginModule` je, že služba telemetrie (MQXR) je spuštěna s identitou `mqm` , nikoli s identitou kanálu MQTT. `mqm` je identita předávaná produktu `NTLoginModule` nebo `UnixLoginModule` pro ověřování, nikoli identita klienta.

Chcete-li tento problém odstranit, napište svůj vlastní přihlašovací modul nebo použijte jiné standardní přihlašovací moduly. Ukázka `JAASLoginModule.java` je dodávána spolu s produktem WebSphere MQ Telemetry. Jedná se o implementaci rozhraní produktu `javax.security.auth.spi.LoginModule` . Použijte jej k vytvoření vlastní metody ověření.

Všechny nové třídy LoginModule , které zadáte, musí být na cestě ke třídě služby telemetrie (MQXR). Neumísťujte své třídy do adresářů produktu WebSphere MQ , které jsou v cestě ke třídám. Vytvořte své vlastní adresáře a definujte celou cestu ke třídám pro službu telemetrie (MQXR).

Cestu ke třídě používanou službou telemetrie (MQXR) můžete rozšířit nastavením cesty ke třídám v souboru `service.env` . `CLASSPATH` musí být velkými písmeny, a příkaz cesty ke třídě může obsahovat pouze literály. Proměnné v proměnné CLASSPATH nelze použít; například `CLASSPATH=%CLASSPATH%` je chybná. Služba telemetrie (MQXR) nastavuje vlastní cestu ke třídám. Hodnota CLASSPATH definovaná v produktu `service.env` je přidána k ní.

Služba telemetrie (MQXR) poskytuje dvě zpětná volání, která vrací hodnoty `Jméno uživatele` a `Heslo` pro klienta připojeného k kanálu MQTT. Hodnoty `Jméno uživatele` a `Heslo` jsou nastaveny v objektu `MqttConnectOptions` . Příklad přístupu k `Username` a `Password`najdete v tématu Obrázek 25 na stránce 116 .

Příklady

Příklad konfiguračního souboru JAAS s jednou pojmenovanou konfigurací, `MQXRConfig`.

```
MQXRConfig {
  samples.JAASLoginModule required debug=true;
  //com.ibm.security.auth.module.NTLoginModule required;
  //com.ibm.security.auth.module.Krb5LoginModule required
  //                 principal=principal@your_realm
  //                 useDefaultCcache=TRUE
  //                 renewTGT=true;
  //com.sun.security.auth.module.NTLoginModule required;
  //com.sun.security.auth.module.UnixLoginModule required;
  //com.sun.security.auth.module.Krb5LoginModule required
  //                 useTicketCache="true"
  //                 ticketCache="${user.home}${/}tickets";
};
```

Obrázek 24. Ukázkový soubor `jaas.config`

Příklad modulu přihlášení JAAS kódovaného tak, aby přijímal hodnoty `Jméno uživatele` a `Heslo` poskytnuté klientem MQTT.

```
public boolean login()
    throws javax.security.auth.login.LoginException {
  javax.security.auth.callback.Callback[] callbacks =
    new javax.security.auth.callback.Callback[2];
  callbacks[0] = new javax.security.auth.callback.NameCallback("NameCallback");
  callbacks[1] = new javax.security.auth.callback.PasswordCallback(
      "PasswordCallback", false);
  try {
    callbackHandler.handle(callbacks);
    String username = ((javax.security.auth.callback.NameCallback) callbacks[0])
        .getName();
    char[] password = ((javax.security.auth.callback.PasswordCallback) callbacks[1])
        .getPassword();
    // Accept everything.
    if (true) {
      loggedIn = true;
    } else
      throw new javax.security.auth.login.FailedLoginException("Login failed");

    principal= new JAASPrincipal(username);

  } catch (java.io.IOException exception) {
    throw new javax.security.auth.login.LoginException(exception.toString());
  } catch (javax.security.auth.callback.UnsupportedCallbackException exception) {
    throw new javax.security.auth.login.LoginException(exception.toString());
  }

  return loggedIn;
}
```

Obrázek 25. Ukázková metoda `JAASLoginModule.Login()`

Koncepty programování klientů

Koncepty popsané v tomto oddílu vám pomohou porozumět klientovi Java pro verzi 3.1 produktu MQTT protocol. Koncepty doplňují dokumentaci k rozhraní API přiloženou k balíku `com.ibm.micro.client.mqttv3`.

`com.ibm.micro.client.mqttv3` contains the classes that provide the public methods for the Java implementations of the MQTT version 3.1 protocol. Balík produktu `com.ibm.micro.client.mqttv3` a doprovodné balíky, které implementují protokol pro produkt Java SE a ME, jsou dodávány s instalací produktu IBM WebSphere MQ Telemetry.

Chcete-li vyvinout a spustit klienta MQTT , musíte tyto balíky zkopírovat nebo nainstalovat na klientské zařízení. Není třeba instalovat oddělené běhové prostředí klienta.

Podmínky licencování pro klienty jsou přidruženy k serveru, ke kterému připojujete klienty.

Klient produktu Java je referenční implementací verze 3.1 produktu MQTT protocol. Můžete implementovat vlastní klienty v různých jazycích vhodných pro různé platformy zařízení. Podrobnosti naleznete v dokumentu MQ Telemetry Transport format and protocol .

Dokumentace k rozhraní API klienta pro balík `com.ibm.micro.client.mqttv3` nevyčiní žádné předpoklady ohledně serveru, ke kterému je klient připojen. Pro odkazy na dokumentaci ke klientským rozhraním API pro knihovny klienta MQTT viz Odkaz na programování klienta MQTT. Chování klienta se může mírně lišit, pokud je připojeno k různým serverům. Níže uvedené popisy popisují chování klienta při připojení ke službě telemetrie IBM WebSphere MQ (MQXR).

Aplikace Klient systému zpráv MQTT pro produkt JavaScript a webové aplikace

Programové webové aplikace a vytváření aplikací systému zpráv byly až do nedávné doby samostatnými disciplínami. Bez ohledu na to, kde se nacházejí vaše předchozí zkušenosti, existují významné výhody v používání produktu JavaScript a systému zpráv. Když kótujete aplikaci systému zpráv jako webovou aplikaci, lze ji stáhnout a spustit v libovolném prohlížeči na dobu určitou. Změníte-li aplikaci, dojde

ke stažení nejnovější verze při každém obnovení prohlížeče. Prohlížeč také vyhledává zabezpečení po zabezpečení a spolehlivý přenos zpráv.

Jak pomocí webové aplikace usnadňuje implementaci aplikace

Pokud máte zkušenosti s vývojem a implementací tradičních aplikací systému zpráv na (například) IBM WebSphere MQ, možná byste byli obeznámeni s následujícím procesem implementace:

1. Administrátor systému nainstaluje nebo vloží do knihovny klienta knihovnu.
2. Administrátor systému zajistí distribuci aplikace systému zpráv koncovým uživatelům a jejich instalaci na jejich lokální systémy.
3. Když se kód změní, administrátor systému opakuje předchozí kroky (takže správa změn je složitá).

Pokud kóete svou aplikaci systému zpráv jako webovou aplikaci, je to proces implementace:

1. Administrátor systému obsluhuje webovou aplikaci a knihovnu klienta na adrese URL.
2. Prohlížeč koncového uživatele se stáhne do webové aplikace a knihovny klienta dohromady.
3. Když se kód změní, je aktualizovaná verze vyzvedne, když se prohlížeč obnoví (takže správa změn je jednoduchá).

Proč můžete chtít používat systém zpráv přímo z prohlížeče ve vašich webových aplikacích

Máte-li zkušenosti s programováním aplikací v produktu JavaScript, měli byste zájem o informace o výhodách poskytovaných systémy zasílání zpráv, jako je například produkt IBM WebSphere MQ:

- Pokud odesíláte a přijímáte zprávy prostřednictvím systému zpráv, je tento systém odpovědný za ujištění, že jsou zprávy doručeny.
- Vzhledem k tomu, že systém zasílání zpráv po doručení vypadá, může vaše webová aplikace "spustit a zapomenout". To značně zjednodušuje logiku programování. Pokud jsou zprávy doručeny pro vás, váš kód nemusí zkontrolovat, zda tam jsou. Vaše aplikace již nepotřebuje zpracovat potvrzení o přijetí nebo uložit nedoručené zprávy a zopakovat je později.
- Systémy systému zpráv poskytují systém zpráv řízený událostmi. Vaše aplikace klienta již nemusí odeslat požadavek a poté průběžně vyzývat k odpovědi. Místo toho server zasílání zpráv odešle zprávu do vaší klientské aplikace, když dojde k zajímavé události. To také znamená, že vaše klientská aplikace bude upozorněna, jakmile se událost stane, spíše než čekat, až se aplikace příště dotazuje serveru.
- Systém zpráv řízený událostmi také masivně snižuje zátěž na zařízení, které hostuje aplikaci klienta, síťový provoz mezi prohlížečem a serverem systému zpráv a zátěž na serveru systému zpráv. Tato problematika se stále více stává, čím dál více systémů běží na mobilních zařízeních a připojuje se přes bezdrátové sítě.

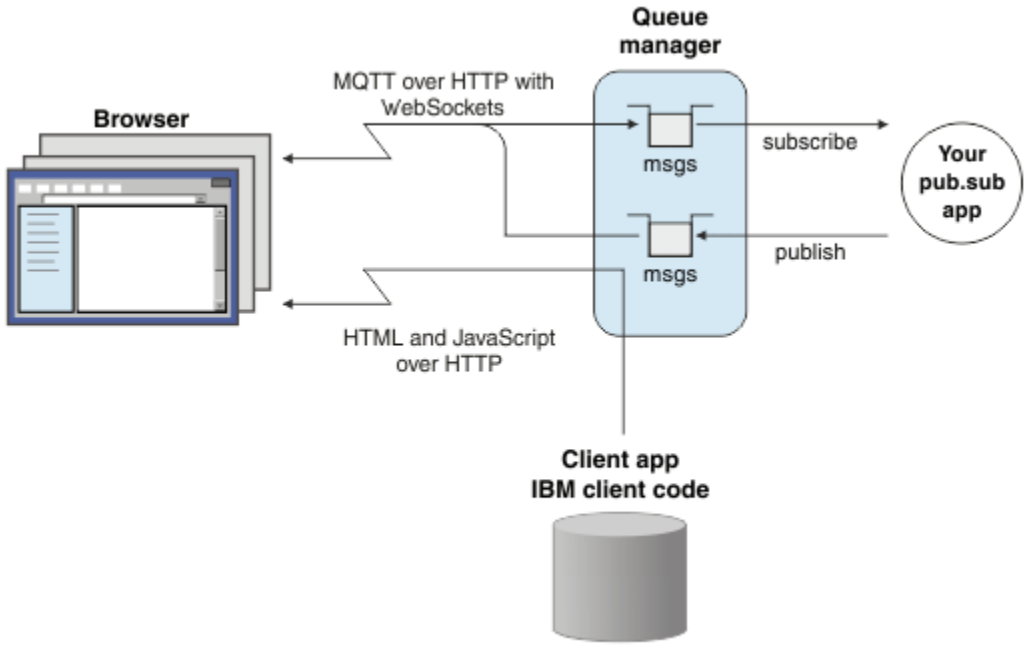
Jak se kousky vejdou dohromady

Produkt Klient systému zpráv MQTT pro produkt JavaScript obsahuje knihovnu klienta a ukázkovou webovou aplikaci, která používá knihovnu. Vykódujete svou vlastní webovou aplikaci, která používá knihovnu. Webová aplikace a knihovna klienta jsou poté zpřístupněny na zvolené adrese URL, například správcem front MQ (jako v následujícím diagramu) nebo prostřednictvím aplikačního serveru. Prohlížeč se stáhne do webové aplikace a knihovny klienta a webová aplikace pak použije prohlížeč k připojení a výměně zpráv se serverem MQTT, jako je IBM WebSphere MQ Telemetry nebo IBM MessageSight.

Jedná se o toky:

1. Každá instance prohlížeče aktualizuje své připojení k adrese URL, na které je webová aplikace k dispozici, a do prohlížeče je načtena jedna verze webové aplikace a klientské knihovny.
2. Webová aplikace se připojuje ke správci front pomocí protokolu MQTT v rámci produktu WebSocket protocola odebírá se k odběru tématu, které je předmětem zájmu.

3. Správce front používá stejné připojení k odesílání zpráv, které odpovídají odběru, zpět do webové aplikace.



Obrázek 26. Použití produktu Klient systému zpráv MQTT pro produkt JavaScript se systémem zpráv pro publikování a odběr

Webová aplikace obsahuje aplikační logiku a adresu URL serveru MQTT. Po otevření v prohlížeči se aplikace připojí k serveru MQTT, vytvoří odběry, které potřebuje, a poté čeká na příjem výstrah, které vyvolávají události, a budou na nich pracovat.

Webová aplikace se připojuje pomocí protokolu MQTT jako přenosového protokolu, který běží nad WebSockets. Většina moderních prohlížečů může vytvářet WebSockets připojení. Pomocí produktu WebSocketsmůže webová aplikace předávat zprávy prostřednictvím bran firewall, které přijímají HTTP a WebSocket protocol, a mohou odesílat pakety dat (známé jako "rámce") stejně jako použití protokolu TCP over IP.

Když je zpráva odeslaná webovou aplikací doručena na server MQTT, zobrazí se aplikace na straně serveru jako zpráva. Zpráva netuší, že zpráva pochází z prohlížeče.

Administrace a řízení serveru MQTT

Server MQTT zpracovává složitost systému zpráv na straně serveru. Zajišťuje doručování zpráv, které obdrží od webové aplikace, a je hostitelem aplikace pro publikování a odběr, která odpovídá na webovou aplikaci. Pro jakýkoli server MQTT je třeba provést následující kroky:

- Vytvořte server.
- Vyberte si port.
- Definujte nový kanál MQTT.
- Nakonfigurujte webovou aplikaci klienta tak, aby se připojovala ke zvolenému portu přes nový kanál MQTT.

Musíte také posloužit ke spuštění spustitelného souboru webové aplikace JavaScript v prohlížeči. Používáte-li produkt IBM WebSphere MQ Telemetry, server MQTT pro vás standardně používá stejný kanál MQTT, který webová aplikace používá k připojení k serveru MQTT. Pokud zkoušíte protokol MQTT, může vám to pomoci urychlit a rychle začít pracovat. Pro provozní použití, zejména v prostředí s vysokou

propustností, můžete raději obsluhovat spustitelný soubor webové aplikace JavaScript na samostatném kanálu pomocí vyhrazeného aplikačního serveru, jako je například produkt WebSphere Application Server.

Poznámka: Protože je navržen pro prostředí s vysokou propustností, IBM MessageSight očekává, že to provedete.

Používáte-li například produkt IBM WebSphere MQ Telemetry, pomocí průvodce **Nový kanál telemetrie** produktu IBM WebSphere MQ Explorer postupujte takto:

1. Vytvořte server.
2. Vyberte port (1883 ve výchozím nastavení).
3. Definujte nový kanál MQTT.
4. Nakonfigurujte webovou aplikaci klienta tak, aby se připojovala ke zvolenému portu přes nový kanál MQTT.

Spustitelný soubor webové aplikace JavaScript je (volitelně) také poskytován prostřednictvím správce front na stejném kanálu. Aby to bylo možné provést, musí správce front podporovat protokol MQTT i HTTP. Pokud již máte správce front, který je nakonfigurovaný pro produkt MQTT, můžete použít nástroj příkazového řádku MQSC a změnit protokol v definici kanálu tak, aby podporoval protokol MQTT i HTTP. Viz ALTER CHANNEL.

Webová aplikace a knihovna klienta Klient systému zpráv MQTT pro produkt JavaScript jsou uloženy na disku ve struktuře, která je definována aplikačním serverem nebo správcem front. Používáte-li produkt IBM WebSphere MQ Telemetry, webová aplikace a knihovna klienta jsou uloženy v následující adresářové struktuře:

```
MQINSTALL
  |
  mqxr
  |  |
  |  SDK
  |    |
  |    WebContent
  |       sample web app (the "Web Messaging Utility" sample HTML pages)
  |       |
  |       WebSocket
  |          IBM client library (the messaging client JavaScript classes)
  |
  qmgrs
     |
     qmgr_name
        |
        mqxr
           |
           WebContent
              |
              your_client_app (your own JavaScript pages)
```

Ukázková webová aplikace a knihovna klienta jsou uloženy v adresáři *MQINSTALL*/mqxr/SDK/WebContent . Materiál v tomto adresáři je obsluhován všemi správci front. Pokud nechcete, aby vaši uživatelé viděli a používali všechen tento materiál, měli byste vytvořit svou vlastní verzi aplikace. Chcete-li tuto aplikaci nebo vaši vlastní náhradní aplikaci zpřístupnit na specifických správcích front, umístěte aplikaci do adresáře *MQINSTALL*/qmgrs/*qmgr_name*/mqxr/WebContent . Chcete-li vybrat aplikaci a přidružené třídy produktu JavaScript , které budou sloužit v adrese URL, správce front bude nejprve hledat ve svém vlastním adresáři WebContent a poté v globálním adresáři WebContent . V předchozím ukázkovém adresářovém stromu správce front obsluhuje *your_client_app* a globální kopie tříd produktu JavaScript .

Chcete-li správce front zastavit obsluhování spustitelných souborů webové aplikace nebo upravit místo, kde správce front hledá spustitelné soubory, nakonfigurujte vlastnost **webcontentpath** a přidejte ji do souboru mqxr.properties . Viz vlastnosti MQXR.

Související pojmy

“Jak naprogramovat aplikace systému zpráv v produktu JavaScript” na stránce 120

Související úlohy

“Připojení Klient systému zpráv MQTT pro produkt JavaScript přes SSL a WebSockets” na stránce 77

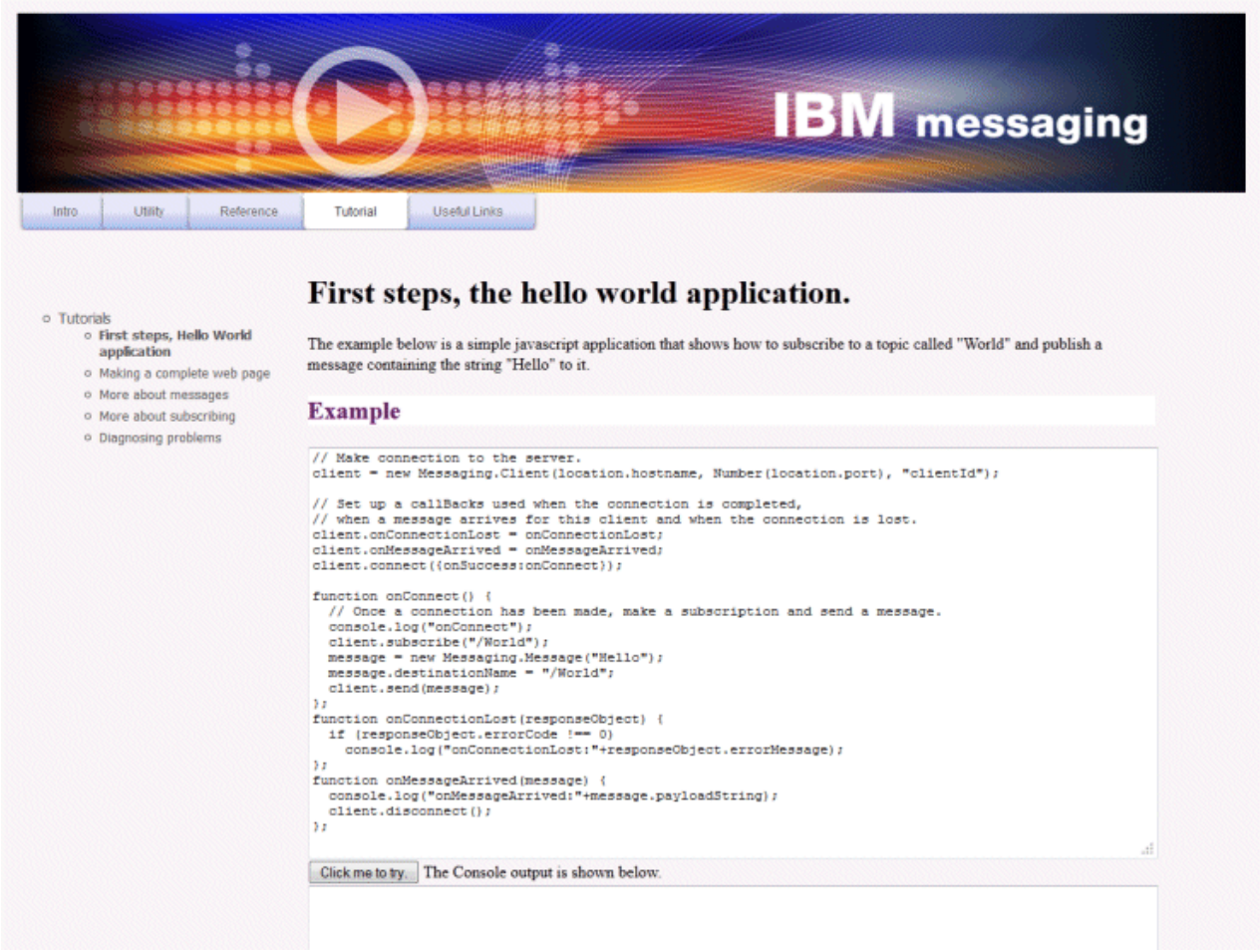
Webovou aplikaci bezpečně připojte k produktu IBM WebSphere MQ pomocí ukázkových HTML stránek Klient systému zpráv MQTT pro produkt JavaScript s SSL a WebSocket protocol.

“Začínáme s produktem Klient systému zpráv MQTT pro produkt JavaScript” na stránce 23
Můžete začít pracovat se serverem Klient systému zpráv MQTT pro produkt JavaScript zobrazením ukázkové domovské stránky klienta systému zpráv a procházením prostředků, na které se odkazuje. Chcete-li zobrazit tuto domovskou stránku, nakonfigurujte server MQTT tak, aby přijímal připojení ze serveru Ukázkové stránky klienta systému zpráv MQTT JavaScript, pak napíšete adresu URL, kterou jste nakonfigurovali na serveru do webového prohlížeče. Produkt Klient systému zpráv MQTT pro produkt JavaScript se automaticky spustí na vašem zařízení a zobrazí se ukázková domovská stránka klienta systému zpráv. Tato stránka obsahuje odkazy na obslužné programy, dokumentaci k programovacímu rozhraní, výukový program a další užitečné informace.

Jak naprogramovat aplikace systému zpráv v produktu JavaScript

Produkt Klient systému zpráv MQTT pro produkt JavaScript obsahuje výukový program, který ukazuje, jak vytvořit jednoduchou webovou aplikaci publikování a odběru. Prozkoumáním aplikačního kódu "První kroky, Ahoj světe" můžete získat základní informace o mechanice programování webových aplikací pro zasílání zpráv.

Pokud se vaše dosavadní zkušenosti týkají především vývoje a implementace tradičních aplikací pro zasílání zpráv, můžete také najít sekci “Tipy pro kódování JavaScript” na stránce 121 užitečná. Jste-li zkušenější vývojář produktu JavaScript , který je novým systémem zpráv, najdete krátký úvod do klíčových konceptů systému zpráv v sekci “Základy systému zpráv” na stránce 123 .



Tipy pro kódování JavaScript

Pokud jste zvyklí na vývoj aplikací systému zpráv, ale nové pro webové aplikace, mohou vám být užitečné následující rady:

Zalamování kódu pro každou událost ve zpětném volání onSuccess

Když kódujete aplikaci systému zpráv, v následujícím pořadí nakóte následující události:

1. connect
2. odebírat
3. publikování
4. přijmout zprávu

Rozhraní API produktu Klient systému zpráv MQTT pro produkt JavaScript je plně asynchronní, což znamená, že se vaše aplikační vlákno při čekání na volání jako spojení nebo přihlášení k odběru neprojeví. Místo toho tyto hovory volají po svém dokončení voláním zpětného volání `onSuccess` nebo `onFailure` . Chcete-li se ujistit, že každá událost byla dokončena před spuštěním další události, je třeba kód zabalit pro každou událost ve zpětném volání `onSuccess` . Například aplikace JavaScript se může vrátit před vytvořením připojení, než se připojení vytvoří. Abyste se ujistili, že se připojení stalo, než se přihlásíte k odběru, musíte vložit kód odběru do zpětného volání `onSuccess` pro připojení.

Tento přístup používá aplikační kód "První kroky, Ahoj světe".

Vnoření kódu aplikace v rámci markupu HTML

Zde je příklad stránky JavaScript :

Zde je uveden zdroj předchozí stránky, který ukazuje, jak je kód aplikace vložen v rámci markupu HTML:

```
<!DOCTYPE html>


<head>
  <script type="text/javascript" src="../WebSocket/mqttws31.js"></script>

  <script type="text/javascript">

  var client;
  var form = document.getElementById("tutorial");

  function doConnect() {
      client = new Messaging.Client("whistler1.hursley.ibm.com", 1883, "ClientId");
      client.onConnect = onConnect;
      client.onMessageArrived = onMessageArrived;
      client.onConnectionLost = onConnectionLost;
      client.connect({onSuccess:onConnect});
  }
```

```
  function doSubscribe() {
      client.subscribe("/World");
  }

  function doSend() {
      message = new Messaging.Message("Hello");
      message.destinationName = "/World";
      client.send(message);
  }

  function doDisconnect() {
      client.disconnect();
  }

  // Web Messaging API callbacks

  function onConnect() {
      var form = document.getElementById("example");
      form.connected.checked= true;
  }

  function onConnectionLost(responseObject) {
      var form = document.getElementById("example");
      form.connected.checked= false;
      if (responseObject.errorCode !== 0)
          alert(client.clientId+"\n"+responseObject.errorCode);
  }

  function onMessageArrived(message) {
      var form = document.getElementById("example");
      form.receiveMsg.value = message.payloadString;
  }

  </script>
</head>

<body>
  <h1>Example Web Messaging web page.</h1>
  <form id="example">
  <fieldset>
  <legend id="Connect" > Connect </legend>
    Make a connection to the server, and set up a call back used if a
    message arrives for this client.
    <br>
    <input type="button" value="Connect" onClick="doConnect(this.form)" name="Connect"/>
    <input type="checkbox" name="connected" disabled="disabled"/>
  </fieldset>

  <fieldset>
  <legend id="Subscribe" > Subscribe </legend>
  Make a subscription to topic "/World".
  <br> <input type="button" value="Subscribe" onClick="doSubscribe(this.form)"/>
  </fieldset>

  <fieldset>
  <legend id="Send" > Send </legend>
    Create a Message object containing the word "Hello" and then publish it at
    the server.
    <br>
    <input type="button" value="Send" onClick="doSend(this.form)"/>
  </fieldset>

  <fieldset>
  <legend id="Receive" > Receive </legend>
    A copy of the published Message is received in the callback we created earlier.
  <textarea name="receiveMsg" rows="1" cols="40" disabled="disabled"></textarea>
  </fieldset>

  <fieldset>
  <legend id="Disconnect" > Disconnect </legend>
    Now disconnect this client from the server.
  <br> <input type="button" value="Disconnect" onClick="doDisconnect()"/>
  </fieldset>
  </form>
</body>
</html>
```

Základy systému zpráv

Zde jsou uvedeny některé informace o systému zpráv na pozadí pro vývojáře webové aplikace, kteří jsou nové pro zasílání zpráv:

Systém zpráv s asynchronním a požárem a zapomenutí.

Protokol MQTT podporuje zajištěné doručování a přenosy typu fire-and-forget. V protokolu je doručování zpráv asynchronní: aplikace předá zprávu do rozhraní API klienta a neprovede žádnou další akci, aby bylo jisté, že je zpráva doručena. Tento přístup je znám jako *fire-and-forget*. Je-li odezva k dispozici, je automaticky odeslána do aplikace.

Asynchronní doručení uvolní aplikaci z jakéhokoli připojení serveru a čeká na zprávy. Model interakce je jako e-mail, ale optimalizovaný pro programování aplikací.

Viz také část "Protokol MQTT" produktu “Úvod do produktu MQTT” na stránce 5

Přehled systému zpráv publikování a odběru.

Poskytovatel informací se nazývá *vydavatel*. Vydavatel poskytuje informace o předmětu, aniž by bylo nutné znát informace o aplikacích, které mají o tyto informace zájem. Vydavatel si vybere *téma*, které je kontejnerem pro zprávy ve specifickém předmětu. Vydavatel potom vygeneruje každou informaci pro subjekt jako zprávu nazvanou *publikace*a odešle ji do přidruženého tématu.

Spotřebitel informací se nazývá *odběratel*. Odběratel vytvoří *odběr* na téma, které se o něj zajímá. Při odeslání nové zprávy na téma se zpráva předá všem odběratelům v rámci daného tématu. Odběratelé mohou vytvářet více odběrů a mohou přijímat informace od mnoha vydavatelů.

Viz také Úvod do systému zpráv typu publikování/odběr produktu IBM WebSphere MQ

Způsob, jakým se odběry a témata shodují.

Používáte-li server IBM WebSphere MQ jako server MQTT, je třeba porozumět tomu, jak produkt IBM WebSphere MQ určuje témata. V produktu IBM WebSphere MQvydavatel vytvoří zprávu a publikuje ji spolu s řetězcem tématu, který nejlépe odpovídá předmětu publikování. Chcete-li přijímat publikování, odběratel vytvoří odběr s použitím řetězce tématu shodujícího se na vzorek, který bude vybírat témata publikování. Správce front doručuje publikování odběratelům, kteří mají odběry, které odpovídají tématu publikování, a jsou autorizováni k přijetí těchto publikování.

Typicky jsou předměty uspořádány hierarchicky, do stromu témat pomocí znaku ' /' pro vytvoření dílčích témat v řetězci tématu. Témata jsou uzly ve stromu témat. Témata mohou být listové uzly bez dalších dílčích témat nebo zprostředkujících uzlů s dílčími tématy. Odběratelé mohou v daném okamžiku používat zástupné znaky k odběru více než jednoho tématu. Například odběr produktu `/sport/tennis` dostane pouze zprávy odeslané do dílčího tématu tenisu, zatímco odběr produktu `/sport/#` dostane zprávy odeslané do libovolného dílčího tématu produktu `/sport`.

Viz také Témata, Stromy témata Schémata zástupných znaků.

Související pojmy

“Aplikace Klient systému zpráv MQTT pro produkt JavaScript a webové aplikace” na stránce 116

Související úlohy

“Připojení Klient systému zpráv MQTT pro produkt JavaScript přes SSL a WebSockets” na stránce 77
Webovou aplikaci bezpečně připojte k produktu IBM WebSphere MQ pomocí ukázkových HTML stránek Klient systému zpráv MQTT pro produkt JavaScript s SSL a WebSocket protocol.

“Začínáme s produktem Klient systému zpráv MQTT pro produkt JavaScript” na stránce 23
Můžete začít pracovat se serverem Klient systému zpráv MQTT pro produkt JavaScript zobrazením ukázkové domovské stránky klienta systému zpráv a procházením prostředků, na které se odkazuje. Chcete-li zobrazit tuto domovskou stránku, nakonfigurujte server MQTT tak, aby přijímal připojení ze serveru Ukázkové stránky klienta systému zpráv MQTT JavaScript, pak napíšete adresu URL, kterou jste nakonfigurovali na serveru do webového prohlížeče. Produkt Klient systému zpráv MQTT pro produkt JavaScript se automaticky spustí na vašem zařízení a zobrazí se ukázková domovská stránka klienta systému zpráv. Tato stránka obsahuje odkazy na obslužné programy, dokumentaci k programovacímu rozhraní, výukový program a další užitečné informace.

Zpětná volání a synchronizace v aplikacích klienta MQTT

Programovací model klienta produktu MQTT rozsáhle používá podprocesy. Podprocesy oddělili aplikaci klienta MQTT , stejně jako mohou, z prodlev při přenosu zpráv ze serveru a ze serveru. Publikování, tokeny doručení a ztracené události připojení se doručují do metod ve třídě zpětného volání, která implementuje `MqttCallback`.

Zpětná volání

Rozhraní `MqttCallback` má tři metody zpětného volání, viz příklad implementace v souboru Callback.java.

`connectionLost(java.lang.Throwable cause)`

`connectionLost` je volán, když komunikační chyba vede k uvolnění spojení. Nazývá se také tehdy, když server po navázání spojení přeruší spojení v důsledku chyby na serveru. Chyby serveru se protokolují do protokolu chyb správce front. Server zruší spojení s klientem a klient zavolá `MqttCallback.connectionLost`.

Jediné vzdálené chyby vyvolané jako výjimky ze stejného podprocesu jako aplikace klienta jsou výjimky z produktu `MqttClient.connect`. Chyby zjištěné serverem po zavedení připojení jsou nahlášeny zpět do metody zpětného volání `MqttCallback.connectionLost` jako `throwables`.

Typické chyby serveru, které vedou k chybě `connectionLost` , jsou chyby autorizace. Server telemetrie se například pokusí publikovat na téma jménem klienta, který není autorizován k publikování v rámci daného tématu. Vše, co má za následek vrácení kódu podmínky produktu `MQCC_FAIL` do serveru telemetrie, může vést k tomu, že připojení bude zrušeno.

`deliveryComplete(MqttDeliveryToken token)`

`deliveryComplete` je volán klientem MQTT , aby předal token doručení zpět aplikaci klienta, viz “Tokeny doručení” na stránce 127. Při použití tokenu doručení může zpětné volání přistupovat k publikované zprávě s použitím metody `token.getMessage`.

Když zpětné volání aplikace vrátí řízení do klienta MQTT po volání metody `deliveryComplete` , doručení je dokončeno. Dokud nebude dokončeno doručení, zprávy s produktem `QoS` 1 nebo 2 jsou zachovány podle třídy perzistence.

Volání do produktu `deliveryComplete` je bodem synchronizace mezi aplikací a třídou perzistence. Metoda `deliveryComplete` se nikdy nevolá dvakrát pro stejnou zprávu.

Když se zpětné volání aplikace vrátí z produktu `deliveryComplete` na klienta MQTT , klient zavolá příkaz `MqttClientPersistence.remove` pro zprávy s hodnotou `QoS` 1 nebo 2. Příkaz `MqttClientPersistence.remove` odstraní lokálně uloženou kopii publikované zprávy.

Z pohledu zpracování transakce je volání na produkt `deliveryComplete` jednofázovou transakcí, která potvrzuje doručení. Pokud zpracování selže během zpětného volání, při restartu klienta `MqttClientPersistence.remove` se znovu volá a odstraní lokální kopie publikované zprávy. Zpětné volání se nevolá znovu. Pokud používáte zpětné volání k uložení protokolu doručených zpráv, nemůžete synchronizovat protokol s klientem MQTT . Chcete-li uložit protokol spolehlivě, aktualizujte protokol ve třídě `MqttClientPersistence` .

Na token doručení a na zprávu odkazuje hlavní aplikační podproces a klient produktu MQTT . Klient MQTT zruší odkaz na objekt `MqttMessage` při dokončení doručení a objekt tokenu doručení, když se klient odpojí. Objekt `MqttMessage` může být po dokončení doručení vyčištěn do paměti, pokud na to klientská aplikace zruší odkaz. Po odpojení relace může být token doručení odebrán.

Po publikování zprávy můžete získat atributy `MqttDeliveryToken` a `MqttMessage` po publikování zprávy. Pokusíte-li se nastavit libovolný atribut `MqttMessage` poté, co byla zpráva publikována, výsledek není definován.

Klient MQTT pokračuje v zpracování potvrzení doručení, pokud se klient znovu připojí k předchozí relaci se stejným parametrem `ClientIdentifier`; viz “Vyčistit relace” na stránce 126. Aplikace klienta MQTT musí nastavit `MqttClient.CleanSession` na `false` pro předchozí relaci a nastavit ji na `false` v nové relaci. Klient produktu MQTT vytváří nové tokeny doručení a objekty zpráv v nové relaci pro nevyřízené doručení. Zotavuje objekty pomocí třídy `MqttClientPersistence` . Pokud má klient aplikace stále odkazy na staré tokeny doručení

a zprávy, vyhodnoťte je. Zpětné volání aplikace se volá v nové relaci pro všechny doručení zahájené v předchozí relaci a dokončené v této relaci.

Zpětné volání aplikace se volá po připojení klienta aplikace, když je dokončeno nevyřízené doručení. Než se klient aplikace připojí, může nevyřízené doručení načíst pomocí metody `MqttClient.getPendingDeliveryTokens` .

Všimněte si, že aplikace klienta původně vytvořila objekt zprávy, který je publikován, a jeho bajtové pole informačního obsahu. Klient MQTT odkazuje na tyto objekty. Objekt zprávy vrácený tokenem doručení v metodě `token.getMessage` nemusí nutně znamenat stejný objekt zprávy vytvořený klientem. Pokud nová instance klienta MQTT znovu vytvoří token doručení, třída `MqttClientPersistence` znovu vytvoří objekt `MqttMessage` . Pro konzistenci `token.getMessage` vrátí hodnotu `null` , pokud `token.isCompleted` je `true`, bez ohledu na to, zda byl objekt zprávy vytvořen klientem aplikace nebo třídou `MqttClientPersistence` .

`messageArrived(MqttTopic topic, MqttMessage message)`

`messageArrived` je volán, když je doručena publikace pro klienta, který odpovídá tématu odběru. `topic` je téma publikace, nikoli filtr odběru. Mohou se lišit, pokud filtr obsahuje zástupné znaky.

Pokud se téma shoduje s více odběry vytvořenými klientem, obdrží klient více kopií publikování. Pokud klient publikuje do tématu, které se také přihlásí k odběru, obdrží kopii své vlastní publikace.

Je-li zpráva odeslána s `QoS` z `1` nebo `2`, zpráva je uložena třídou `MqttClientPersistence` před voláním klienta MQTT `messageArrived`. Příkaz `messageArrived` se chová jako `deliveryComplete`: volá se pouze jednou pro publikování a lokální kopie této publikace je odstraněna produktem `MqttClientPersistence.remove` , když se příkaz `messageArrived` vrátí klientovi MQTT . Klient MQTT zruší své odkazy na téma a zprávu, když se `messageArrived` vrátí klientovi MQTT . Objekty tématu a zprávy jsou shromažďovány v rámci uvolňování paměti, pokud klient aplikace neuchoval odkaz na objekty.

Zpětná synchronizace, rozdělování na podprocesy a synchronizace aplikací klienta

Klient MQTT volá metodu zpětného volání na samostatný podproces do hlavního podprocesu aplikace. Klientská aplikace nevytváří podproces pro zpětné volání, je vytvořena klientem MQTT .

Klient MQTT synchronizuje metody zpětného volání. V daném okamžiku je spuštěna pouze jedna instance metody zpětného volání. Synchronizace umožňuje snadno aktualizovat objekt, který obsahuje informace o tom, které publikace byly doručeny. Jedna instance serveru `MqttCallback.deliveryComplete` se spustí v daném okamžiku, a proto je bezpečné aktualizovat záznam, aniž by došlo k další synchronizaci. Je také tomu tak, že v daném okamžiku dorazí pouze jedna publikace. Váš kód v metodě `messageArrived` může aktualizovat objekt, aniž by se synchronizoval. Pokud odkazujete na záznam nebo objekt, který se právě aktualizuje, v jiném podprocesu synchronizujte záznam nebo objekt.

Token doručení poskytuje mechanizmus synchronizace mezi hlavním podprocesem aplikace a doručením publikace. Metoda `token.waitForCompletion` čeká, dokud nebude dokončeno doručení určité publikace, nebo dokud nevyprší volitelný časový limit. Produkt `token.waitForCompletion` můžete použít v několika jednoduchých způsobech ke zpracování jedné publikace v daném okamžiku:

1. Chcete-li klienta aplikace pozastavit až do dokončení doručení publikace, přečtěte si téma PubSync.java.
2. Proveďte synchronizaci s metodou `MqttCallback.deliveryComplete` . Pouze když se `MqttCallback.deliveryComplete` vrátí na klienta MQTT , obnoví se `token.waitForCompletion` . Pomocí tohoto mechanizmu můžete synchronizovat spouštěný kód v produktu `MqttCallback.deliveryComplete` před spuštěním kódu v hlavním podprocesu aplikace.

Co když jste chtěli publikovat bez čekání na doručení každé publikace, ale chcete potvrdit, kdy byly všechny publikace doručeny? Pokud publikujete na jednom podprocesu, poslední publikování, které má být odesláno, je také poslední, které má být doručeno.

Synchronizace požadavků odeslaných na server

Tabulka 4. Chování synchronizace metod, které vedou k požadavkům na server.

Tato tabulka uvádí metody v klientu Java produktu MQTT , který odesílá požadavek na server. Pro každou metodu tabulka popisuje podmínky, za kterých metoda buď čeká, nebo vrátí, a jak dlouho bude metoda čekat.

Metoda	Synchronizace	Interval časového limitu
`MqttClient.Connect`	Čeká na navázání spojení se serverem.	30 sekund při výchozím nastavení, nebo podle nastavení parametru.
`MqttClient.Disconnect`	Čeká na dokončení práce klienta MQTT a jeho odpojení od relace TCP/IP.	30 sekund při výchozím nastavení, nebo podle nastavení parametru.
`MqttClient.Subscribe`	Čeká na dokončení požadavku na odběr.	30 sekund při výchozím nastavení, nebo podle nastavení parametru.
`MqttClient.UnSubscribe`	Čeká na dokončení požadavku na zrušení odběru.	30 sekund při výchozím nastavení, nebo podle nastavení parametru.
`MqttClient.Publish`	Po předání požadavku klientovi produktu MQTT se okamžitě vrátí k podprocesu aplikace.	Není.
`MqttDeliveryToken.waitForCompletion`	Čeká na vrácení doručovacího tokenu.	Neomezené při výchozím nastavení nebo hodnota nastavená parametrem.

Vyčistit relace

Klient MQTT a služba telemetrie (MQXR) udržují informace o stavu relace. Informace o stavu se používají k zajištění "alespoň jednou" a "přesně jednou" doručení a "přesně jednou" na příjmu publikací. Stav relace také zahrnuje odběry vytvořené klientem MQTT. Můžete zvolit spuštění klienta MQTT s nebo bez udržování informací o stavu mezi relacemi. Změňte režim čisté relace nastavením `MqttConnectOptions.cleanSession` před připojením.

Připojíte-li klientskou aplikaci MQTT pomocí metody `MqttClient.connect` , klient identifikuje připojení pomocí identifikátoru klienta a adresy serveru. Server kontroluje, zda byly informace o relaci uloženy z předchozího připojení k serveru. Pokud předchozí relace stále existuje a `cleanSession=true`, pak jsou předchozí informace o relaci na klientovi a serveru vymazány. Je-li `cleanSession=false` předchozí relace obnovena. Neexistuje-li žádná předchozí relace, bude spuštěna nová relace.

Poznámka: Administrátor produktu WebSphere MQ může vynutit zavření otevřené relace a odstranění všech informací o relaci. Pokud klient znovu otevře relaci s produktem `cleanSession=false`, spustí se nová relace.

Publikace

Pokud použijete výchozí `MqttConnectOptions`nebo nastavíte `MqttConnectOptions.cleanSession` na `true` před připojením klienta, všechny nevyřízené doručení publikování pro klienta se odeberou, když se klient připojí.

Nastavení čisté relace nemá žádný vliv na publikace odeslané s produktem QoS=0. Pro QoS=1 a QoS=2může použití `cleanSession=true` vést ke ztrátě publikování.

Odběry

Pokud použijete výchozí `MqttConnectOptions`nebo nastavíte `MqttConnectOptions.cleanSession` na `true` před připojením klienta, všechny staré odběry klienta se odeberou, když se klient připojí. Všechny nové odběry, které klient během relace vytvoří, jsou při odpojení odebrány.

Pokud před připojením nastavíte `MqttConnectOptions.cleanSession` na `false` , všechny odběry vytvořené klientem budou přidány ke všem odběrům, které existovaly pro klienta dříve, než se připojí. Všechny odběry zůstávají aktivní, když se klient odpojí.

Dalším přístupem k pochopení, jak atribut `cleanSession` ovlivňuje odběry, je považovat jej za modální atribut. Ve svém výchozím režimu, `cleanSession=true`, vytváří klient odběry a přijímá publikace pouze v rámci dané relace. V alternativním režimu, `cleanSession=false`, jsou odběry trvalé. Klient se může připojit a odpojit a jeho odběry zůstanou aktivní. Když se klient znovu připojí, přijme všechny nedoručené publikace. V době připojení může upravit sadu odběrů, které jsou jeho jménem aktivní.

Před připojením musíte nastavit režim `cleanSession` ; režim trvá pro celou relaci. Chcete-li změnit jeho nastavení, je třeba klienta odpojit a znovu připojit. Pokud změníte režimy z použití `cleanSession=false` na `cleanSession=true`, všechny předchozí odběry pro klienta a všechny publikace, které nebyly přijaty, budou vyřazeny.

Identifikátor klienta

Identifikátor klienta je 23bajtový řetězec, který identifikuje klienta MQTT . Identifikátor klienta musí být jedinečný pro všechny klienty, kteří se připojují k serveru, a nesmí být stejný jako název správce front na serveru. V rámci těchto omezení budete moci používat libovolný identifikační řetězec. Je důležité mít proceduru pro přidělování identifikátorů klienta a prostředků ke konfiguraci klienta se zvoleným identifikátorem.

Identifikátor klienta se používá při administraci systému MQTT . S potencionálně stovkami tisíc klientů, kteří mají spravovat, musíte být schopni rychle identifikovat konkrétního klienta. Předpokládejme například, že zařízení má nefunkčnost, a vy budete informováni o tom, že zákazník zazvoní na help desk. Jak zákazník identifikuje zařízení a jak korelovat s identifikací na serveru, který je obvykle připojen ke klientovi? Je třeba se poradit s databází, která mapuje každé zařízení na identifikátor klienta a na server? Identifikujete název zařízení, ke kterému serveru je připojen? Když procházíte přes připojení klienta MQTT , každé připojení je označeno identifikátorem klienta. Potřebujete vyhledat tabulku, chcete-li mapovat identifikátor klienta na fyzické zařízení?

Identifikujete identifikátor klienta určité zařízení, uživatele nebo aplikaci spuštěnou na klientovi? Pokud zákazník nahradí vadné zařízení novým zařízením, má nové zařízení stejný identifikátor jako staré zařízení? Přidělíte nový identifikátor? Změníte-li fyzické zařízení, ale zachovejte stejný identifikátor, význačné publikace a aktivní odběry se automaticky přenesou na nové zařízení.

Jak se ujistíte, že identifikátory klientů jsou jedinečné? Stejně jako systém pro generování jedinečných identifikátorů, musíte mít spolehlivý proces pro nastavení identifikátoru na klientovi. Možná klientské zařízení je "black-box", bez uživatelského rozhraní. Vyráběte zařízení s identifikátorem klienta-například pomocí jeho MAC adresy? Nebo máte instalaci softwaru a proces konfigurace, který konfiguruje zařízení před aktivací zařízení?

Můžete vytvořit identifikátor klienta z 48bitové adresy MAC zařízení, abyste zachovali identifikátor krátký a jedinečný. Pokud velikost přenosu není kritickým problémem, můžete použít zbývajících 17 bajtů, abyste mohli snáze spravovat adresu.

Tokeny doručení

Když klient publikuje na téma nový token doručení, dojde k vytvoření nového tokenu doručení. Použijte token doručení k monitorování doručení publikování nebo k blokování klientské aplikace, dokud nebude doručení dokončeno.

Token je objekt `MqttDeliveryToken` . Vytvoří se voláním metody MqttTopic.publish () a zachová ji klient produktu MQTT , dokud nebude relace klienta odpojena a doručení je dokončeno.

Normální použití tokenu je zkontrolovat, zda je doručení dokončeno. Zablokujte klientskou aplikaci, dokud nebude dodávka dokončena, a to pomocí vráceného tokenu pro volání `token.waitForCompletion`. Případně poskytněte obslužnou rutinu `MqttCallBack` . Pokud klient produktu MQTT přijal všechna potvrzení, která očekává jako část doručení publikace, zavolá příkaz `MqttCallBack.deliveryComplete` předáním tokenu doručení jako parametru.

Dokud nebude dodávka dokončena, můžete ji zkontrolovat pomocí vráceného tokenu doručení voláním produktu `token.getMessage`.

Dokončené dodávky

Dokončení dodávek je asynchronní a závisí na kvalitě služby přidružené k publikování.

Nejvíce jednou

`QoS=0`
Doručení je dokončeno okamžitě po návratu z produktu `MqttTopic.publish`. `MqttCallback.deliveryComplete` se volá okamžitě.

Nejméně jednou

`QoS=1`
Doručení je dokončeno, když bylo od správce front přijato potvrzení o publikování. `MqttCallback.deliveryComplete` se volá, když je přijato potvrzení. Zpráva může být doručena více než jednou dříve, než se zavolá `MqttCallback.deliveryComplete` , pokud jsou komunikace pomalé nebo nespolehlivé.

Přesně jednou

`QoS=2`
Doručení je dokončeno, když klient obdrží zprávu o dokončení, že publikování bylo publikováno odběratelům. `MqttCallback.deliveryComplete` se volá, jakmile se přijme zpráva o publikování. Nečeká na zprávu o dokončení.

Za výjimečných okolností se vaše klientská aplikace nemusí vrátit ke klientovi MQTT z produktu `MqttCallback.deliveryComplete` normálně. Víte, že dodávka byla dokončena, protože byl volán `MqttCallback.deliveryComplete` . Pokud klient restartuje stejnou relaci, produkt `MqttCallback.deliveryComplete` se znovu nezavolá.

Neúplné dodávky

Pokud není doručení dokončeno po odpojení relace klienta, můžete klienta znovu připojit a dokončit doručení. Doručování zprávy můžete dokončit pouze v případě, že byla zpráva publikována v relaci s atributem `MqttConnectionOptions` nastaveným na hodnotu `false`.

Vytvořte klienta pomocí stejného identifikátoru klienta a adresy serveru a potom se připojte znovu, nastavte atribut `cleanSession` `MqttConnectionOptions` na hodnotu `false` znovu. Pokud jste nastavili `cleanSession` na `true`, nevyřízené tokeny doručení budou zahozeny.

Můžete zkontrolovat, zda neexistují nějaké nevyřízené doručení voláním produktu `MqttClient.getPendingDeliveryTokens`. Před připojením klienta můžete volat `MqttClient.getPendingDeliveryTokens` .

Datum poslední vůle a potvrzení

Pokud došlo k neočekávanému ukončení připojení klienta MQTT , můžete produkt WebSphere MQ Telemetry nakonfigurovat tak, aby odeslal publikaci "last will and testament". Předefinujte obsah publikace a téma, kterému jej chcete odeslat. "Poslední vůle a testament" je vlastnost připojení. Vytvořte jej před připojením klienta.

Vytvořte téma pro poslední vůli a testament. Můžete vytvořit téma jako např. MQTTManagement/Connections/*server URI*/*client identifer*/Lost.

Nastavte "poslední vůli a testament" pomocí metody MqttConnectionOptions.setWill(MqttTopic lastWillTopic, byte [] lastWillPayload, int lastWillQos, boolean lastWillRetained) .

Zvažte vytvoření časového razítka ve zprávě lastWillPayload . Zahrnout další informace o klientovi, které pomáhají při identifikaci klienta a okolnosti připojení. Předejte objekt MqttConnectionOptions konstruktoru MqttClient .

Nastavte parametr lastWillQos na hodnotu 1 nebo 2, aby byla zpráva perzistentní v produktu IBM WebSphere MQa aby byla zajištěna její doručení. Chcete-li zachovat informace o naposledy ztraceném připojení, nastavte parametr lastWillRetained na hodnotu true.

Publikování "poslední vůle a testament" je odesláno odběratelům, pokud připojení skončí neočekávaně. Je odeslán, pokud připojení skončí, aniž by klient volal metodu MqttClient.disconnect .

Chcete-li monitorovat připojení, doplním publikace "last will and testament" s dalšími publikacemi k záznamu připojení a programovaným odpojením.

Perzistence zpráv v klientech produktu MQTT

Publikační zprávy jsou prováděny jako trvalé, pokud jsou odeslány s kvalitou služby alespoň jednou, nebo "přesně jednou". Na klientovi můžete implementovat vlastní mechanismus perzistence nebo použít výchozí mechanismus perzistence, který je dodáván s klientem. Perzistence funguje v obou směrech, pro publikace, které jsou odeslány na klienta nebo z klienta.

V produktu MQTTmá perzistence zpráv dva aspekty, způsob přenosu zprávy a informace o tom, zda je zpráva zařazena do fronty na serveru MQTT jako trvalá zpráva.

1. Perzistence zpráv klientských párů MQTT s kvalitou služeb. V závislosti na tom, jakou kvalitu služby vyberete pro zprávu, bude zpráva trvalá. Perzistence zpráv je nezbytná k implementaci požadované kvality služby.

   Určíte-li "maximálně jednou", QoS=0, klient zprávu zahodí, jakmile bude publikován. Dojde-li k selhání v předchozím zpracování zprávy, zpráva se neodešle znovu. I v případě, že klient zůstane aktivní, zpráva se znovu neodešle. Chování zpráv produktu QoS=0 se shoduje s chováním IBM WebSphere MQ rychlých přechodných zpráv.

   Je-li zpráva publikována klientem s QoS z 1 nebo 2, je vytrvalá. Zpráva je uložena lokálně a pouze vyřazena z klienta, když již není potřeba zajistit "alespoň jednou", QoS=1, nebo "přesně jednou", QoS=2, doručení.
2. Je-li zpráva označena jako QoS 1 nebo 2, je zařazena do fronty jako trvalá zpráva. Je-li označeno jako QoS=0, pak se zařadí do fronty jako přechodná zpráva. V produktu IBM WebSphere MQ jsou přechodné zprávy přenášeny mezi správci front "přesně jednou", pokud kanál zpráv nemá atribut NPMSPEED , který je nastaven na hodnotu FAST.

Trvalá publikace je uložena v klientu, dokud ji nebude přijímat klientská aplikace. Pro produkt QoS=2je publikace vyřazena z klienta, když zpětné volání aplikace vrátí řízení. V případě QoS=1 může aplikace obdržet publikování znovu, pokud dojde k selhání. V případě systému QoS=0zpětné volání přijme publikování ne více než jednou. Pokud dojde k selhání nebo je-li klient odpojen v době publikování, nemusí být tato publikace zveřejněna.

Pokud se přihlásíte k odběru tématu, můžete snížit úroveň služeb QoS , se kterou odběratel přijímá zprávy, aby odpovídal schopnostem perzistence. Publikace, které jsou vytvořeny na vyšší verzi QoS , jsou odeslány s nejvyšší hodnotou QoS , kterou odběratel požadoval.

Ukládání zpráv

Implementace datového úložiště na malých zařízeních se velmi liší. Model dočasně ukládající trvalé zprávy v úložišti, který je spravován klientem MQTT , může být příliš pomalý, nebo požadovat příliš mnoho

úložiště. V mobilních zařízeních může mobilní operační systém poskytovat službu úložiště, která je ideální pro zprávy produktu MQTT .

Klient MQTT má dvě rozhraní perzistence, aby byla zajištěna flexibilita při plnění omezení malých zařízení. Rozhraní definují operace, které jsou zapojeny do ukládání trvalých zpráv. Rozhraní jsou popsána v dokumentaci rozhraní API pro produkt Klient MQTT pro produkt Java. Pro odkazy na dokumentaci ke klientským rozhraním API pro knihovny klienta MQTT viz Odkaz na programování klienta MQTT. Rozhraní můžete implementovat tak, aby vyhovovala zařízení. Klient produktu MQTT spuštěný v prostředí Java SE má výchozí implementaci rozhraní, která ukládají trvalé zprávy v systému souborů. Používá balík `java.io` . Klient má také výchozí implementaci pro prostředí Java ME, `MqttDefaultMIDPPersistence`.

Třídy perzistence

`MqttClientPersistence`
: Předejte instanci implementace produktu `MqttClientPersistence` na klienta produktu MQTT jako parametr konstruktoru `MqttClient` . Pokud vynecháte argument `MqttClientPersistence` z konstruktoru `MqttClient` , klient MQTT ukládá trvalé zprávy pomocí třídy `MqttDefaultFilePersistence` nebo `MqttDefaultMIDPPersistence`.

`MqttPersistable`
: `MqttClientPersistence` získává a vkládá objekty `MqttPersistable` pomocí klíče úložiště. Musíte poskytnout implementaci produktu `MqttPersistable` , stejně jako implementaci produktu `MqttClientPersistence` , pokud nepoužíváte `MqttDefaultFilePersistence` nebo `MqttDefaultMIDPPersistence`.

`MqttDefaultFilePersistence`

: Klient MQTT poskytuje třídu `MqttDefaultFilePersistence` . Pokud konkretizovat `MqttDefaultFilePersistence` v aplikaci klienta, můžete poskytnout adresář k ukládání trvalých zpráv jako parametru konstruktoru `MqttDefaultFilePersistence` .

: Alternativně může klient MQTT vytvořit instanci `MqttDefaultFilePersistence` a umístit soubory do výchozího adresáře. Název adresáře je *`client identifier`*`-tcp `*`hostname portnumber`*. `"\", "\\", "/", ":"` a `" "` jsou odebrány z řetězce názvu adresáře.

: Cesta k adresáři je hodnota vlastnosti systému `rcp.data`. Není-li parametr `rcp.data` nastaven, cesta je hodnota vlastnosti systému `usr.data`.

: `rcp.data` je vlastnost přidružená k instalaci platformy OSGi nebo platformy Eclipse Rich Client Platform (RCP).

: `usr.data` je adresář, ve kterém byl spuštěn příkaz jazyka Java, který spustil aplikaci.

`MqttDefaultMIDPPersistence`
: `MqttDefaultMIDPPersistence` má výchozí konstruktor a žádné parametry. Používá balík produktu `javax.microedition.rms.RecordStore` k ukládání zpráv.

Publikace

Publikace jsou instance `MqttMessage` , které jsou přidruženy k řetězci tématu. Klient produktu MQTT může vytvářet publikace k odesílání do produktu IBM WebSphere MQa k odběru témat v produktu IBM WebSphere MQ MQ za přihlášení k odběru publikací.

`MqttMessage` má jako svůj informační obsah pole bajtů. Zamířit, aby zprávy byly co nejmenší. Maximální délka zprávy povolené protokolem MQTT je 250 MB.

Klientský program MQTT obvykle používá `java.lang.String` nebo `java.lang.StringBuffer` k manipulaci s obsahem zpráv. Pro usnadnění práce má třída `MqttMessage` metodu `toString` pro převod jejího informačního obsahu na řetězec. Chcete-li vytvořit informační obsah bajtového pole z `java.lang.String` nebo `java.lang.StringBuffer`, použijte metodu `getBytes` .

Metoda `getBytes` převádí řetězec na výchozí znakovou sadu pro platformu. Standardní znaková sada je obecně `UTF-8`. Příručky MQTT , které obsahují pouze text, jsou obvykle kódovány v produktu `UTF-8`. Chcete-li potlačit výchozí znakovou sadu, použijte metodu `getBytes("UTF8")` .

V produktu IBM WebSphere MQje publikace MQTT přijata jako zpráva `jms-bytes` . Zpráva obsahuje složku `MQRFH2` obsahující složku `<mqtt>`a složku `<mqps>` . Složka `<mqtt>` obsahuje položku `clientId` a `qos`, ale tento obsah se může v budoucnu měnit.

`MqttMessage` má tři další atributy: kvalitu služby, ať už je uchován, a zda je duplikát. Duplicitní příznak je nastaven pouze v případě, že kvalita služby je "alespoň jednou", nebo "přesně jednou". Pokud byla zpráva poslána dříve a klientem MQTT nebyla dostatečně rychle potvrzena, zpráva se odešle znovu s duplicitním atributem nastaveným na `true`.

Publikování

Chcete-li vytvořit publikování v aplikaci klienta MQTT , vytvořte `MqttMessage`. Nastavte informační obsah, kvalitu služby a informace o tom, zda je uchován, a volejte metodu `MqttTopic.publish(MqttMessage message)` , je vrácena `MqttDeliveryToken` a dokončení publikování je asynchronní.

Případně může klient produktu MQTT vytvořit dočasný objekt zprávy z parametrů v metodě `MqttTopic.publish(byte [] payload, int qos, boolean retained)` při vytváření publikace.

Pokud má publikování alespoň jednu úroveň kvality služeb `QoS=1` nebo `QoS=2`nebo "právě jednou", zavolá klient MQTT rozhraní `MqttClientPersistence` . Zavolá produkt `MqttClientPersistence` , aby uložil zprávu před vrácením doručovacího tokenu do aplikace.

Aplikace se může rozhodnout blokovat, dokud nebude zpráva doručena na server, pomocí metody `MqttDeliveryToken.waitForCompletion` . Alternativně může aplikace pokračovat bez blokování. Chcete-li zkontrolovat, zda jsou publikace doručeny, bez blokování, registrujte instanci třídy zpětného volání, která implementuje `MqttCallback` s klientem MQTT . Klient MQTT volá metodu `MqttCallback.deliveryComplete` , jakmile je publikace doručena. V závislosti na kvalitě služby může být doručení téměř okamžité pro `QoS=0`, nebo může nějakou dobu trvat, než `QoS=2`.

Pokud je doručení dokončeno, použijte metodu `MqttDeliveryToken.isComplete` . Zatímco hodnota `MqttDeliveryToken.isComplete` je `false`, můžete volat `MqttDeliveryToken.getMessage` pro získání obsahu zprávy. Pokud je výsledek volání `MqttDeliveryToken.isComplete` `true`, zpráva byla vyřazena a zavoláním příkazu `MqttDeliveryToken.getMessage` by došlo k výjimce ukazatele null. Mezi `MqttDeliveryToken.getMessage` a `MqttDeliveryToken.isCompleteneexistuje` žádná vestavěná synchronizace.

Pokud se klient odpojí před přijetím všech nevyřízených tokenů doručení, může se před připojením dotazovat nová instance klienta na nevyřízené tokeny doručení. Dokud se klient nepřipojí, nejsou dokončeny žádné nové dodávky a je bezpečné volat `MqttDeliveryToken.getMessage`. Chcete-li zjistit, které publikace nebyly doručeny, použijte metodu `MqttDeliveryToken.getMessage` . Nevyřízené tokeny doručení jsou zrušeny, pokud se připojíte k výchozí hodnotě `MqttConnectOptions.cleanSession` , `true`.

přihlášení odběru

Správce front nebo produkt IBM MessageSight je odpovědný za vytváření publikování pro odeslání na odběratele produktu MQTT . Správce front kontroluje, zda filtr témat v odběru vytvořeném klientem MQTT odpovídá řetězci tématu v publikování. Shoda může být buď přesná shoda, nebo může shoda obsahovat zástupné znaky. Před postoupením publikace odběrateli správce front zkontroluje správce front atributy tématu přidružené k této publikaci. Následuje postup vyhledávání popsaný v tématu Přihlášení k odběru pomocí řetězce tématu, který obsahuje zástupné znaky k identifikaci, zda objekt administrativního tématu uděluje oprávnění uživatele k odběru.

Když klient MQTT obdrží publikování s "alespoň jednou" kvalitou služby, volá metodu `MqttCallback.messageArrived` , aby zpracují publikování. Je-li kvalita služby publikace "přesně jednou", `QoS=2`, klient MQTT volá rozhraní produktu `MqttClientPersistence` , aby uložil zprávu při jejím přijetí. Pak zavolá příkaz `MqttCallback.messageArrived`.

Kvality služby poskytované klientem MQTT

Klient produktu MQTT poskytuje tři úrovně kvality služby pro dodávání publikací do produktu WebSphere MQ a do klienta MQTT : "maximálně jednou", "alespoň jednou" a "přesně jednou". Když klient MQTT odešle požadavek produktu IBM WebSphere MQ k vytvoření odběru, odešle se požadavek s kvalitou služby "alespoň jednou".

Quality of service of a publication is an attribute of `MqttMessage`. Je nastaven pomocí metody `MqttMessage.setQos`.

Metoda `MqttClient.subscribe` může snížit kvalitu služby aplikovanou na publikace odeslané klientovi na téma. Kvalita služeb publikace postoupené odběrateli se může lišit od kvality služby publikace. Nižší z těchto dvou hodnot se používá k předání publikace.

Nejvíce jednou
`QoS=0`

Zpráva je doručena nejvíce jednou nebo není doručena vůbec. Její doručení po síti není potvrzeno.
Zpráva není uložena. Při odpojení klienta nebo selhání serveru může dojít ke ztrátě zprávy.
`QoS=0` je nejrychlejší způsob přenosu. Někdy se říká "oheň a zapomenout".
Protokol MQTT nevyžaduje od serverů předávání publikačních publikací na straně `QoS=0` klientovi. Pokud je klient odpojen v době, kdy server obdrží publikování, může být publikování zrušeno, v závislosti na serveru. Služba telemetrie (MQXR) nevyřazovat zprávy odeslané s produktem `QoS=0`. Jsou uloženy jako přechodné zprávy a jsou zahozeny pouze v případě, že je správce front zastaven.

Nejméně jednou
`QoS=1`

`QoS=1` je výchozí režim přenosu.
Zpráva se vždy doručí alespoň jednou. Pokud odesilatel neobdrží potvrzení, je zpráva odeslána znovu s příznakem `DUP` , dokud není přijato potvrzení. Jako příjemce lze vícekrát odeslat stejnou zprávu a může ji několikrát zpracovat.
Zpráva musí být uložena lokálně na odesílateli a přijímači, dokud nebude zpracována.
Zpráva se odstraní z přijímače poté, co zpracoval zprávu. Je-li příjemcem zprostředkovatel, zpráva se publikuje na své odběratele. Je-li příjemcem klient, zpráva se doručí do aplikace odběratele. Jakmile je zpráva odstraněna, příjemce odešle potvrzení odesilateli.
Zpráva se odstraní od odesílatele poté, co mu bylo přijato potvrzení od příjemce.

Přesně jednou
`QoS=2`

Zpráva se vždy doručí přesně jednou.
Zpráva musí být uložena lokálně na odesílateli a přijímači, dokud nebude zpracována.
`QoS=2` je nejbezpečnější, ale nejpomalejší způsob přenosu. Mezi odesilatelem a příjemcem je třeba provést alespoň dva páry přenosů před tím, než je zpráva odstraněna od odesílatele. Zprávu lze na příjemce zpracovat po prvním přenosu.
V první dvojici přenosů odešle odesilatel zprávu a získá potvrzení od příjemce, že uložil zprávu. Pokud odesilatel neobdrží potvrzení, je zpráva odeslána znovu s příznakem `DUP` , dokud není přijato potvrzení.
Ve druhém páru přenosů odesilatel sděluje přijímači, že může dokončit zpracování zprávy, `"PUBREL"`. Pokud odesilatel neobdrží potvrzení o zprávě `"PUBREL"` , je zpráva `"PUBREL"` poslána znovu, dokud neobdrží potvrzení. Odesilatel odstraní zprávu, která byla uložena, když přijme potvrzení pro zprávu `"PUBREL"` .
Příjemce může zprávu zpracovat v první nebo druhé fázi za předpokladu, že tuto zprávu nezpracuje znovu. Je-li příjemcem zprostředkovatel, publikuje zprávu pro odběratele. Je-li příjemcem klient, doručí zprávu do aplikace odběratele. Příjemce pošle zprávu o dokončení zpět odesílateli, že dokončil zpracování zprávy.

Zachovaná publikování a klienti MQTT

Pokud vytvoříte odběr tématu, které má zachované publikování, bude vám okamžitě předáno nejnovější zachované publikování v tématu.

Pomocí metody `MqttMessage.setRetained` lze určit, zda má být publikování na téma zachováno, či nikoli.

Chcete-li odstranit zachované publikování v produktu IBM WebSphere MQ, spusťte příkaz MQSC CLEAR TOPICSTR**CLEAR TOPICSTR** .

Pokud vytvoříte publikování s informačním obsahem s hodnotou null, bude odběratelům předáno prázdné publikování. Ostatní zprostředkovatelé produktu MQTT mohou nepředat prázdnou publikaci odběratelům.

Pokud publikujete nezachované publikování na téma, které má zachované publikování, zachované publikování nebude mít vliv na zachované publikování. Aktuální odběratelé obdrží novou publikaci. Noví odběratelé obdrží zachované publikování jako první a poté obdrží nové publikace.

Když vytvoříte nebo aktualizujete zachované publikování, odešlete publikaci se `QoS` nebo 1 nebo 2. Pokud ji odešlete pomocí `QoS` z 0, produkt IBM WebSphere MQ vytvoří netrvalé zachované publikování. Publikování nebude zachováno, je-li správce front zastaven.

Použijte zachovaná publikování a zaznamenejte nejnovější hodnotu měření. Noví odběratelé uchovaného tématu okamžitě obdrží nejnovější hodnotu měření. Pokud od odběratele, který byl naposledy přihlášen k tématu publikování, nebyla provedena žádná nová měření, a pokud se odběratel znovu přihlásí, odběratel obdrží nejnovější zachované publikování v rámci tématu znovu.

Odběry

Vytvoření odběrů pro registraci zájmu v tématech publikování pomocí filtru témat. Klient může vytvořit více odběrů nebo odběr obsahující filtr témat, který používá zástupné znaky, a zaregistrovat tak zájem o více témat. Publikace na témata, která odpovídají filtrům, se posílají na klienta. Odběry mohou zůstat aktivní, když je klient odpojen. Publikace se odesílají klientovi, jakmile se znovu připojí.

Vytvořte odběry pomocí metod `MqttClient.subscribe` , které jsou předáním jednoho nebo více filtrů témat a parametrů kvality služby. Parametr quality of service nastavuje maximální kvalitu služby, kterou je odběratel připraven použít pro příjem zprávy. Zprávy odeslané tomuto klientovi nemohou být doručeny s vyšší kvalitou služby. Kvalita služby je nastavena na nižší z původní hodnoty, když byla zpráva publikována a úroveň uvedená pro odběr. Výchozí kvalita služby pro příjem zpráv je `QoS=1`, alespoň jednou.

Samotný požadavek na odběr je odeslán s produktem `QoS=1`.

Publications přijímá odběratel, když klient MQTT volá metodu `MqttCallback.messageArrived` . Metoda `messageArrived` také předává řetězec tématu, se kterým byla zpráva publikována, na odběratele.

Pomocí metod `MqttClient.unsubscribe` můžete odebrat odběr nebo sadu či odběry.

Příkaz WebSphere MQ může odebrat odběr. Seznam odběrů pomocí programu Průzkumník produktu WebSphere MQ nebo pomocí příkazů **runmqsc** nebo PCF. Všechny odběry klienta MQTT mají název. Zobrazí se název formuláře: *`ClientIdentifier`*:*`Topic name`*

Pokud použijete výchozí `MqttConnectOptions`nebo nastavíte `MqttConnectOptions.cleanSession` na `true` před připojením klienta, všechny staré odběry klienta se odeberou, když se klient připojí. Všechny nové odběry, které klient během relace vytvoří, jsou při odpojení odebrány.

Pokud před připojením nastavíte `MqttConnectOptions.cleanSession` na `false` , všechny odběry vytvořené klientem budou přidány ke všem odběrům, které existovaly pro klienta dříve, než se připojí. Všechny odběry zůstávají aktivní, když se klient odpojí.

Dalším přístupem k pochopení, jak atribut `cleanSession` ovlivňuje odběry, je považovat jej za modální atribut. Ve svém výchozím režimu, `cleanSession=true`, vytváří klient odběry a přijímá publikace pouze v rámci dané relace. V alternativním režimu, `cleanSession=false`, jsou odběry trvalé. Klient se může

připojit a odpojit a jeho odběry zůstanou aktivní. Když se klient znovu připojí, přijme všechny nedoručené publikace. V době připojení může upravit sadu odběrů, které jsou jeho jménem aktivní.

Před připojením musíte nastavit režim `cleanSession` ; režim trvá pro celou relaci. Chcete-li změnit jeho nastavení, je třeba klienta odpojit a znovu připojit. Pokud změníte režimy z použití `cleanSession=false` na `cleanSession=true`, všechny předchozí odběry pro klienta a všechny publikace, které nebyly přijaty, budou vyřazeny.

Publikace, které odpovídají aktivním odběrům, se odešlou klientovi, jakmile jsou publikovány. Je-li klient odpojen, odešle se klientovi, pokud se znovu připojí ke stejnému serveru se stejným identifikátorem klienta a `MqttConnectOptions.cleanSession` nastaveným na `false`.

Odběry pro určitého klienta jsou identifikovány identifikátorem klienta. Klienta můžete znovu připojit z jiného klientského zařízení na stejný server a pokračovat se stejnými odběry a přijímat nedoručené publikace.

Řetězce témat a filtry témat v klientech MQTT

Řetězce témat a filtry témat se používají pro publikování a odběr. Syntaxe řetězců témat a filtrů v klientech produktu MQTT je z velké části stejné jako řetězce témat v produktu IBM WebSphere MQ.

Řetězce témat se používají k odeslání publikování na odběratele. Vytvořte řetězec tématu za použití metody `MqttClient.getTopic(java.lang.String topicString)`.

Filtry témat se používají k odběru témat a přijímat publikování. Filtry témat mohou obsahovat zástupné znaky. Se zástupnými znaky se můžete přihlásit k odběru více témat. Vytvořte filtr témat pomocí metody odběru; například `MqttClient.subscribe(java.lang.String topicFilter)`.

Řetězce tématu

Syntaxe řetězce tématu IBM WebSphere MQ je popsána v tématu Řetězce tématu. Syntaxe řetězců témat MQTT je popsána ve třídě `MqttClient` v dokumentaci rozhraní API pro Klient MQTT pro produkt Java. Pro odkazy na dokumentaci ke klientským rozhraním API pro knihovny klienta MQTT viz Odkaz na programování klienta MQTT..

Syntaxe jednotlivých typů řetězců témat je téměř identická. Jsou zde čtyři menší rozdíly:

1. Topic strings sent to IBM WebSphere MQ by MQTT clients must follow the convention for queue manager names. Zejména ne řetězce témat nemohou obsahovat pomlčky.
2. Maximální délka se liší. Řetězce témat IBM WebSphere MQ jsou omezeny na 10,240 znaků. Klient MQTT může vytvořit řetězce témat až 65535 bajtů.
3. Řetězec tématu vytvořený klientem MQTT nemůže obsahovat znak null.
4. V produktu WebSphere Message Broker byla úroveň tématu null, `'.../...'` byla neplatná. Hodnoty témat s hodnotou null jsou podporovány produktem IBM WebSphere MQ.

Na rozdíl od IBM WebSphere MQ publish/subscribe, `mqttv3` protokol nemá koncepci objektu administrativního tématu. Řetězec tématu nelze zkonstruovat z objektu tématu a z řetězce tématu. Nicméně řetězec tématu je mapován na administrativní téma v produktu WebSphere MQ. Řízení přístupu přidružené k administrativnímu tématu určuje, zda je publikování publikováno do tématu, nebo zrušeno. Atributy, které jsou použity ke zveřejnění při jeho předání odběratelům, jsou ovlivněny atributy administrativního tématu.

Filtry témat

Syntaxe filtru témat IBM WebSphere MQ je popsána v tématu Schéma zástupných znaků na základě témat. Syntaxe filtrů témat, které lze sestavit s klientem MQTT , je popsána ve třídě `MqttClient` v dokumentaci k rozhraním API pro Klient MQTT pro produkt Java. Pro odkazy na dokumentaci ke klientským rozhraním API pro knihovny klienta MQTT viz Odkaz na programování klienta MQTT.

Syntaxe jednotlivých typů filtrů témat je téměř identická. Jediný rozdíl je v tom, jak různí zprostředkovatelé MQTT interpretují filtr témat. V produktu WebSphere Message Broker V6lze zástupný

znak s více úrovněmi použít pouze na konci filtru témat. V produktu WebSphere MQlze zástupný znak s více úrovněmi použít na libovolné úrovni ve stromu témat; například USA/#/Dutchess County.

Odkaz na programování klienta MQTT

Zde jsou odkazy na produkt Mobile Messaging and M2M Client Packa na přidruženou dokumentaci klienta API.

V produktu Mobile Messaging and M2M Client Packjsou knihovny klienta MQTT provázány s jejich generovanou dokumentací k rozhraní API. Balík klienta můžete stáhnout z webu IBM messaging community downloads.

Online kopie nejnovější dokumentace k rozhraní API můžete zobrazit pomocí následujících odkazů na projekt Eclipse Paho :

- Klient MQTT pro třídy Java
- Knihovna klienta MQTT pro jazyk C
- Asynchronní knihovna klienta MQTT pro jazyk C

Poznámka:

1. Propojte aplikace MQTT Java s balíkem `org.eclipse.paho.client.mqttv3` spíše než s produktem `com.ibm.micro.client.mqttv3`. existující balík nahradí. Balík produktu `com.ibm.micro.client.mqttv3` je k dispozici pro podporu existujících aplikací produktu MQTT Java .
2. V 7.5.0.1 Propojte aplikace klienta MQTT pro C s knihovnou `MQTTAsync` , a tím raději knihovnu `MQTTClient` . Produkt `MQTTClient` poskytuje podporu existujících aplikací MQTT pro jazyk C.
3. Server Klient systému zpráv MQTT pro produkt JavaScript vyžaduje server MQTT , který podporuje produkt WebSockets. Například, IBM WebSphere MQ Version 7.5 a pozdější verze to dělají.

Začínáme se servery MQTT

Servery systému zpráv, které podporují přenosový protokol produktu MQTT , jsou dostupné z produktu IBM a dalších. Nejzákladnější server MQTT umožňuje výměnu zpráv pomocí mobilních aplikací a zařízení podporovaných knihovnami klienta MQTT . IBM WebSphere MQ a IBM MessageSight jsou servery MQTT z produktu IBM. Kromě toho, že se chová jako základní servery MQTT , si také vyměňují zprávy mezi aplikacemi klienta MQTT a podnikovými aplikacemi. Všechny servery MQTT od IBM podporují protokol MQTT version 3.1 a MQTT přes WebSocket protocol.

Aktuální servery MQTT z produktu IBM

IBM WebSphere MQ

- Produkt IBM WebSphere MQ poskytuje podnikový systém zpráv. Komponenta telemetrie umožňuje produktu IBM WebSphere MQ také fungovat jako server MQTT .
- To podporuje vaše mobilní, machine-to-machine (M2M) a aplikace založené na zařízení a také jim umožňuje vyměňovat si zprávy s aplikacemi podnikového systému zpráv, jako jsou aplikace IBM WebSphere MQ a JMS .
- Instalace produktu IBM WebSphere MQ obsahuje kopii sady nástrojů MQTT SDK z produktu IBM. Tato sada SDK poskytuje ukázkové klientské aplikace klienta MQTT a knihovny klienta MQTT , které tyto aplikace podporují.

  Poznámka: Chcete-li získat nejvíce aktuální verzi této sady SDK, stáhněte Mobile Messaging and M2M Client Pack. Další informace viz “Začínáme s klienty MQTT” na stránce 10.
- Podpora produktu MQTT byla poprvé obsažena v produktu IBM WebSphere MQ Version 7.0.1. Úplné informace o jednotlivých verzích produktu IBM WebSphere MQnaleznete v následující dokumentaci produktu:
  - WebSphere MQ Telemetry verze 7.5

– WebSphere MQ Telemetry verze 7.1

Stručný úvod k produktu IBM WebSphere MQa kroky k zahájení práce s komponentou IBM WebSphere MQ Telemetry naleznete v tématu “IBM WebSphere MQ jako server MQTT” na stránce 137.

IBM MessageSight

- IBM MessageSight je server MQTT založený na zařízení, který může současně připojit masivní počet klientů MQTT a poskytuje výkon a rozšiřitelnost potřebnou k uspokojení rostoucího množství mobilních zařízení a senzorů. Podporuje protokol MQTT version 3.1 a MQTT v rámci WebSocket protocol.

- Hlavní vlastnosti a výhody produktu IBM MessageSight jako serveru MQTT jsou následující:
  - Vysoký výkon, spolehlivost a škálovatelný systém zpráv.
  - Navrženo speciálně pro scénáře typu počítač-na-počítač (M2M) a Internet of Things, tím, že podporuje rozsáhlé komunity pro souběžně připojené koncové body.
  - Snadné instalace a použití. Může být spuštěno a spuštěno za méně než 30 minut.
  - Podpora pro nativní mobilní aplikace, které zahrnují Android a iOS.
  - Integrace s produktem IBM WebSphere MQ jako zprostředkovatel publikování a odběru.
- Rychlý úvod do produktu IBM MessageSightnaleznete v části Úvod MessageSight na webu YouTube a oznámení MessageSight. Podrobné technické informace naleznete v tématu Dokumentace k produktu MessageSight.

IBM WebSphere MQ Telemetry daemon for devices

- To je také známo jako IBM WebSphere MQ Telemetry advanced client for C. Jedná se o server s malým obsazeným prostorem MQTT , který se obvykle spouští v satelitních lokalitách nebo zařízeních v blízkosti okraje sítě; například v rámečcích s horním okrajem, vzdálených telemetrických jednotek nebo prodejních terminálů s možností prodejního místa.
- Typickým použitím je soustředit mnoho klientských připojení produktu MQTT , které jsou pak připojeny k produktu IBM WebSphere MQ přes internet v jediném připojení MQTT . Můžete například instalovat velký počet senzorů v budově, připojit je k serveru IBM WebSphere MQ Telemetry daemon for devicesa připojit démona k IBM WebSphere MQ.
- IBM WebSphere MQ Telemetry daemon for devices je součástí produktu IBM WebSphere MQ. Samostatná licence je vyžadována pro připojení k produktu IBM WebSphere MQ. Viz IBM Oznámení o softwaru 212-091 pro Spojené státy 212-091.

Really Small Message Broker

- Really Small Message Broker (RSMB) je verze produktu IBM WebSphere MQ Telemetry daemon for devices. Hlavní rozdíl je v použití. RSMB je malý testovací server, dostupný z produktu IBM alphaWorksa určený k použití při vyhodnocování nebo experimentování s řešeními založenými na protokolu MQTT. RSMB podporuje MQTT na řadě platforem Linux , na Windows XP, na Apple Mac OS X Leoparda na Unslung (Linksys NSLU12)

Předchozí servery MQTT z produktu IBM

WebSphere Message Broker ***(nyní známá jako IBM Integration Bus)***

- Produkt WebSphere Message Broker verze 6 poskytl svůj vlastní server MQTT . Podpora byla nahrazena v produktu WebSphere Message Broker verze 7 pomocí komponenty telemetrie IBM WebSphere MQ.

Další servery MQTT

Produkt MQTT.org udržuje seznam serverů a zprostředkovatelů produktu MQTT na své stránce Software včetně otevřených zdrojových serverů.

Související úlohy

"Začínáme s klienty MQTT" na stránce 10
Můžete začít vyvíjet mobilní aplikaci nebo aplikaci machine-to-machine (M2M) sestavením a spuštěním ukázkové aplikace klienta MQTT , která používá knihovnu klienta MQTT . Ukázkové aplikace a přidružené knihovny klienta jsou k dispozici v adresáři Mobile Messaging and M2M Client Pack na adrese IBM. Existují verze aplikací a klientských knihoven napsané v adresáři Java, v adresáři JavaScripta v adresáři C. Tyto aplikace můžete spouštět na většině platforem a zařízení, včetně zařízení a produktů Android z produktu Apple.

IBM WebSphere MQ jako server MQTT

Úvod do používání serveru MQTT , který je zahrnutý v produktu IBM WebSphere MQ.

Chcete-li začít, postupujte podle kroků uvedených v následujících článcích:

- "instalaceIBM WebSphere MQ" na stránce 137
- "Konfigurace služby MQTT z příkazového řádku" na stránce 139
- "Konfigurace služby MQTT s produktem IBM WebSphere MQ Explorer" na stránce 141

Poznámka: Můžete začít rychle pomocí příkladu rozhraní příkazového řádku. Avšak, pokud se vaše konfigurace výrazně liší od příkladu, potřebujete více znalostí a dovedností, abyste mohli efektivně používat rozhraní příkazového řádku. Pomocí rozhraní produktu IBM WebSphere MQ Explorer lze snadno začít a snadno provádět standardní konfigurační úlohy.

Klíčové koncepční informace o komponentě IBM WebSphere MQ Telemetry naleznete v následujících článcích v dokumentaci k produktu IBM WebSphere MQ :

- Připojení telemetrických zařízení ke správci front
- Služba (MQXR) telemetrie
- Telemetry Channels, Kanály telemetrie

Související informace

Konfigurace správce front pro telemetrii v systémech Linux a AIX
Konfigurace správce front pro telemetrii v systému Windows
Konfigurace distribuovaných front pro odesílání zpráv klientům MQTT
Správa produktu WebSphere MQ Telemetry

instalaceIBM WebSphere MQ

Postupujte podle těchto pokynů, chcete-li získat a instalovat produkt IBM WebSphere MQ a nakonfigurovat produkt IBM WebSphere MQ Telemetry na serveru Windows nebo Linux.

Než začnete

Informace o operačních systémech, které jsou podporovány službou MQTT spuštěnou v produktu IBM WebSphere MQ, viz Požadavky na systém produktu IBM WebSphere MQ Telemetry .

Získejte kopii instalačních materiálů IBM WebSphere MQ a licencí jedním z následujících způsobů:

1. Zeptejte se administrátora IBM WebSphere MQ na instalační materiály a potvrďte, že můžete přijmout licenční smlouvu.
2. Získejte 90denní zkušební kopii produktu IBM WebSphere MQ. Viz Vyhodnotit: IBM WebSphere MQ.
3. Zakupte IBM WebSphere MQ. Viz Stránka produktuIBM WebSphere MQ.

Informace o této úloze

Nainstalujte IBM WebSphere MQ jako `root` na Linuxa jako administrátor na Windows. V době instalace vyberte další volby `Telemetrie` a `Klienti telemetrie` a nainstalujte komponentu produktu IBM WebSphere MQ Telemetry . Vytvořte ID uživatele pro administraci produktu IBM WebSphere MQa zkontrolujte, zda je ID uživatele "guest" definováno. ID uživatele "guest" se používá v ukázkové konfiguraci služby produktu MQTT k autorizaci přístupu produktu MQTT k produktu IBM WebSphere MQ.

Po instalaci produktu IBM WebSphere MQspusťte službu MQTT tak, že provedete kroky uvedené v tématu “Konfigurace služby MQTT z příkazového řádku” na stránce 139 nebo “Konfigurace služby MQTT s produktem IBM WebSphere MQ Explorer” na stránce 141.

Postup

1. Přihlaste se jako `root` na Linuxnebo jako administrátor na Windows.
2. Nainstalujte produkt IBM WebSphere MQ.

   Postupujte podle pokynů v tématu Instalace serveru WebSphere MQ v systému Linux nebo Instalace serveru WebSphere MQ v systému Windows. Vyberte volbu `Služba telemetrie` a `Klienti telemetrie` , chcete-li instalovat komponentu produktu IBM WebSphere MQ Telemetry .

   V systému Linuxvezměte v úvahu pokyny v sekci "Co dělat dál" , abyste učinili primární instalaci. I když je tato instalace jedinou instalací produktu IBM WebSphere MQ na vaší pracovní stanici, proveďte její primární. Viz téma Jediná instalace produktu WebSphere MQ verze 7.1nebo novější, nakonfigurovaná jako primární instalace.

   Chcete-li přesně dodržet pokyny ke konfiguraci vzorku, musíte provést primární instalaci.

   Více instalací: Chcete-li pracovat s neprimární instalací, spusťte příkaz setmqenv . Nastaví prostředí IBM WebSphere MQ v příkazovém okně na pracovní stanici. Viz Vícenásobné instalace.

   Za předpokladu, že jste přijali výchozí umístění instalace nabízené instalačním programem, je produkt IBM WebSphere MQ nainstalován v následujících adresářích:

   Linux 64 bitů

   ```
   /opt/mqm
   ```

   Windows 32 bitů

   ```
   C:\Program Files\IBM\WebSphere MQ
   ```

   Windows 64 bitů

   ```
   C:\Program Files (x86)\IBM\WebSphere MQ
   ```

   Instalační adresář se zobrazí jako *MQ_INSTALLATION_PATH*
3. Volitelné: Přidejte uživatele, kterého chcete spravovat, IBM WebSphere MQ se skupinou `mqm` na této pracovní stanici.

   This step is optional on Windows because you can administer IBM WebSphere MQ as a Windows administrator. Viz Oprávnění k administraci produktu WebSphere MQ na systémech UNIX a Windows.

   Je-li vaše pracovní stanice Windows členem domény, prostudujte si téma Doména systému Windows 2000 s jinou než výchozí hodnotou, nebo doména Windows 2003 a Windows Server 2008 s výchozími oprávněními zabezpečení.

V systému Linuxvytvoří instalační program uživatele mqmjako člena skupiny mqm. Poskytněte tomuto uživateli heslo, nebo vytvořte jiného uživatele s mqm jako jeho primární skupinou.

4. Volitelné: Přihlaste se pomocí uživatele, kterého jste učinili členem skupiny mqm .

   This step is optional on Windows because you can administer IBM WebSphere MQ as a Windows administrator.

5. Zkontrolujte, že ID uživatele "guest" je definováno na vaší pracovní stanici.

   ID uživatele "guest" je "guest" na Windows a "nobody" na Linux. ID uživatele "guest" nevyžaduje žádná oprávnění operačního systému nebo práva.

Výsledky

Nainstalovali jste produkt IBM WebSphere MQ na pracovní stanici jako primární instalaci produktu IBM WebSphere MQ a vytvořili skupinu mqm. Instalace poskytuje oprávnění pro administraci produktu IBM WebSphere MQ na členy skupiny mqm . Členové skupiny administrátorů v systému Windows mají také oprávnění ke správě produktu IBM WebSphere MQ.

Jak pokračovat dále

1. Nakonfigurujte službu MQTT z příkazového řádku nebo z IBM WebSphere MQ Explorer; viz “Konfigurace služby MQTT s produktem IBM WebSphere MQ Explorer” na stránce 141 nebo “Konfigurace služby MQTT z příkazového řádku” na stránce 139.
2. Otestujte své klienty Android, iOS, WebSockets, Javaa "C" MQTT .
3. Když dokončíte testování, odeberte správce front a službu MQTT spuštěním příkazu *MQ_INSTALLATION_PATH*\mqxr\samples\CleanupMQM.bat na Windows a *MQ_INSTALLATION_PATH*/mqxr/samples/CleanupMQM.sh na Linux.

Související informace

Instalace produktu WebSphere MQ Telemetry

Instalace serveru WebSphere MQ v systému Linux

Instalace serveru WebSphere MQ v systému Windows

Konfigurace služby MQTT z příkazového řádku

Postupujte podle těchto pokynů ke konfiguraci IBM WebSphere MQ pomocí příkazového řádku pro spuštění ukázkových aplikací IBM WebSphere MQ Telemetry . Tento postup ukazuje, jak spustit skript pro vytvoření služby MQTT na novém správci front s názvem MQXR_SAMPLE_QM.

Než začnete

Chcete-li nastavit službu MQTT , musíte mít administrativní přístup ke správci front IBM WebSphere MQ . Máte několik způsobů, jak získat přístup ke správci front:

1. Získejte kopii produktu IBM WebSphere MQ a vytvořte správce front na své pracovní stanici s produktem Linux nebo Windows . Chcete-li získat a instalovat produkt IBM WebSphere MQ, postupujte podle pokynů v části “instalaceIBM WebSphere MQ” na stránce 137 . Všimněte si, že při instalaci musíte také vybrat Telemetry Service a Telemetry Clients . Chcete-li přidat tyto volby, můžete také upravit existující instalaci.
2. Obraťte se na administrátora produktu IBM WebSphere MQ a požádejte jej o administrativní přístup ke správci front na serveru, který má produkt IBM WebSphere MQ Telemetry nainstalován jako volbu.

   V 7.5.0.1 Kromě názvu správce front je třeba zadat alespoň dva porty TCP/IP pro produkt MQTT a pro prostor MQTT v rámci produktu WebSockets. Pokud plánujete připojení zabezpečených klientů, vyžadujete alespoň dva další porty.

Chcete-li provést kroky v úloze přesně tak, jak jsou popsány, musíte být schopni vytvořit správce front s názvem MQXR_SAMPLE_QMa port TCP/IP 1883 musí být nepoužívaný.

Informace o této úloze

V této úloze spustíte skript, který vytvoří správce front, a poté nakonfiguruje službu MQTT tak, aby naslouchala připojením klienta MQTT V3.1 na portu 1883. Konfigurace poskytuje všem oprávnění pro publikování a odběr jakéhokoli tématu. Konfigurace zabezpečení a řízení přístupu je minimální a je určena pouze pro správce front, který je v zabezpečené síti s omezeným přístupem. Chcete-li spustit IBM WebSphere MQ a MQTT v nezabezpečeném prostředí, musíte nakonfigurovat zabezpečení. Chcete-li nakonfigurovat zabezpečení pro IBM WebSphere MQ a MQTT, podívejte se na související odkazy na konci této úlohy.

Postup

1. Přihlaste se s ID uživatele, který má administrativní oprávnění k produktu IBM WebSphere MQ.

   Chcete-li definovat ID uživatele s administrativním oprávněním k produktu IBM WebSphere MQ, viz krok 3 v příručce “instalaceIBM WebSphere MQ” na stránce 137.
2. Otevřete příkazové okno a spusťte ukázkový příkazový skript k vytvoření a spuštění ukázkového správce front nazvaného `MQXR_SAMPLE_QM` a služby MQTT .

   Cesta k ukázkovému skriptu je `%MQ_FILE_PATH%\mqxr\samples\SampleMQM.bat` na Windows a *`MQ_INSTALLATION_PATH`*`/mqxr/samples/SampleMQM.sh` na Linux.

   Chcete-li vytvořit a nakonfigurovat správce front, zadejte následující příkaz:

   - Windows

     ```
     "%MQ_FILE_PATH%\mqxr\samples\SampleMQM.bat"
     ```

   - Linux

     ```
     MQ_INSTALLATION_PATH/mqxr/samples/SampleMQM.sh
     ```

Výsledky

Ukázka vytvoří kanál MQTT s názvem `PlainText` s těmito vlastnostmi v systému Windows:

```
com.ibm.mq.MQXR.channel/PlainText: \
com.ibm.mq.MQXR.Protocol=MQTT;\
com.ibm.mq.MQXR.Port=1883;\
com.ibm.mq.MQXR.Backlog=4096;\
com.ibm.mq.MQXR.UserName=Guest;\
com.ibm.mq.MQXR.StartWithMQXRService=true
```

Vlastnosti kanálu v systému Linux jsou stejné jako vlastnosti Windows, kromě `com.ibm.mq.MQXR.UserName=nobody`.

Klienti MQTT V3.1 , kteří se připojují k portu 1883, přistupují k serveru IBM WebSphere MQ s ID uživatele nastaveným v proměnné *`com.ibm.mq.MQXR.UserName`*. Ukázkový skript autorizuje ID uživatele pomocí následujících příkazů obslužného programu IBM WebSphere MQ :

```
setmqaut -m MQXR_SAMPLE_QM -t topic -n SYSTEM.BASE.TOPIC -p com.ibm.mq.MQXR.UserName -all +pub
+sub
setmqaut -m MQXR_SAMPLE_QM -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p com.ibm.mq.MQXR.UserName -all
+put
```

První příkaz dává uživateli oprávnění k publikování a přihlášení k odběru témat, která dědí jejich oprávnění od základního tématu. Druhý příkaz dává uživateli oprávnění vkládat zprávy do přenosové fronty `SYSTEM.MQTT.TRANSMIT.QUEUE` . Služba MQTT odesílá zprávy na serveru `SYSTEM.MQTT.TRANSMIT.QUEUE` jako publikace odběratelům MQTT .

Skript spustí službu MQTT ve frontě, aby naslouchala připojením na portu 1883.

Jak pokračovat dále

Postupujte takto, chcete-li testovat připojení spuštěním ukázkové aplikace produktu MQTT V3.1 Java .

Zdroj pro ukázkovou aplikaci Java je v souboru `MQTTV3Sample.java` .

Ke spuštění ukázky jsou nutná dvě příkazová okna. Spusťte ukázku jako odběratel v jednom okně a jako vydavatel na straně druhé.

- Windows Chcete-li spustit odběratele, spusťte příkaz

  ```
  "%MQ_FILE_PATH%\mqxr\samples\RunMQTTV3Sample.bat" -a subscriber
  ```

  Chcete-li publikovat, spusťte příkaz:

  ```
  "%MQ_FILE_PATH%\mqxr\samples\RunMQTTV3Sample.bat"
  ```

- Linux Chcete-li spustit odběratele, spusťte příkaz

  ```
  MQ_INSTALLATION_PATH/mqxr/samples/RunMQTTV3Sample.sh -a subscriber
  ```

  Chcete-li publikovat, spusťte příkaz:

  ```
  MQ_INSTALLATION_PATH/mqxr/samples/RunMQTTV3Sample.sh
  ```

Vydavatel a předplatitel zapisují výstup do svých příkazových oken:

```
Connected to tcp://localhost:1883
Publishing to topic "MQTTV3Sample/Java/v3" qos 2
Disconnected
Press any key to continue . . .
```

Obrázek 27. Výstup z vydavatele

```
Connected to tcp://localhost:1883
Subscribing to topic "MQTTV3Sample/#" qos 2
Press <Enter> to exit
Topic:          MQTTV3Sample/Java/v3
Message:        Message from MQTTv3 Java client
QoS:            2
```

Obrázek 28. Výstup z odběratele

Server je nyní připraven pro otestování vaší aplikace MQTT V3.1 .

Související úlohy

Konfigurace služby MQTT pomocí Průzkumníka produktu WebSphere MQ
Chcete-li spustit ukázkové klienty produktu IBM WebSphere MQ Telemetry , postupujte podle těchto pokynů ke konfiguraci produktu IBM WebSphere MQ pomocí produktu IBM WebSphere MQ Explorer . Tento postup ukazuje, jak vytvořit službu MQTT spuštěním průvodce konfigurací produktu `Define sample` .

Související informace

WebSphere MQ Telemetry
Vývoj aplikací pro produkt WebSphere MQ Telemetry
Správa produktu WebSphere MQ Telemetry
Zabezpečení produktu WebSphere MQ Telemetry

Konfigurace služby MQTT s produktem IBM WebSphere MQ Explorer

Chcete-li spustit ukázkové klienty produktu IBM WebSphere MQ Telemetry , postupujte podle těchto pokynů ke konfiguraci produktu IBM WebSphere MQ pomocí produktu IBM WebSphere MQ Explorer .

Tento postup ukazuje, jak vytvořit službu MQTT spuštěním průvodce konfigurací produktu `Define sample` .

Než začnete

Chcete-li nastavit službu MQTT , musíte mít administrativní přístup ke správci front IBM WebSphere MQ . Máte několik způsobů, jak získat přístup ke správci front:

1. Získejte kopii produktu IBM WebSphere MQ a vytvořte správce front na své pracovní stanici s produktem Linux nebo Windows . Chcete-li získat a instalovat produkt IBM WebSphere MQ, postupujte podle pokynů v části “instalaceIBM WebSphere MQ” na stránce 137 . Všimněte si, že při instalaci musíte také vybrat `Telemetry Service` a `Telemetry Clients` . Chcete-li přidat tyto volby, můžete také upravit existující instalaci.
2. Obraťte se na administrátora produktu IBM WebSphere MQ a požádejte jej o administrativní přístup ke správci front na serveru, který má produkt IBM WebSphere MQ Telemetry nainstalován jako volbu.
   V 7.5.0.1 Kromě názvu správce front je třeba zadat alespoň dva porty TCP/IP pro produkt MQTT a pro prostor MQTT v rámci produktu WebSockets. Pokud plánujete připojení zabezpečených klientů, vyžadujete alespoň dva další porty.

Chcete-li provést kroky v úloze přesně tak, jak jsou popsány, musíte být schopni vytvořit správce front s názvem `MQXR_SAMPLE_QM`a port TCP/IP 1883 musí být nepoužívaný.

Informace o této úloze

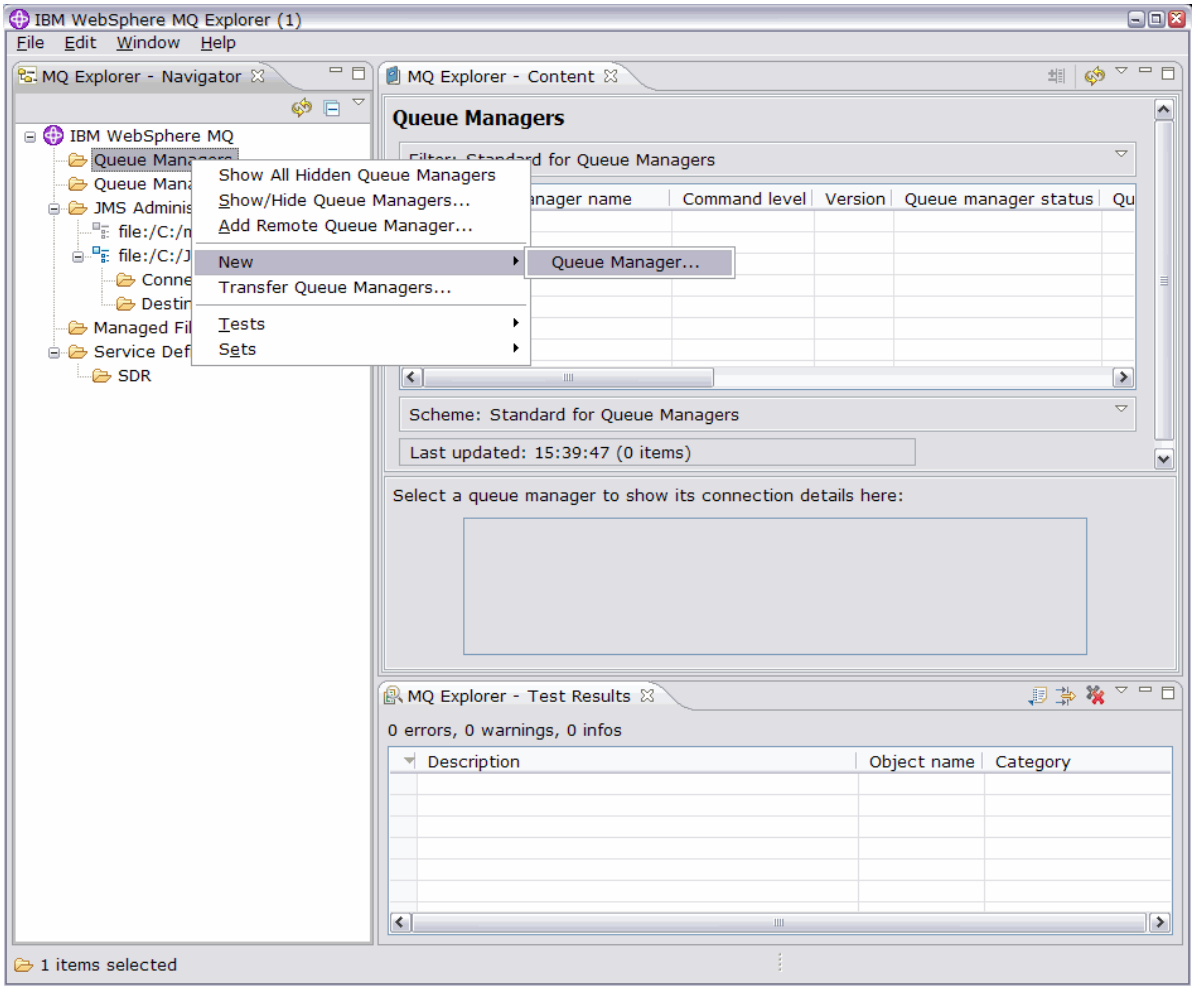
V této úloze spustíte průvodce konfigurací produktu IBM WebSphere MQ Explorer `Define sample` , abyste vytvořili službu MQTT , která bude naslouchat připojením klienta MQTT V3.1 na portu 1883. Konfigurace poskytuje všem oprávnění pro publikování a odběr jakéhokoli tématu. Konfigurace zabezpečení a řízení přístupu je minimální a je určena pouze pro správce front, který je v zabezpečené síti s omezeným přístupem. Chcete-li spustit IBM WebSphere MQ a MQTT v nezabezpečeném prostředí, musíte nakonfigurovat zabezpečení. Chcete-li nakonfigurovat zabezpečení pro IBM WebSphere MQ a MQTT, podívejte se na související odkazy na konci této úlohy.

Postup

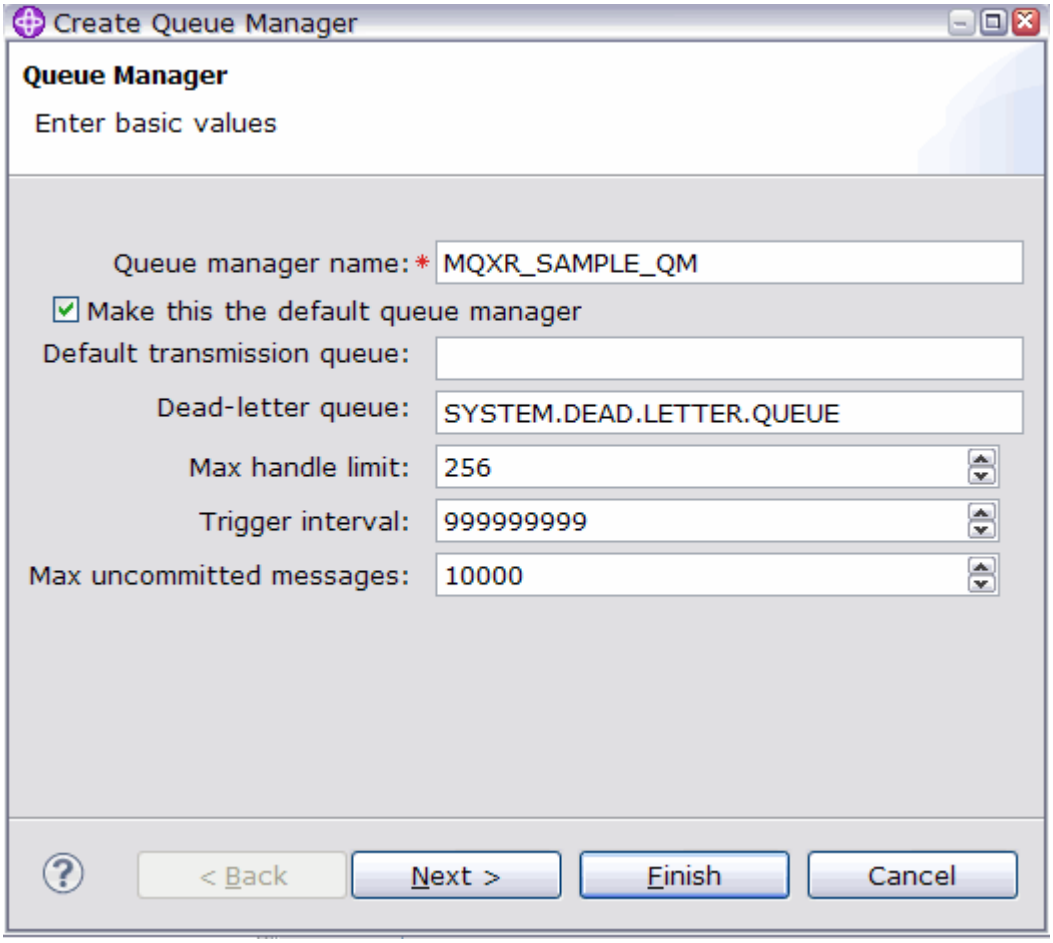
1. Přihlaste se s ID uživatele, který má administrativní oprávnění k produktu IBM WebSphere MQ.

   Chcete-li definovat ID uživatele s administrativním oprávněním k produktu IBM WebSphere MQ, viz krok 3 v příručce “instalaceIBM WebSphere MQ” na stránce 137.
2. Otevřete příkazové okno a spusťte příkaz IBM WebSphere MQ Explorer **strmqcfg** , který spustí příkaz IBM WebSphere MQ Explorer.
3. Vytvoření správce front

   a) Spustit průvodce **Nový správce front**

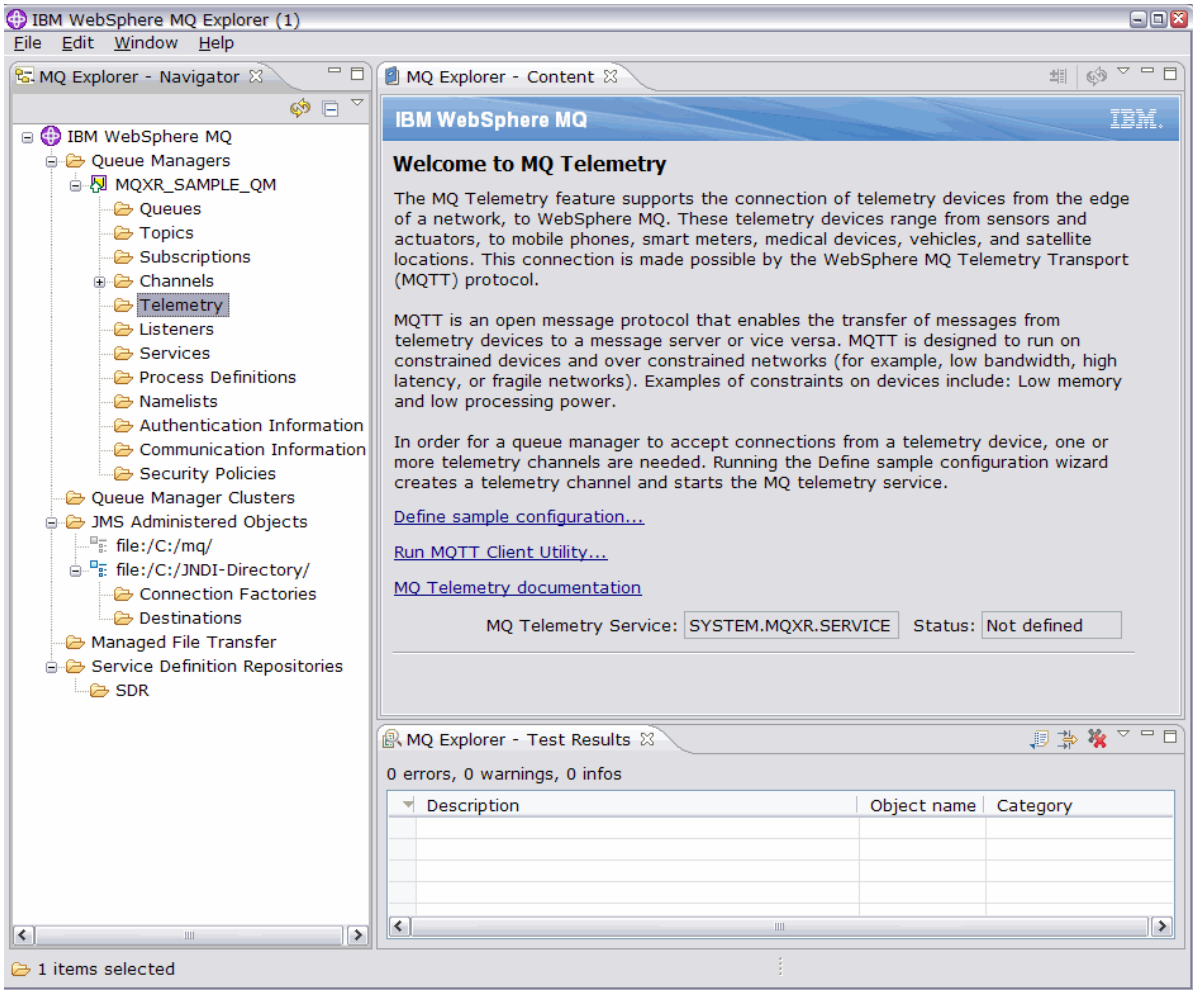


b) Zadejte **Název správce fron**a název **fronty nedoručených zpráv**. Z důvodu pohodlí jej můžete nastavit jako výchozí správce front. Klepněte na tlačítko **Dokončit**.



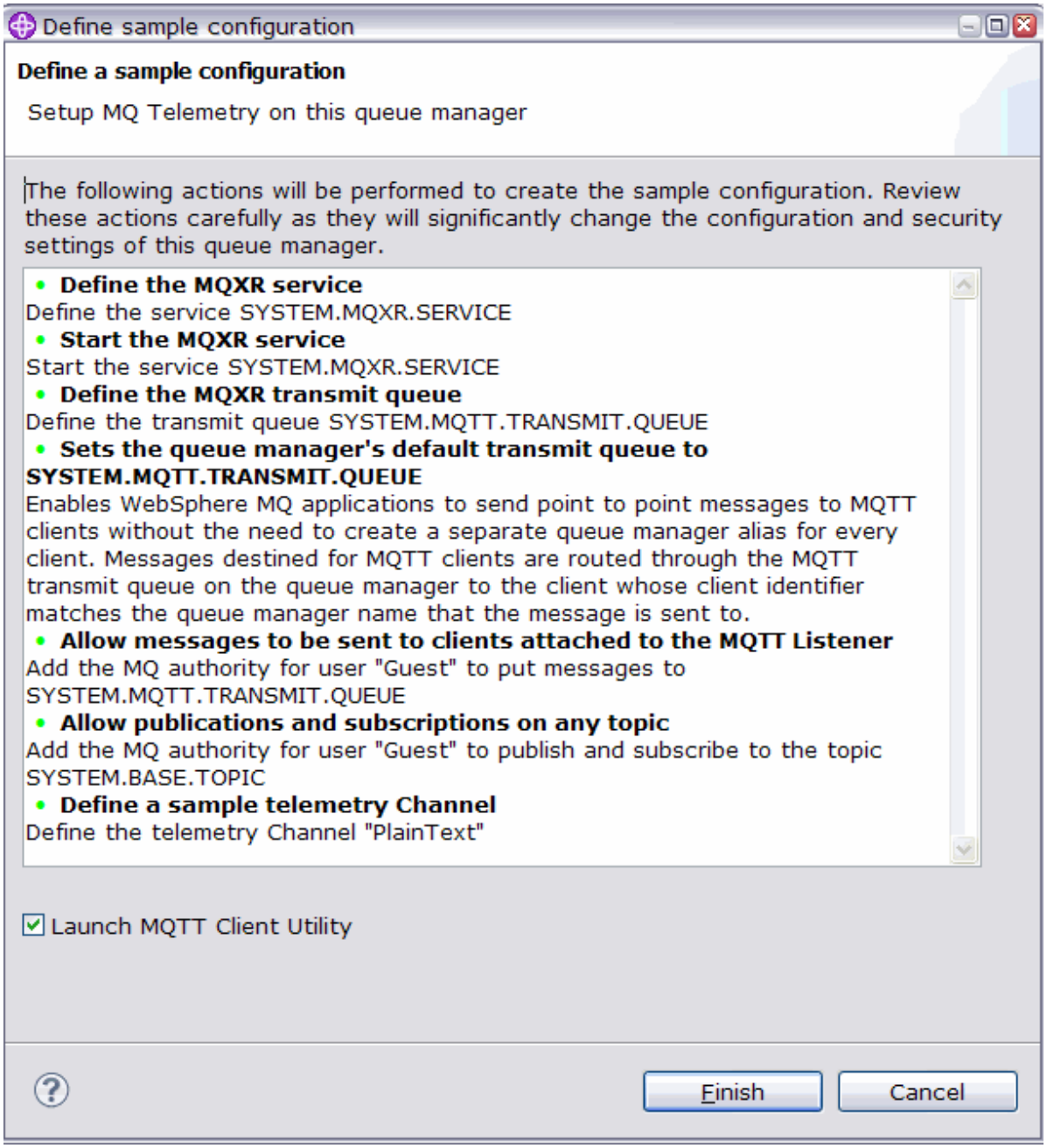
Produkt IBM WebSphere MQ Explorer vytvoří správce front a spustí jej.

4. Spusťte průvodce **Definovat ukázkovou konfiguraci** produktu Telemetry.
   a) Otevřete složku Telemetrie pro správce front.



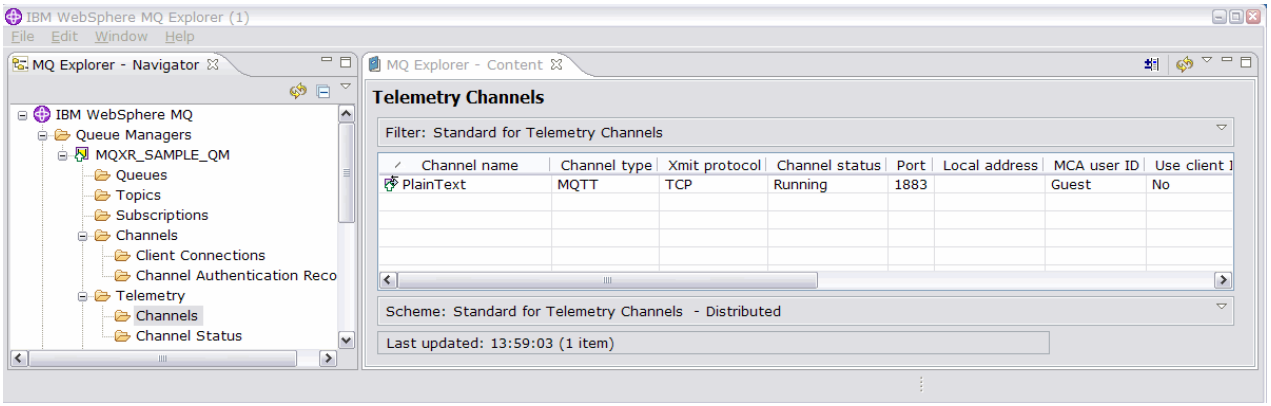
b) Klepnutím na volbu **Definovat ukázkovou konfiguraci** spusťte průvodce.

c) Klepnutím na tlačítko **Dokončit** vytvořte službu telemetrie a spusťte obslužný program klienta produktu MQTT .



Výsledky

Chcete-li vypsat ukázkové kanály, otevřete složku Kanály telemetrie.



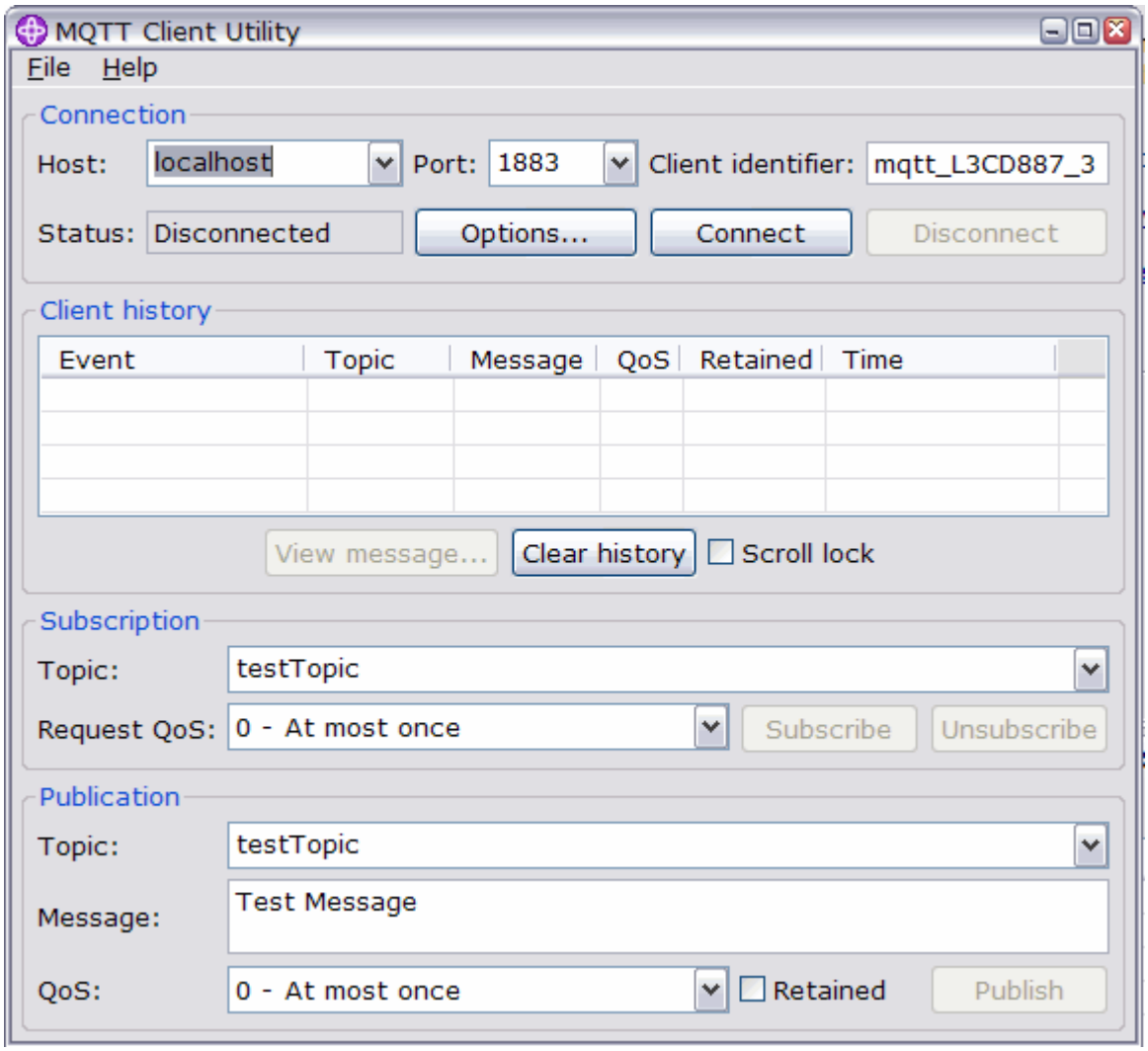
Můžete upravit vlastnosti tohoto kanálu a přidat a odstranit kanály v tomto okně.

Jak pokračovat dále

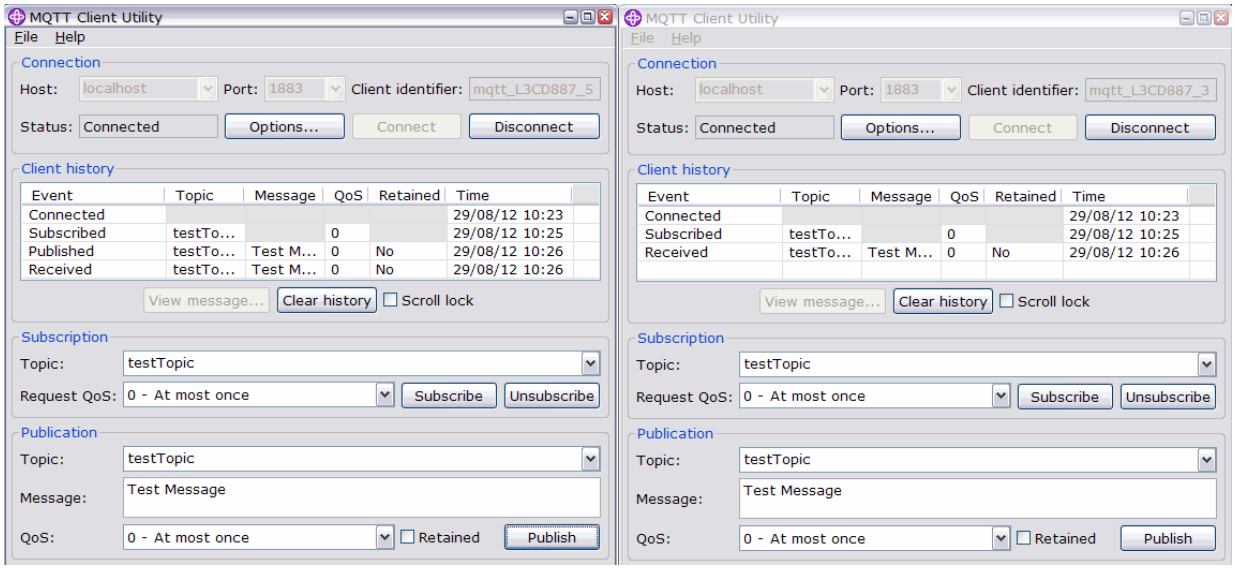
Otestujte připojení spuštěním obslužného programu klienta produktu MQTT .

1. Chcete-li spustit obslužný program klienta, otevřete složku **Telemetrie** a dvakrát klepněte na volbu **Spustit obslužný program klienta protokolu MQTT** .

   Otevřou se dvě okna **Obslužný program klienta protokolu MQTT** , identická, ale pro různé identifikátory klientů.



2. V obou oknech klepněte na volbu **Připojit** .
3. V obou oknech klepněte na volbu **Odebírat** .
4. V každém okně klepněte na tlačítko **Publikovat** . Výsledky jsou zobrazeny v Obrázek 29 na stránce 148



Obrázek 29. Výsledky

5. V obou oknech klepněte na tlačítko **Odpojit** .

Server je nyní připraven pro otestování vaší aplikace MQTT V3.1 .

Související úlohy

Konfigurace služby MQTT z příkazového řádku
Postupujte podle těchto pokynů ke konfiguraci IBM WebSphere MQ pomocí příkazového řádku pro spuštění ukázkových aplikací IBM WebSphere MQ Telemetry . Tento postup ukazuje, jak spustit skript pro vytvoření služby MQTT na novém správci front s názvem MQXR_SAMPLE_QM.

Správa produktu WebSphere MQ Telemetry

Související informace

WebSphere MQ Telemetry
Správa produktu WebSphere MQ Telemetry s programem Průzkumník produktu WebSphere MQ
Vývoj aplikací pro produkt WebSphere MQ Telemetry
Zabezpečení
Zabezpečení produktu WebSphere MQ Telemetry

Démon IBM WebSphere MQ Telemetry pro koncepce zařízení

Démon IBM WebSphere MQ Telemetry pro zařízení je rozšířenou aplikací klienta MQTT V3 . Použijte jej k ukládání a předávání zpráv z jiných klientů MQTT. Připojuje se k produktu IBM WebSphere MQ jako ke klientovi MQTT, ale k němu se můžete připojit i k dalším klientům MQTT.

Démon je zprostředkovatel publikování a odběru. Klienti MQTT V3 se k němu připojují, aby mohli publikovat a odebírat témata, pomocí řetězců témat pro publikování a filtry témat k odběru. Řetězec tématu je hierarchický, s úrovněmi témat děleno /. Filtry témat jsou řetězce témat, které mohou obsahovat zástupné znaky pro jednu úroveň + a zástupný znak více úrovní # jako poslední část řetězce tématu.

Poznámka: Zástupné znaky v démonu se řídí přísnějšími pravidly produktu WebSphere Message Broker, v6. IBM WebSphere MQ se liší. Podporuje víceúrovňové zástupné znaky více úrovní; zástupné znaky mohou stát v libovolném počtu úrovní hierarchie, kdekoli v řetězci tématu.

K démonu se připojuje více klientů MQTT v3 pomocí portu modulu listener. Výchozí port modulu listener je upravitelný. Můžete definovat více portů modulu listener a přidělit jim různé obory názvů, viz “Démon WebSphere MQ Telemetry pro porty modulu listener zařízení” na stránce 156. Démon je sám o sobě klientem MQTT v3 . Nakonfigurujte připojení mostu démona k připojení démona k portu modulu listener jiného démona nebo ke službě WebSphere MQ Telemetry (MQXR).

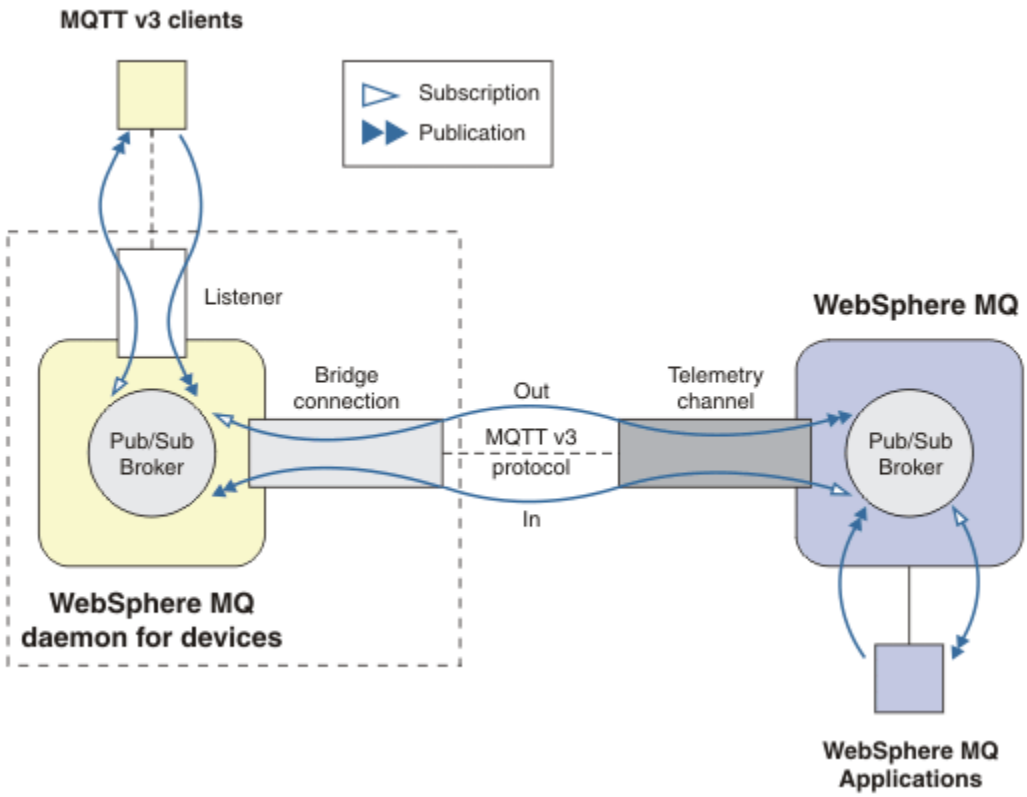
Pro zařízení lze konfigurovat více mostů pro démona WebSphere MQ Telemetry. Mosty slouží k připojení sítě démonů, které si mohou vyměňovat publikace.

Každý most může publikovat a odebírat témata na svém lokálním démonu. Může také publikovat a odebírat témata z jiného démona, zprostředkovatele publikování a odběru WebSphere MQ nebo jakéhokoli jiného zprostředkovatele MQTT v3 , ke kterému je připojen. Pomocí filtru témat můžete vybrat publikace k šíření z jednoho zprostředkovatele do jiného. Publikace lze šířit v obou směrech. Můžete propagovat veřejné služby z lokálního démona do každého z připojených vzdálených zprostředkovatelů nebo z libovolného z připojených zprostředkovatelů na místní démona; viz “Démon IBM WebSphere MQ Telemetry pro zařízení mostů” na stránce 149.

Démon IBM WebSphere MQ Telemetry pro zařízení mostů

Most démona IBM WebSphere MQ Telemetry pro zařízení spojuje dva zprostředkovatele publikování/odběru s použitím protokolu MQTT v3 . Most šíří publikace od jednoho zprostředkovatele do druhého, a to v obou směrech. Na jednom konci je démon WebSphere MQ Telemetry pro připojení mostu zařízení a v druhém případě může být správce front nebo jiný démon. Správce front je připojen k připojení mostu pomocí kanálu telemetrie. Démon je připojen k mostu mostu pomocí modulu listener démona.

Démon IBM WebSphere MQ Telemetry pro zařízení podporuje jedno nebo více souběžných připojení k jiným zprostředkovatelům. Připojení z démona se nazývají mosty a jsou definována položkami připojení v konfiguračním souboru démona. Připojení k produktu IBM WebSphere MQ jsou prováděna pomocí kanálů telemetrie IBM WebSphere MQ , jak je znázorněno na následujícím obrázku:



Obrázek 30. Připojování IBM WebSphere MQ Telemetry daemon for devices k IBM WebSphere MQ

Most připojuje démona k jinému zprostředkovateli jako klient MQTT v3 . Parametry mostu zrcadlí atributy klienta MQTT v3 .

Most je více než připojení. Působí jako agent publikování a odběru umístěný mezi dvěma zprostředkovateli publikování/odběru. Lokální zprostředkovatel je démon IBM WebSphere MQ Telemetry pro zařízení

a vzdálený zprostředkovatel je libovolný zprostředkovatel publikování/odběru, který podporuje protokol MQTT v3 . Vzdáleného zprostředkovatele je obvykle jiný démon nebo IBM WebSphere MQ.

Úloha mostu je pro šíření publikování mezi dvěma zprostředkovateli. Most je obousměrný. Šíří publikace v obou směrech. Obrázek 30 na stránce 149 ilustruje způsob, jakým most připojuje démona IBM WebSphere MQ Telemetry pro zařízení k produktu IBM WebSphere MQ. Produkt “Příklad nastavení tématu pro most” na stránce 150 používá příklady pro ilustraci způsobu použití parametru tématu ke konfiguraci mostu.

Šipky In a Out v Obrázek 30 na stránce 149 označují obousměrný most mostu. Na jednom konci šipky se vytvoří odběr. Publikace, které odpovídají odběru, jsou publikovány do zprostředkovatele na opačném konci šipky. Šipka je označena v závislosti na toku publikací. Publications flow In to the daemon and Out from the daemon. Význam popisků je použit v syntaxi příkazu. Nezapomeňte, že hodnoty In a Out odkazují na tok publikování, a nikoli na místo, kam se má odběr odeslat.

Ostatní klienti, aplikace nebo zprostředkovatelé mohou být připojeni buď k produktu IBM WebSphere MQ , nebo k démonu WebSphere MQ Telemetry pro zařízení. Publikují se a přihlašují se k tématům na zprostředkovateli, ke kterým jsou připojeni. Je-li zprostředkovatel IBM WebSphere MQ, témata mohou být klastrovaná nebo distribuovaná a nejsou explicitně definována v lokálním správci front.

Použití mostů

Spojujte démony společně pomocí připojení mostu a modulů listener. Spojujte démony a správce front dohromady pomocí připojení mostu a kanálů telemetrie. Když spojíte více zprostředkovatelů dohromady, je možné vytvářet smyčky. Buďte opatrní: Publikování by mohly cirkulovat kolem cyklu brokerů, nezjištěné.

Některé z důvodů používání démonů přemostěných do produktu IBM WebSphere MQ jsou následující:

Snižte počet připojení klienta protokolu MQTT k produktu WebSphere MQ .
Pomocí hierarchie démonů můžete připojit mnoho klientů k produktu WebSphere MQ; více klientů než kolik se může připojit k jednomu správci front současně.

Uložit a předat zprávy mezi klienty MQTT a produktem WebSphere MQ
Mezi klienty a produktem IBM WebSphere MQmůžete používat ukládání a přesměrování, pokud klienti nemají své vlastní úložiště. Můžete použít více typů připojení mezi klientem MQTT a produktem WebSphere MQ; viz téma Koncepty a scénáře telemetrie k monitorování a řízení.

Filtrování publikací vyměněných mezi klienty MQTT a produktem WebSphere MQ
Publikace se dělí pouze na zprávy, které se zpracovávají lokálně, a zprávy, které zahrnují jiné aplikace. Lokální publikace mohou zahrnovat řídicí toky mezi senzory a regulátory a vzdálené publikace zahrnují požadavky na odečty, stav a konfigurační příkazy.

Změna prostorů témat publikací
Vyvarujte se řetězcům témat z klientů připojených k různým portům modulu listener z kolidujících s jinými porty modulu listener. Tento příklad používá démona k označení odečtů měřidel pocházejících z různých budov; viz Oddělení prostorů témat různých skupin klientů.

Příklad nastavení tématu pro most

Publikovat vše do vzdáleného zprostředkovatele-použití výchozích hodnot

Výchozí směr se nazývá outa most publikuje témata do vzdáleného zprostředkovatele. Parametr topic určuje, která témata mají být šířena s použitím filtrů témat.

Most používá parametr topic v produktu Obrázek 31 na stránce 151 k přihlášení k odběru všeho, který je publikován na lokální démon klientem MQTT nebo jinými zprostředkovateli. Most publikuje témata ke vzdálenému zprostředkovateli, který je připojen k mostu.

```
connection Daemon1
topic #
```

Obrázek 31. Publikovat vše ve vzdáleném zprostředkovateli

Publikovat vše do vzdáleného zprostředkovatele-explicitní

Nastavení topic v následujícím fragmentu kódu poskytuje stejný výsledek jako použití výchozích nastavení. Jediný rozdíl je, že parametr **direction** je explicitní. Pomocí směru `out` se přihlaste k odběru lokálního zprostředkovatele, démona a publikujte na vzdáleném zprostředkovateli. Publikace vytvořené na lokálním démonu, k jejichž odběru je tento most odebírán, jsou publikovány na vzdáleném zprostředkovateli.

```
connection Daemon1
topic # out
```

Obrázek 32. Publikovat vše do vzdáleného zprostředkovatele-explicitní

Publikovat vše pro lokální zprostředkovatele

Místo použití směru `out`můžete nastavit opačný směr, `in`. Následující fragment kódu konfiguruje most k přihlášení k odběru všeho publikovaného ve vzdáleném zprostředkovateli, který je připojen k mostu. Most publikuje témata do lokálního zprostředkovatele, démon.

```
connection Daemon1
topic # in
```

Obrázek 33. Publikovat vše pro lokální zprostředkovatele

Publikovat vše z tématu exportu na lokálním zprostředkovateli do tématu importu na vzdáleném zprostředkovateli

Pomocí dvou dalších parametrů témat, **local_prefix** a **remote_prefix**, upravte filtr témat, `#` v předchozích příkladech. Jeden parametr se používá k úpravě filtru témat použitého v odběru a druhý parametr se používá k úpravě tématu, na které je publikování publikováno. Výsledkem je nahrazení začátku řetězce tématu použitého v jednom zprostředkovateli jiným řetězcem tématu na druhém zprostředkovateli.

V závislosti na směru příkazu tématu se význam hodnot **local_prefix** a **remote_prefix** vrací. Je-li směr `out`, použije se standardní hodnota **local_prefix** jako část odběru tématu a **remote_prefix** nahradí část **local_prefix** řetězce tématu ve vzdálené publikaci. Je-li směr `in`, stane se **remote_prefix** součástí vzdáleného odběru a **local_prefix** nahradí část **remote_prefix** řetězce tématu.

První část řetězce tématu se často považuje za definici prostoru tématu. Použijte další parametry ke změně prostoru tématu, na který je téma publikováno. Toto můžete provést, chcete-li se vyhnout kolizím tématu s jiným tématem na cílovém zprostředkovateli nebo odebrat řetězec tématu bodu připojení.

Jako příklad, v následujícím fragmentu kódu jsou všechny publikace k řetězci tématu `export/#` v démonu znovu publikovány na `import/#` ve vzdáleném zprostředkovateli.

```
topic # out export/ import/
```

Obrázek 34. Publikovat vše z tématu exportu na lokálním zprostředkovateli do tématu importu na vzdáleném zprostředkovateli

Publikovat vše do tématu importu v lokálním zprostředkovateli z tématu exportu ve vzdáleném zprostředkovateli

Následující fragment kódu ukazuje obrácenou konfiguraci. Most se přihlásí k odběru všeho publikovaného pomocí řetězce tématu produktu `export/#` na vzdáleném zprostředkovateli a publikuje jej na serveru `import/#` na lokálním zprostředkovateli.

```
connection Daemon1
topic # in import/ export/
```

Obrázek 35. Publikovat vše do tématu importu v lokálním zprostředkovateli z tématu exportu ve vzdáleném zprostředkovateli

Publikovat vše z bodu připojení produktu 1884/ do vzdáleného zprostředkovatele s původními řetězci témat

V následujícím fragmentu kódu se most přihlásí k odběru všeho publikovaného klienty připojenými k bodu připojení 1884/ na lokálním démonu. Most publikuje vše publikované na místo připojení ke vzdálenému zprostředkovateli. Řetězec bodu připojení 1884/ je odebrán z témat publikovaných na vzdáleném zprostředkovateli. Hodnota *local_prefix* je stejná jako řetězec bodu připojení 1884/a *vzdálená_předpona* je prázdný řetězec.

```
listener 1884
mount_point 1884/
connection Daemon1
topic # out 1884/ ""
```

Obrázek 36. Publikujte vše z bodu připojení serveru 1884/ do vzdáleného zprostředkovatele s původními řetězci témat.

Oddělování prostorů témat různých klientů připojených k různým démonům

Předpokládejme, že je aplikace zapsána pro elektrická měřidla, aby bylo možné publikovat odečty měřidla pro budovu. Odečty jsou publikovány s použitím klientů MQTT k hostiteli hostovanému ve stejné budově. Téma vybrané pro publikace je `power`. Stejná aplikace je implementována na celou řadu budov v komplexu. Pro monitorování lokalit a datové úložiště jsou odečty ze všech budov agregovány pomocí připojení mostu. Připojení propojují démony budovy s produktem WebSphere MQ v centrálním umístění.

Ve všech budovách se používá identická klientská aplikace. Tato aplikace publikuje na téma `power`. Údaje však musí být rozlišeny podle budovy. Toto se provádí démonem pro každou budovu, která přidává číslo budovy jako předponu názvu tématu. Most z první budovy v komplexu používá předponu `meters/building01/`, od sestavení dvou předpon je `meters/building02/`. Odečty z ostatních budov sledují stejný vzorek. Produkt WebSphere MQ proto přijímá odečty s tématy jako `meters/building01/power`.

Konfigurační soubor pro každý démon má příkaz topic, který následuje za vzorem v následujícím fragmentu kódu:

```
connection Daemon1
topic power out "" meters/building01/
```

Obrázek 37. Oddělit prostory témat klientů připojených k různým démonům

V předchozím fragmentu kódu je prázdný řetězec zástupný symbol pro nepoužívaný parametr local_prefix.

Poznámka: Tento příklad je poněkud umělý a je určen pouze pro ilustraci. V praxi bude prostor tématu, který aplikace publikuje, pravděpodobně konfigurovatelný.

Oddělit prostory témat klientů připojených ke stejnému démonu

Předpokládejme, že k připojení všech měřičů napájení se používá jeden démon. Za předpokladu, že v aplikaci lze nakonfigurovat připojení k různým portům, můžete tyto budovy rozlišit připojením měřidel z různých budov k různým portům modulu listener, jako v následujícím fragmentu kódu. Opět se jedná o příklad, který je možné použít; ilustruje, jak lze použít body připojení.

```
listener 1884
mount_point meters/building01/
listener 1885
mount_point meters/building02/
connection Daemon1
topic meters/+/power out
```

Obrázek 38. Oddělit prostory témat klientů připojených ke stejnému démonu

Přemapování různých témat pro publikace proudící v obou směrech

V konfiguraci v následujícím fragmentu kódu se most přihlásí k odběru jednoho tématu b ze vzdáleného zprostředkovatele a předává publikace o produktu b lokálním démonu, čímž se změní téma na a. Most se také přihlašuje k odběru jediného tématu x na lokálním zprostředkovateli a předává publikace o produktu x vzdáleným zprostředkovatelem, přičemž se změní téma na y.

```
connection Daemon1
topic "" in a b
topic "" out x y
```

Obrázek 39. Přemapování různých témat pro publikace proudící v obou směrech

Důležitým bodem v tomto příkladu je to, že různá témata jsou přihlášena k odběru a publikována v obou zprostředkovatelů. Prostory témat u obou zprostředkovatelů jsou disjunktní.

Přemapování stejných témat u publikací tekoucích v obou směrech (cyklus)

Na rozdíl od předchozího příkladu má konfigurace v produktu Obrázek 40 na stránce 154obecně výsledky ve smyčce. V příkazu tématu `topic "" in a b`se most přihlásí k odběru b vzdáleně a publikuje se lokálně do produktu a . V druhém příkazu tématu se most přihlásí k odběru lokálně z produktu a a publikuje se do produktu b vzdáleně. Stejná konfigurace může být napsána tak, jak je zobrazeno v Obrázek 41 na stránce 154.

Obecný výsledek je takový, že pokud se klient publikuje do b vzdáleně, publikování se přenese na lokální démona jako publikování na téma a. Avšak při publikování z mostu do lokálního démona na téma ase publikování shoduje s odběrem, který byl proveden z mostu do lokálního tématu a. Odběr je `topic "" out a b`. V důsledku toho je publikování převedeno zpět na vzdálený zprostředkovatel jako publikování na téma b. Most je nyní přihlášen k odběru vzdáleného tématu ba cyklus začne znovu.

Někteří zprostředkovatelé implementují detekci smyčky, aby se zabránilo tomu, že se smyčka děje. Ale mechanismus detekce smyčky musí fungovat, když jsou k sobě propojeny různé typy zprostředkovatelů. Detekce smyčky nebude fungovat, pokud je produkt WebSphere MQ přemostěný do démona WebSphere MQ Telemetry pro zařízení. Je funkční, pokud je dva démon IBM WebSphere MQ Telemetry pro zařízení propojen mostem dohromady. Ve výchozím nastavení je detekce smyčky zapnutá; viz try_private.

```
connection Daemon1
topic "" in a b
topic "" out a b
```

Obrázek 40. !Přemapování stejných témat u publikací tekoucích v obou směrech

```
connection Daemon1
topic "" both a b
```

Obrázek 41. !Přemapujte stejná témata pro publikace, které tečou v obou směrech, pomocí produktu `both`.

Konfigurace v produktu Obrázek 39 na stránce 153 je stejná jako Obrázek 40 na stránce 154.

Dostupnost připojení mostu IBM WebSphere MQ Telemetry daemon for devices

Nakonfigurovat více adres připojení mostu IBM WebSphere MQ Telemetry daemon for devices pro připojení k prvnímu dostupnému vzdálenému zprostředkovateli. Je-li zprostředkovatelem správce front s více instancemi, zadejte obě jeho adresy TCP/IP. Nakonfigurujte primární připojení pro připojení nebo opětovné připojení k primárnímu serveru, je-li k dispozici.

Parametr mostu připojení addressesje seznam adres soketů TCP/IP. Most se pokusí o připojení k každé adrese postupně, dokud se nestane úspěšným připojením. Parametry připojení round_robin a start_type řídí, jak se adresy použijí, jakmile se provede úspěšné připojení.

Má-li parametr start_type hodnotu auto, manualnebo lazy, pokus o připojení se nezdaří, pokud se připojení nezdaří, a to se pokusí o nové připojení. Každou adresu používá postupně, s přibližně 20 sekundový prodlevou při každém pokusu o připojení. Má-li parametr `start_type` hodnotu once, nedojde k automatickému pokusu o nové připojení, pokud se připojení nezdaří.

Pokud má parametr round_robin hodnotu true, pokusy o připojení mostu začínají na první adrese v seznamu a v seznamu se pokusí o každou adresu v seznamu. Je-li seznam vyčerpán, spustí se znovu na první adrese. Je-li v seznamu pouze jedna adresa, zkusí ji znovu každých 20 sekund.

Má-li parametr round_robin hodnotu false, je dána přednost první adrese v seznamu, která se nazývá primární server. Pokud se první pokus o připojení k primárnímu serveru nezdaří, bude most pokračovat v pokusu o nové připojení k primárnímu serveru na pozadí. Současně se most pokouší připojit pomocí jiných adres v seznamu. Když se pokusy o připojení k primárnímu serveru pokusí o úspěch, odpojí se most od aktuálního připojení a přepne se na připojení k primárnímu serveru.

Pokud je připojení odpojeno dobrovolně, například zadáním příkazu **`connection_stop`** , pak se znovu pokusí o použití stejné adresy znovu. Je-li připojení odpojeno z důvodu selhání připojení nebo vzdáleného zprostředkovatele, který ruší připojení, most počká 20 sekund. Pak se pokusí připojit k další adrese v seznamu, nebo stejnou adresu, pokud je v seznamu pouze jedna adresa.

Připojení ke správci front s více instancemi

V konfiguraci správce front s více instancemi je správce front spuštěn na dvou různých serverech s různými adresami IP. Kanály telemetrie jsou obvykle konfigurovány bez konkrétní adresy IP. Jsou

konfigurovány pouze s číslem portu. Když je kanál telemetrie spuštěn, standardně vybere první dostupnou síťovou adresu na lokálním serveru.

Konfigurujte parametr addresses v rámci připojení mostu se dvěma adresami IP používanými správcem front. Nastavte parametr round_robin na hodnotu true.

Dojde-li k selhání aktivní instance správce front, správce front se přepne na instanci v pohotovostním režimu. Démon zjistí, že připojení k aktivní instanci se přerušilo a pokouší se znovu navázat spojení s rezervní instancí. Používá druhou adresu IP v seznamu adres konfigurovaných pro připojení mostu.

Správce front, k němuž je připojen most, je stále stejného správce front. Správce front obnoví svůj vlastní stav. Je-li cleansession nastaveno na hodnotu false, relace připojení mostu se obnoví do stejného stavu jako před překonáním selhání. Připojení bude pokračovat po prodlevě. Zprávy, které mají "alespoň jednou" nebo "nejvýše jednou" kvalitu služby, se neztratí a odběry budou pokračovat v práci.

Doba opětovného připojení závisí na počtu kanálů a klientů, které se restartují při spuštění instance v pohotovostním režimu, a počet zpráv, které byly zainfraly. Připojení mostu se může pokusit o opakované připojení k oběma adresám IP před opětným usídlným připojením.

Nekonfigurujte telemetrický kanál správce front s více instancemi s konkrétní adresou IP. Adresa IP je platná pouze na jednom serveru.

Používáte-li alternativní řešení vysoké dostupnosti, které spravuje adresu IP, může být správné konfigurovat kanál telemetrie se specifickou adresou IP.

cleansession

Připojení mostu je relací klienta MQTT v3 . Můžete řídit, zda připojení spustí novou relaci, nebo zda obnoví existující relaci. Pokud obnoví existující relaci, připojení mostu zachová odběry a zachované publikace z předchozí relace.

Nenastavujte cleansession na false, pokud adresa zobrazuje více IP adres a IP adresy se připojují k telemetrickým kanálům hostovaným různými správci front nebo k různým telemetrickým démonům. Stav relace se nepřenáší mezi správci front nebo démony. Pokus o restartování existující relace v jiném správci front nebo démonu vede ke spuštění nové relace. Nejisté zprávy jsou ztraceny a odběry se nemusí chovat podle očekávání.

oznámení

Aplikace může sledovat, zda je připojení mostu spuštěno pomocí oznámení. Oznámení je publikace, která má hodnotu 1, připojená nebo 0, odpojená. Je publikován do souboru *topicString* , který je definován parametrem `notification_topic` . Výchozí hodnota *topicString* je $SYS/`broker/connection/` *clientIdentifier* /`state`. Výchozí hodnota *topicString* obsahuje předponu $SYS. Přihlaste se k odběru témat začínajících řetězcem $SYS definováním filtru tématu začínajícího řetězcem $SYS. Filtr témat *#* se přihlásí k odběru všeho, nepřihlásí se k odběru témat začínajících řetězcem $SYS na démonu. Představte si $SYS jako definování speciálního prostoru tématu systému, který se liší od prostoru tématu aplikace.

Volba Oznámení umožňuje produktu IBM WebSphere MQ Telemetry daemon for devices upozornit klienty protokolu MQTT v případě, že je k mostu připojen nebo odpojen.

keepalive_interval

Parametr připojení mostu keepalive_interval nastavuje interval mezi mostem, který odesílá příkaz ping protokolu TCP/IP na vzdálený server. Výchozí interval je 60 sekund. Testování spojení zabraňuje tomu, aby byla relace TCP/IP uzavřena vzdáleným serverem nebo brána firewall, která zjišťuje dobu nečinnosti připojení.

CLIENTID

Připojení mostu je relací klienta MQTT v3 a má `clientIdentifier` nastavovacího parametru připojení mostu clientid. Chcete-li obnovit předchozí relaci tak, že nastavíte parametr cleansession na hodnotu false, bude hodnota proměnné prostředí `clientIdentifier` použitá v každé relaci shodná. Výchozí hodnota clientid je *hostname.connectionName*, která zůstává stejná.

Instalace, ověření, konfigurace a řízení démona produktu WebSphere MQ Telemetry pro zařízení

Instalace, konfigurace a řízení démona jsou založené na souborech.

Nainstalujte démona zkopírováním sady SDK (Software Development Kit) na zařízení, na kterém chcete spustit démona.

Jako příklad spusťte obslužný program klienta protokolu MQTT a připojte se k démonu WebSphere MQ Telemetry pro zařízení jako zprostředkovatel publikování a odběru. Informace naleznete v tématu Použití démona WebSphere MQ Telemetry pro zařízení jako zprostředkovatele publikování a odběru.

Konfigurujte démona vytvořením konfiguračního souboru, viz WebSphere MQ Telemetry démon pro konfigurační soubor zařízení.

Řídit běžící démona vytvořením příkazů v souboru `amqtdd.upd`. Každých 5 sekund démon čte soubor, spouští příkazy a odstraňuje soubor; viz WebSphere MQ Telemetry démon pro příkazový soubor zařízení.

Démon WebSphere MQ Telemetry pro porty modulu listener zařízení

Připojte klienty MQTT V3 k démonu WebSphere MQ Telemetry pro zařízení používající porty modulu listener. Můžete kvalifikovat port modulu listener s bodem připojení a maximálním počtem připojení.

Port modulu listener musí odpovídat číslu portu uvedenému v metodě klienta MQTT `connect(serverURI)` klienta připojujícího se k tomuto portu. Ve výchozím nastavení je na klientovi i na démona nastaven 1883.

Výchozí port pro démona můžete změnit nastavením globální definice `port` v konfiguračním souboru démona. Specifické porty můžete nastavit přidáním definice `posluchače` do konfiguračního souboru démona.

Pro každý port modulu listener, který není výchozím portem, můžete určit místo připojení pro izolaci klientů. Klienti s připojením k portu s bodem připojení jsou izolováni od ostatních klientů; viz “Démon WebSphere MQ Telemetry pro body připojení zařízení” na stránce 156.

Můžete omezit počet klientů, kteří se mohou připojit k libovolnému portu. Nastavením globální definice `max_connections` omezíte připojení na výchozí port, nebo kvalifikujte každý port modulu listener parametrem `max_connections`.

Příklad

Příklad konfiguračního souboru, který mění výchozí port z 1883 na 1880, a omezuje připojení na port 1880 na 10000. Připojení k portu 1884 jsou omezena na 1000. Klienti připojené k portu 1884 jsou izolováni od klientů připojených k jiným portům.

```
port 1880
max_connections 10000
listener 1884
mount_point 1884/
max_connections 1000
```

Démon WebSphere MQ Telemetry pro body připojení zařízení

K portu modulu listener, který používají klienti MQTT k připojení k démonu WebSphere MQ Telemetry pro zařízení, můžete přidružit bod připojení s portem modulu listener. Bod připojení izoluje publikování a odběry vyměňované klienty MQTT pomocí jednoho portu modulu listener od klientů MQTT připojených k jinému portu modulu listener.

Klienti připojené k portu modulu listener s bodem připojení nemohou nikdy přímo vyměňovat témata s klienty připojenými k žádným jiným portům modulu listener. Klienti připojené k portu modulu listener bez bodu připojení mohou publikovat nebo odebírat témata libovolného klienta. Klienti si nejsou vědomi toho, zda jsou připojeny k bodu připojení, nebo ne; nečiní žádný rozdíl v řetězcích témat vytvořených klienty.

Bod připojení je řetězec textu, který je vložen před řetězec tématu publikování a odběrů. Je předponou o všech řetězcích témat vytvořených klienty připojenými k portu modulu listener s bodem připojení. Řetězec textu je odebrán ze všech řetězců témat odeslaných klientům připojeným k portu modulu listener.

Pokud port modulu listener nemá bod připojení, řetězce témat publikování a odběrů vytvořených a přijímaných klienty připojenými k portu se nezmění.

Vytvořte řetězce bodu připojení s koncovým `/`. Tímto způsobem je bod připojení nadřízeným tématem stromu témat pro bod připojení.

Příklad

Konfigurační soubor obsahuje následující porty modulu listener:

```
listener 1883
mount_point 1883/
listener 1884 127.0.0.1
mount_point 1884/
listener 1885
```

Klient, připojený k portu 1883, vytváří odběr produktu `MyTopic`. Démon registruje odběr jako `1883/MyTopic`. Jiný klient připojený k portu 1883 publikuje zprávu na téma `MyTopic`. Démon změní řetězec tématu na `1883/MyTopic` a vyhledá odpovídající odběry. Odběratel na portu 1883 obdrží publikování s původním řetězcem tématu `MyTopic`. Démon odebral předponu bodu připojení z řetězce tématu.

Jiný klient, připojený k portu 1884, také publikuje na téma `MyTopic`. Tento čas démon registruje téma jako `1884/MyTopic`. Odběratel na portu 1883 neobdržel publikování, protože jiný bod připojení má za následek odběr s jiným řetězcem tématu.

Klient, připojený k portu 1885, publikuje na téma, `1883/MyTopic`. Démon nemění řetězec tématu. Odběratel na portu 1883 přijímá publikování do produktu `MyTopic`.

Démon WebSphere MQ Telemetry pro kvalitu zařízení služby, trvalé odběry a zachované publikace

Nastavení kvality služby se vztahuje pouze na spuštěného démona. Pokud se démon zastaví, ať už řízeným způsobem, nebo kvůli selhání, stav inflatéových zpráv je ztracen. Doručování zprávy alespoň jednou, nebo nanejvýš jednou, nelze zaručit, pokud se démon zastaví. Démon WebSphere MQ Telemetry pro zařízení podporuje omezené trvání. Nastavte konfigurační parametr **`retained_persistence`** pro uložení zachovaných publikování a odběrů, pokud je démon vypnut.

Na rozdíl od produktu WebSphere MQ démon WebSphere MQ Telemetry pro zařízení nežurnálovat trvalá data. Stav relace, stav zprávy a zachované publikace nejsou ukládány transakčně. Démon ve výchozím nastavení vyřazuje všechna data, když se zastaví. Můžete nastavit volbu pro periodické check-check-checkpoint a zachované publikace. Stav zprávy se vždy ztratí, když se démon zastaví. Všechny nezachované publikace jsou ztraceny.

Nastavte volbu konfigurace démona `Retained_persistence` na `true`, chcete-li ukládat zachovaná publikování pravidelně do souboru. Když se démon restartuje, zachované publikace, které byly naposledy uloženy automaticky, byly obnoveny. Ve výchozím nastavení nejsou zachované zprávy vytvořené klienty obnoveny při restartu démona.

Nastavte volbu konfigurace démona `Retained_persistence` na `true`, chcete-li uložit odběry vytvořené v trvalé relaci pravidelně do souboru. Je-li volba `Retained_persistence` nastavena na hodnotu `true`, dojde k obnovení odběrů, které klienti vytvářejí v relaci s volbou `CleanSession` nastaveným na hodnotu `false`, "trvalou relací". Démon obnoví odběry, když se restartuje, což začne přijímat publikování. Klient přijme publikace, když se restartuje s volbou `CleanSession` na `false`. Stav relace klienta se standardně neukládá, když se démon zastaví, a proto se neobnoví odběry, a to ani v případě, že klient nastaví `CleanSession` na `false`.

`Retained_persistence` je mechanizmus automatického ukládání. Nemohou uložit nejnovější zachované publikace nebo odběry. Můžete měnit způsob ukládání zachovaných publikování a odběrů.

Nastavte interval mezi uloženími nebo počet změn mezi uložením, pomocí voleb konfigurace `autosave_on_changes` a `autosave_interval`.

Ukázková konfigurace pro nastavení perzistence

```
# Sample configuration
# Daemon listens on port 1882 with persistence in /tmp
# Autosave every minute
port 1882
persistence_location /tmp/
retained_persistence true
autosave_on_changes false
autosave_interval 60
```

Démon WebSphere MQ Telemetry pro zabezpečení zařízení

Démon WebSphere MQ Telemetry pro zařízení může ověřovat klienty, kteří se k němu připojují, používat pověření k připojení k jiným zprostředkovatelům a řídit přístup k tématům. Zabezpečení démona je omezeno tím, že je sestavováno pomocí klienta WebSphere MQ Telemetry C, který neposkytuje podporu SSL. Připojení k démonu a od něj proto nejsou šifrovány a nelze je ověřovat pomocí certifikátů.

Ve výchozím nastavení není zapnuto žádné zabezpečení.

Ověřování klientů

Klienti MQTT mohou nastavit jméno uživatele a heslo pomocí metod `MqttConnectOptions.setUserName` a `MqttConnectOptions.setPassword`.

Ověřte klienta, který se připojuje k démonu, kontrolou jména uživatele a hesla poskytnutého klientem proti položkám v souboru hesel. Chcete-li povolit ověření, vytvořte soubor hesel a nastavte parametr `password_file` v konfiguračním souboru démona; viz password_file.

Nastavte parametr `allow_anonymous` v konfiguračním souboru démona tak, aby se klienti připojující bez uživatelských jmen nebo hesel připojovali k démonu, který kontroluje ověření; viz allow_anonymous. Pokud klient poskytuje jméno uživatele nebo heslo, je vždy zkontrolováno proti souboru hesel, je-li nastaven parametr `password_file` .

Nastavte parametr `clientity_prefixes` v konfiguračním souboru démona a omezte připojení na konkrétní klienty. Klienti musí mít `clientIdentifiers` , které začínají jednou z předpon uvedených v parametru `clientid_prefixes` ; viz clientid_prefixes.

Zabezpečení připojení mostu

Každý démon WebSphere MQ Telemetry pro připojení mostu zařízení je klient MQTT V3 . Můžete nastavit jméno uživatele a heslo pro každé připojení mostu jako parametr připojení mostu v konfiguračním souboru démona; viz username a password. Most se poté může ověřit u zprostředkovatele.

Řízení přístupu k tématům

Jsou-li klienti ověřováni, může démon také poskytovat řízení přístupu k tématům pro každého uživatele. Démon uděluje řízení přístupu na základě shody s tématem, ke kterému klient buď publikuje, nebo se přihlašuje s řetězcem tématu přístupu v souboru řízení přístupu; viz acl_file.

Seznam přístupových práv má dvě části. První část řídí přístup pro všechny klienty, včetně anonymních klientů. Druhá část má sekci pro libovolného uživatele v souboru hesel. Vypisuje specifické řízení přístupu pro každého uživatele.

Příklad

Parametry zabezpečení jsou zobrazeny v následujícím příkladu.

```
acl_file c:\WMQTDaemon\config\acl.txt
password_file c:\WMQTDaemon\config\passwords.txt
allow_anonymous true
connection Daemon1
username daemon1
password deamonpassword
```

Obrázek 42. Konfigurační soubor démona

```
Fred:Fredpassword
Barney:Barneypassword
```

Obrázek 43. Soubor hesel, passwords.txt

```
topic home/public/#
topic read meters/#
user Fred
topic write meters/fred
topic home/fred/#
user Barney
topic write meters/barney
topic home/barney/#
```

Obrázek 44. Soubor řízení přístupu, acl.txt

Odstraňování problémů klientů MQTT

Podívejte se na úlohu odstraňování problémů, která vám pomůže vyřešit problém se spuštěnými klienty MQTT .

Související úlohy

“Trasování a ladění klienta Java MQTT (Paho)” na stránce 168
Výchozí modul protokolování používá standardní protokolovací zařízení jazyka Java, které je známé jako `java.util.logging` (JSR47). Můžete ji nakonfigurovat buď pomocí konfiguračního souboru, nebo programově.

“Trasování klienta MQTT JavaScript” na stránce 170
Klienta JavaScript můžete použít ke shromažďování trasování tím, že změníte webovou aplikaci klienta na volání metod na připojeném objektu klienta.

“Trasování služby telemetrie (MQXR)” na stránce 163
Postupujte podle těchto pokynů, chcete-li spustit trasování služby telemetrie, nastavit parametry, které řídí trasování, a najít výstup trasování.

“Trasování klienta Java protokolu MQTT v3” na stránce 165
Postupujte podle těchto pokynů, chcete-li vytvořit trasování klienta protokolu MQTT Java a řídit jeho výstup.

“Trasování klienta MQTT pro jazyk C” na stránce 166
Nastavení proměnné prostředí `MQTT_C_CLIENT_TRACE` pro trasování klientské aplikace C klienta MQTT .

“Vyřešení problému: Klient MQTT se nepřipojí” na stránce 177
Vyřešte problém programu klienta MQTT, který selhal při pokusu o připojení ke službě telemetrie (MQXR).

“Problém při řešení problému: Připojení klienta MQTT bylo zrušeno” na stránce 179
Zjistěte, co způsobuje, že klient vyvolal neočekávané výjimky produktu `ConnectionLost` po úspěšném připojení a spuštění buď krátkodobě, nebo dlouho.

“Řešení problému: Ztracené zprávy v aplikaci MQTT” na stránce 180
Vyřešte problém ztráty zprávy. Je zpráva netrvalá, odeslána na nesprávné místo nebo nebyla nikdy odeslána? Nesprávně kódovaný klientský program může ztratit zprávy.

“Řešení problému: Služba telemetrie (MQXR) se nespustí” na stránce 182
Vyřešte problém týkající se spuštění služby telemetrie (MQXR). Zkontrolujte instalaci produktu WebSphere MQ Telemetry a žádné soubory nebyly přesunuty, přesunuty nebo mají chybná oprávnění. Zkontrolujte cesty použité službou telemetrie (MQXR) a vyhledejte servisní programy telemetrie (MQXR).

“Vyřešení problému: přihlašovací modul JAAS , který není volán službou telemetrie” na stránce 183
Zjistěte, zda váš přihlašovací modul JAAS není volán službou telemetrie (MQXR), a nakonfigurujte službu JAAS , chcete-li problém opravit.

“Řešení problému: Spuštění nebo spuštění démona” na stránce 186
Informace o odstraňování problémů s démonem najdete v protokolu démona IBM WebSphere MQ Telemetry pro zařízení konzoly zařízení, zapnutí trasování nebo použití tabulky projevů v tomto tématu.

“Řešení problému: Klienti MQTT se nepřipojují k démonu” na stránce 187
Klienti se k démonu nepřipojují, nebo se démon nepřipojuje k jiným démonům nebo k telemetrickým kanálu WebSphere MQ .

Související odkazy

“Umístění protokolů telemetrie, protokolů chyb a konfiguračních souborů” na stránce 160
Vyhledejte protokoly, protokoly chyb a konfigurační soubory použité produktem IBM WebSphere MQ Telemetry.

“MQTT v3 Kódy příčiny klienta Java” na stránce 162
Vyhledejte příčiny kódů příčiny ve výjimce klienta Java protokolu MQTT v3 nebo v události throwable.

“Systémové požadavky pro použití šifrovacích sad SHA-2 s klienty MQTT” na stránce 171
Pro Java 6 od IBM, SR13 a dále můžete použít šifrovací sady SHA-2 k zabezpečení vašich MQTT kanálů a klientských aplikací. Avšak šifrovací sady SHA-2 nejsou standardně povoleny, dokud Java 7 od IBM, SR4 a dále, takže ve starších verzích musíte uvést požadovanou sadu. Pokud spouštíte klienta MQTT s vlastním prostředím JRE, je třeba zajistit, aby podporoval šifrovací sady SHA-2 . Aby vaše klientské aplikace používaly šifrovací sady SHA-2 , klient musí také nastavit kontext SSL na hodnotu, která podporuje protokol TLS (Transport Layer Security) verze 1.2.

“Omezení v podpoře prohlížečů pro webové aplikace mobilního systému zpráv přes SSL” na stránce 172
Mezi různými prohlížeči existují rozdíly ve schopnosti, na různých platformách. Porozumění těmto rozdílům pomáhá konfigurovat vaše aplikace, vydavatele certifikátů (CA) a certifikáty klientů pro připojení pomocí Klient systému zpráv MQTT pro produkt JavaScript přes SSL a WebSockets.

Umístění protokolů telemetrie, protokolů chyb a konfiguračních souborů

Vyhledejte protokoly, protokoly chyb a konfigurační soubory použité produktem IBM WebSphere MQ Telemetry.

Poznámka: Příklady jsou kódovány pro systém Windows. Změňte syntaxi tak, aby se spustili příklady v systému Linux .

Protokoly na straně serveru

Průvodce instalací produktu IBM WebSphere MQ Telemetry zapisuje zprávy do svého instalačního protokolu:

```
WMQ program directory\mqxr
```

Služba telemetrie (MQXR) zapisuje zprávy do protokolu chyb správce front produktu WebSphere MQ a do souborů FDC do adresáře chyb produktu IBM WebSphere MQ :

```
WMQ data directory\Qmgrs\qMgrName\errors\AMQERR01.LOG
WMQ data directory\errors\AMQnnn.n.FDC
```

Zapíše také protokol pro službu telemetrie (MQXR). Protokol zobrazuje vlastnosti, se kterou služba spustila, a chyby, které nalezl jako zástupce pro klienta MQTT. Například zrušení odběru z odběru, který klient nevytvořil. Cesta k protokolu je:

```
WMQ data directory\Qmgrs\qMgrName\errors\mqxr.log
```

Ukázková konfigurace telemetrie produktu IBM WebSphere MQ vytvořená pomocí produktu IBM WebSphere MQ Explorer spustí službu telemetrie pomocí příkazu **runMQXRService**. **runMQXRService** je v *WMQ Telemetry install directory*\bin. To zapisuje do:

```
WMQ data directory\Qmgrs\qMgrName\mqxr.stdout
WMQ data directory\Qmgrs\qMgrName\mqxr.stderr
```

Upravte **runMQXRService** tak, aby se zobrazovaly cesty konfigurované pro službu telemetrie (MQXR) nebo echo inicializace před spuštěním služby telemetrie (MQXR).

Konfigurační soubory na straně serveru

Kanály telemetrie a služba telemetrie (MQXR)

Omezení: Formát, umístění, obsah a interpretace konfiguračního souboru kanálu telemetrie se mohou v budoucích verzích změnit. Chcete-li konfigurovat kanály telemetrie, je třeba použít produkt IBM WebSphere MQ Explorer.

Produkt IBM WebSphere MQ Explorer uloží konfigurace telemetrie do souboru `mqxr_win.properties` v systému Windowsa soubor `mqxr_unix.properties` v systému Linux. Soubory vlastností se ukládají do konfiguračního adresáře telemetrie:

```
WMQ data directory\Qmgrs\qMgrName\mqxr
```

Obrázek 45. Konfigurační adresář Telemetry v systému Windows

```
/var/mqm/qmgrs/qMgrName/mqxr
```

Obrázek 46. Konfigurační adresář telemetrie v systému Linux

Prostředí JVM

Nastavte vlastnosti jazyka Java, které jsou předávány jako argumenty služby telemetrie (MQXR) v souboru `java.properties`. Vlastnosti v souboru jsou předávány přímo do prostředí JVM se spuštěnou službou telemetrie (MQXR). Tyto vlastnosti jsou předávány jako další vlastnosti JVM na příkazovém řádku Java. Vlastnosti nastavené na příkazovém řádku mají přednost před vlastnostmi, které byly přidány do příkazového řádku ze souboru `java.properties` .

Najděte soubor `java.properties` ve stejné složce jako konfigurace telemetrie, prohlédněte si téma Obrázek 45 na stránce 161 a Obrázek 46 na stránce 161.

Upravte `java.properties` tak, že uvedete každou vlastnost jako samostatný řádek. Naformátujte každou vlastnost přesně tak, jak byste měli předat vlastnost do prostředí JVM jako argument. Příklad:

```
-Xmx1024m
-Xms1024m
```

JAAS

Konfigurační soubor JAAS je popsán v tématu Konfigurace kanálu JAAS kanálu telemetrie , která obsahuje ukázkový konfigurační soubor JAAS , JAAS.config, dodávaný s produktem IBM WebSphere MQ Telemetry.

Pokud nakonfigurujete službu JAAS, téměř jistě budete psát třídu za účelem ověření uživatelů, kteří nahradí standardní procedury ověřování JAAS .

Chcete-li zahrnout třídu `Login` do cesty ke třídám používané v cestě ke třídám služby telemetrie (MQXR), zadejte konfigurační soubor produktu WebSphere MQ `service.env` .

Nastavte cestu ke třídě pro svůj JAAS `LoginModule` v `service.env`. Tuto proměnnou nelze použít, `%classpath%` v `service.env`. Cesta ke třídě v produktu `service.env` je přidána do cesty ke třídě, která je již nastavena v definici služby telemetrie (MQXR).

Zobrazte cesty ke třídám používané službou telemetrie (MQXR) přidáním `echo set classpath` do `runMQXRService.bat`. Výstup se odešle do `mqxr.stdout`.

Výchozí umístění pro soubor `service.env` je:

```
WMQ data directory\service.env
```

Přepište tato nastavení pomocí souboru `service.env` pro každého správce front v následujícím umístění:

```
WMQ data directory\Qmgrs\qMgrName\service.env
```

Obrázek 47 na stránce 162 ukazuje vzorový soubor `service.env` pro použití ukázky `LoginModule.class`.

```
CLASSPATH=WMQ Install Directory\mqxr\samples
```

Poznámka: `service.env` nesmí obsahovat žádné proměnné. Nahraďte skutečnou hodnotu parametru *`WMQ Install Directory`*.

Obrázek 47. Ukázka `service.env` pro Windows

Trasovat
Servisní technik IBM vás může požádat, abyste nakonfigurovali trasování, viz "Trasování služby telemetrie (MQXR)" na stránce 163. Parametry pro konfigurované trasování jsou uloženy ve dvou souborech:

```
WMQ data directory\Qmgrs\qMgrName\mqxr\trace.config
WMQ data directory\Qmgrs\qMgrName\mqxr\mqxrtrace.properties
```

Soubory protokolu na straně klienta

Výchozí třída perzistence souboru v klientovi Java SE MQTT dodávaná s produktem IBM WebSphere MQ Telemetry vytvoří složku s názvem: *clientIdentifier*`-tcp`*hostNameport* nebo *clientIdentifier*`-ssl`*hostNameport* v pracovním adresáři klienta. Název složky určuje `název_hostitele` a `port` použitý při pokusu o připojení. Složka obsahuje zprávy, které byly uloženy třídou perzistence. Zprávy se odstraní, když byly úspěšně doručeny.

Složka se odstraní, když se ukončí klient s vyčištěnou relací.

Je-li trasování klienta zapnuto, neformátovaný protokol je při výchozím nastavení uložen v pracovním adresáři klienta. Trasovací soubor se nazývá `mqtt-`*`n`*`.trc` .

Konfigurační soubory na straně klienta

Nastavte vlastnosti trasování a zabezpečení SSL pro klienta protokolu MQTT Java pomocí souborů vlastností Java nebo nastavte vlastnosti programově. Předejte vlastnosti klientovi MQTT Java pomocí přepínače prostředí JVM `-D` : například,

```
Java -Dcom.ibm.micro.client.mqttv3.trace=c:\\MqttTrace.properties
     -Dcom.ibm.ssl.keyStore=C:\\MyKeyStore.jks
```

Viz téma "Trasování klienta Java protokolu MQTT v3" na stránce 165.Pro odkazy na dokumentaci ke klientským rozhraním API pro knihovny klienta MQTT viz Odkaz na programování klienta MQTT.

MQTT v3 Kódy příčiny klienta Java

Vyhledejte příčiny kódů příčiny ve výjimce klienta Java protokolu MQTT v3 nebo v události throwable.

Tabulka 5. MQTT v3 Kódy příčiny klienta Java		
Kód příčiny	**Hodnota**	**Příčina**
REASON_CODE_BROKER_UNAVAILABLE	3	
REASON_CODE_CLIENT_ALREADY_CONNECTED	32100	Klient je již připojen.
REASON_CODE_CLIENT_ALREADY_DISCONNECTED	32101	Klient je již odpojen.
REASON_CODE_CLIENT_DISCONNECT_PROHIBITED	32107	Vygeneruje se, když byl učiněn pokus o volání `MqttClient.disconnect` z metody na `MqttCallback`.
REASON_CODE_CLIENT_DISCONNECTING	32102	Klient se aktuálně odpojuje a nemůže přijímat žádnou novou práci.
REASON_CODE_CLIENT_EXCEPTION	0	Klient narazil na výjimku.
REASON_CODE_CLIENT_NOT_CONNECTED	32104	Klient není připojen k serveru.
REASON_CODE_CLIENT_TIMEOUT	32000	Vypršel časový limit klienta při čekání na odpověď ze serveru.
REASON_CODE_FAILED_AUTHENTICATION	4	Ověřování na serveru se nezdařilo, kvůli chybnému jménu uživatele nebo heslu.
REASON_CODE_INVALID_CLIENT_ID	2	Server odmítl zadané ID klienta.
REASON_CODE_INVALID_PROTOCOL_VERSION	1	Požadovaná verze protokolu není serverem podporována.
REASON_CODE_NO_MESSAGE_IDS_AVAILABLE	32001	Interní chyba způsobená tím, že nejsou k dispozici žádná nová ID zpráv.
REASON_CODE_NOT_AUTHORIZED	5	Nemáte oprávnění k provedení požadované operace.
REASON_CODE_SERVER_CONNECT_ERROR	32103	Nelze se připojit k serveru.
REASON_CODE_SOCKET_FACTORY_MISMATCH	32105	Identifikátor URI serveru a zadaný SocketFactory se neshodují.
REASON_CODE_SSL_CONFIG_ERROR	32106	Chyba konfigurace SSL.
REASON_CODE_UNEXPECTED_ERROR	6	Došlo k neočekávané chybě.

Trasování služby telemetrie (MQXR)

Postupujte podle těchto pokynů, chcete-li spustit trasování služby telemetrie, nastavit parametry, které řídí trasování, a najít výstup trasování.

Než začnete

Trasování je funkce podpory. Pokud vás servisní technik produktu IBM požádá o trasování služby telemetrie (MQXR), postupujte podle těchto pokynů. Dokumentace produktu nedokumentuje formát trasovacího souboru ani to, jak ji použít k ladění klienta.

Informace o této úloze

Příkazy IBM WebSphere MQ **strmqtrc** a **endmqtrc** můžete použít ke spuštění a zastavení trasování produktu IBM WebSphere MQ . Produkt **strmqtrc** zachycuje trasování pro službu telemetrie (MQXR). Při použití **strmqtrc** dojde k prodlevě až na několik sekund před spuštěním trasování služby telemetrie. Další informace o trasování produktu IBM WebSphere MQ naleznete v tématu Použití trasování. Případně můžete trasovat službu telemetrie (MQXR) pomocí následujícího postupu:

Postup

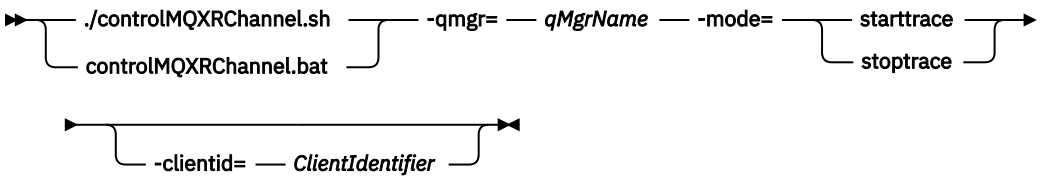
1. Nastavte volby trasování tak, aby bylo možné kontrolovat množství podrobností a velikost trasování. Volby se vztahují na trasování spuštěné buď příkazem **strmqtrc** , nebo příkazem **controlMQXRChannel** .

   Nastavte volby trasování v následujících souborech:

   ```
   mqxrtrace.properties
   trace.config
   ```

   Soubory jsou v adresáři:

   - V systémech Windows, *WebSphere MQ data directory*\qmgrs*qMgrName* \mqxr.
   - V systémech Linux, var/mqm/qmgrs/ *qMgrName*/mqxr.
2. Otevřete příkazové okno v následujícím adresáři:
   - V systémech Windows , *WebSphere MQ installation directory*\mqxr\bin .
   - V systémech Linux /opt/mqm/mqxr/bin.
3. Chcete-li spustit trasování produktu SYSTEM.MQXR.SERVICE , spusťte následující příkaz:



Povinné parametry

qmgr=*qmgrName*
Nastavte parametr *qmgrName* na název správce front.

mode=starttrace| stoptrace
Chcete-li trasování zahájit, nastavte parametr starttrace , chcete-li trasování ukončit, nastavte parametr stoptrace

Nepovinné parametry

clientid=*ClientIdentifier*
Nastavte parametr *ClientIdentifier* na hodnotu ClientIdentifier klienta. clientid filtruje trasování na jednoho klienta. Chcete-li trasovat více klientů, spusťte příkaz trasování vícekrát.

Příklad:

```
/opt/mqm/mqxr/bin/controlMQXRChannel.sh -qmgr=QM1 -mode=starttrace -clientid=
problemclient
```

Výsledky

Chcete-li zobrazit výstup trasování, přejděte do následujícího adresáře:

- V systémech Windows , *WebSphere MQ data directory*\trace.
- V systémech Linux, /var/mqm/trace .

Trasovací soubory jsou pojmenovány mqxr_PPPPP.trc, kde PPPPP je ID procesu.

Související odkazy

strmqtrc

Trasování klienta Java protokolu MQTT v3

Postupujte podle těchto pokynů, chcete-li vytvořit trasování klienta protokolu MQTT Java a řídit jeho výstup.

Než začnete

Toto téma je použitelné pouze pro produkt IBM WebSphere MQ verze 7.5.0.0. Informace popisující trasování klienta Java pro pozdější verze viz “Trasování a ladění klienta Java MQTT (Paho)” na stránce 168.

Trasování je funkce podpory. Postupujte podle těchto pokynů, pokud vás servisní technik produktu IBM požádá o trasování klienta MQTT Java. Dokumentace produktu nedokumentuje formát trasovacího souboru ani to, jak ji použít k ladění klienta.

Trasování pracuje pouze pro klienta Java produktu WebSphere MQ Telemetry .

Informace o této úloze

Poznámka: Příklady jsou kódovány pro Windows. Změňte syntaxi tak, aby se spustili příklady v systému Linux .[2].

Postup

1. Vytvořte soubor vlastností Java obsahující konfiguraci trasování.

   V souboru vlastností uveďte následující volitelné vlastnosti. Je-li klíč vlastnosti zadán více než jednou, nastaví se poslední výskyt vlastnosti.

   a) com.ibm.micro.client.mqttv3.trace.outputName

   Adresář, do kterého má být zapsán trasovací soubor. Výchozí hodnota je pracovním adresářem klienta. Trasovací soubor se nazývá mqtt-*n*.trc.

   ```
   java com.ibm.micro.client.mqttv3.trace.outputName=c:\\MQTT_Trace
   ```

   b) com.ibm.micro.client.mqttv3.trace.count

   Počet trasovacích souborů, které se mají zapsat. Předvolba je jeden soubor, neomezené velikosti.

   ```
   java com.ibm.micro.client.mqttv3.trace.count=5
   ```

   c) com.ibm.micro.client.mqttv3.trace.limit

   Maximální velikost souboru pro zápis, výchozí hodnota je 500000. Limit se použije pouze v případě, že je požadován více než jeden trasovací soubor.

   ```
   java com.ibm.micro.client.mqttv3.trace.limit=100000
   ```

   d) com.ibm.micro.client.mqttv3.trace.client.*clientIdentifier*.status

[2] Java používá správný oddělovač cesty. Oddělovač v souboru vlastností lze kódovat jako '/' nebo '\\'; '\' je řídicí znak.

Zapnout nebo vypnout trasování na klienta. Je-li *clientIdentifier*=*, trasování je zapnuto nebo vypnuto pro všechny klienty. Ve výchozím nastavení je trasování vypnuto pro všechny klienty.

```
java com.ibm.micro.client.mqttv3.trace.client.*.status=on
```

```
java com.ibm.micro.client.mqttv3.trace.client.Client10.status=on
```

2. Předání souboru vlastností trasování do prostředí JVM pomocí systémové vlastnosti.

```
java -Dcom.ibm.micro.client.mqttv3.trace=c:\\MqttTrace.properties
```

3. Spusťte klienta.
4. Převeďte trasovací soubor z binárního kódování na text nebo `.html`. Zadejte následující příkaz:

```
java com.ibm.micro.client.mqttv3.internal.trace.TraceFormatter [-i traceFile] [-o
outputFile] [-h] [-d
time]
```

kde argumenty jsou:

-?
Zobrazí nápovědu.

-i traceFile
Povinné Předává se ve vstupním souboru (například `mqtt-0.trc`).

-o outputFile
Povinné Definuje výstupní soubor (například `mqtt-0.trc.html` nebo `mqtt-0.trc.txt`).

-h
Výstup jako HTML. Přípona výstupních souborů musí být `.html`. Není-li tento parametr zadán, bude výstupem prostý text.

-d time
Odsazuje řádek s hodnotou * , pokud je časový rozdíl v milisekundách větší nebo roven času (> =). Nelze použít pro výstup HTML.

Následující příklad zobrazí trasovací soubor ve formátu HTML

```
java com.ibm.micro.client.mqttv3.internal.trace.TraceFormatter -i mqtt-0.trc -o
mqtt-0.trc.html -h
```

Druhý příklad bude výstupní soubor trasování jako prostý text s libovolnými následnými časovými razítky, které mají milisekundy s rozdílem 50 nebo více s odsazením hvězdičkou (*).

```
java com.ibm.micro.client.mqttv3.internal.trace.TraceFormatter -i mqtt-0.trc -o
mqtt-0.trc.txt -d 50
```

Poslední příklad bude výstupem souboru trasování jako prostý text:

```
java com.ibm.micro.client.mqttv3.internal.trace.TraceFormatter -i mqtt-0.trc -o
mqtt-0.trc.txt
```

Trasování klienta MQTT pro jazyk C

Nastavení proměnné prostředí `MQTT_C_CLIENT_TRACE` pro trasování klientské aplikace C klienta MQTT .

Než začnete

MQTT client for C trace is available for both the pre-built Windows and Linux MQTT client for C libraries, and for the iOS libraries you build yourself.

Informace o této úloze

Nastavte proměnnou prostředí MQTT_C_CLIENT_TRACE tak, aby obsahovala cestu k souboru, který má obsahovat výstup trasování. Výstup trasování je zapsán do souboru.

Postup

Nastavte MQTT_C_CLIENT_TRACE=mqttcclient.log před spuštěním klientské aplikace C MQTT .

a) Například upravte ukázkový skript v produktu “Začínáme s klientem MQTT pro jazyk C” na stránce 25:

```
@echo off
setlocal
set MQTT_C_CLIENT_TRACE=mqttcclient.log
call "C:\Program Files\Microsoft Visual Studio 10.0\VC\vcvarsall.bat" x86
cl  /nologo /D "WIN32" /I "..\include" "MQTTV3Sample.c" /link /
nologo ..\windows_ia32\mqttv3c.lib
set path=%path%;..\windows_ia32;
start "MQTT Subscriber" MQTTV3Sample -a subscribe -b localhost -p 1883
@rem Sleep for 2 seconds
ping -n 2 127.0.0.1 > NUL 2>&1
MQTTV3Sample -b localhost -p 1883
pause
endlocal
```

b) Spusťte skript z adresáře %sdkroot%/sdk/client/c/samples .

Výsledky

Výstupní soubory trasování začínají následujícími řádky:

```
=========================================================
                   Trace Output
=========================================================
19700101 000000.000 (8084)  (1)> Socket_outInitialize:113
19700101 000000.000 (8084)   (2)> SocketBuffer_initialize:81
19700101 000000.000 (8084)   (2)< SocketBuffer_initialize:85
19700101 000000.000 (8084)  (1)< Socket_outInitialize:129
19700101 000000.000 (8084)  (1)> Thread_create_sem:189
19700101 000000.000 (8084)  (1)< Thread_create_sem:222 (0)
19700101 000000.000 (8084)  (1)> Thread_create_sem:189
19700101 000000.000 (8084)  (1)< Thread_create_sem:222 (0)
19700101 000000.000 (8084)  (1)> Thread_create_sem:189
19700101 000000.000 (8084)  (1)< Thread_create_sem:222 (0)
19700101 000000.000 (8084)  (1)> Thread_create_sem:189
19700101 000000.000 (8084)  (1)< Thread_create_sem:222 (0)
19700101 000000.000 (8084)  (1)> MQTTPersistence_create:43
19700101 000000.000 (8084)  (1)< MQTTPersistence_create:89 (0)
19700101 000000.000 (8084)  (1)> MQTTPersistence_initialize:104
19700101 000000.000 (8084)  (1)< MQTTPersistence_initialize:112 (0)
19700101 000000.000 (8084) (0)< MQTTClient_create:267 (0)
19700101 000000.000 (8084) (0)> MQTTClient_connect:701
19700101 000000.000 Connecting to serverURI localhost:1883
20130201 125912.234 (8084)  (1)> MQTTProtocol_connect:93
20130201 125912.234 (8084)   (2)> MQTTProtocol_addressPort:43
20130201 125912.234 (8084)   (2)< MQTTProtocol_addressPort:68
20130201 125912.234 (8084)   (2)> Socket_new:594
20130201 125912.234 New socket 1860 for localhost, port 1883
```

Související úlohy

“Začínáme s klientem MQTT pro jazyk C” na stránce 25
Vstaňte a běží s ukázkovým klientem MQTT pro C na libovolné platformě, na které můžete kompilovat zdroj C. Ověřte, že můžete spustit ukázkového klienta MQTT pro prostředí C s produktem buď IBM MessageSight , nebo IBM WebSphere MQ jako server MQTT.

Trasování a ladění klienta Java MQTT (Paho)

Výchozí modul protokolování používá standardní protokolovací zařízení jazyka Java, které je známé jako `java.util.logging` (JSR47). Můžete ji nakonfigurovat buď pomocí konfiguračního souboru, nebo programově.

Informace o této úloze

Poznámka: Klient služby Paho Java lze použít pouze pro verze produktu IBM WebSphere MQ 7.5.0.1 a novější. Informace popisující trasování klienta Java v produktu IBM WebSphere MQ verze 7.5.0.0viz “Trasování klienta Java protokolu MQTT v3” na stránce 165.

Poznámka: Trasování je funkce podpory. Pokud vás servisní technik IBM požádá o trasování klienta MQTT Java, postupujte podle těchto pokynů. Dokumentace produktu nedokumentuje formát trasovacího souboru ani to, jak ji použít k ladění klienta. Trace only works for the IBM WebSphere MQ Telemetry Java client.

Nejjednodušší metodou použití konfiguračního souboru je uvedení jeho názvu ve vlastnosti `java.util.logging.config.file`.

Soubor pracovních vlastností `jsr47min.properties` je poskytován v balíku `org.eclipse.paho.client.mqttv3.logging`

Zařízení protokolování JSR47 lze použít v mnoha ohledech:

- Chcete-li shromažďovat zprávy z vybrané sady balíků, postupujte takto:
- Chcete-li shromažďovat zprávy z úrovně protokolování a pod úrovní protokolu
- Chcete-li vybrat více míst určení pro zprávy protokolu,
- Poskytnutím vestavěného zapisovače protokolu, který zapisuje do souboru a řídí velikost a počet použitých souborů.
- Poskytnutím zabudovaného zapisovače protokolu, který zapisuje do paměti a umožňuje zápisu do zpráv v paměti na základě aktivační události
- Je-li aplikace, která používá knihovnu klienta MQTT, instrumentována pomocí JSR47, jsou zprávy z aplikace a knihovny klienta promíchány

Je poskytnuta třída obslužného programu, která pomáhá shromažďovat ladicí informace. Tato třída zahrnuje zprávy protokolu a trasování, které jsou popsány dříve, ale mohou shromažďovat informace, jako jsou systémové vlastnosti Java a hodnoty proměnných zevnitř klienta Paho.

Poskytovaná služba ladění je poskytována v rámci veřejné třídy Debug, která je součástí balíku `org.eclipse.paho.client.mqttv3.util`. Instance ladění může být získána pomocí metody getDebug() na asynchronních a synchronních objektech klienta MQTT.

Příklad:

```
MqttClient cl = new MqttClient();
Debug d = cl.getDebug();
```

Metoda dumpClientDebug () vypíše maximální množství ladicích informací. Musí být povoleno protokolovací zařízení, aby bylo možné zachytit úplné informace o ladění, které jsou zapsány do protokolu. Chcete-li zachytit úplné informace ladění, vyvolejte metodu výpisu paměti, pokud je známo, že se problém vyskytuje, například po výskytu určité výjimky.

Postup

1. Vytvořte konfigurační soubor nebo použijte dodaný soubor jsr47min.properties .

   Používáte-li poskytnutý soubor vlastností, zkontrolujte, zda je spouštěč nastaven na správnou úroveň chyby. Při výchozím nastavení je tato chyba nastavena na závažnou chybu úrovně, ale může být

nezbytné průběžné zapisování trasování do souboru místo jeho uchování v paměti, dokud nedojde k chybě. Chcete-li to provést, změňte:

```
java.util.logging.MemoryHandler.push=SEVERE
```

na

```
java.util.logging.MemoryHandler.push=ALL
```

2. Předání konfiguračního souboru trasování do prostředí JVM pomocí systémové vlastnosti.

   Používáte-li soubor jsr4min.properties , jedná se o následující:

```
java -Djava.util.logging.config.file=C:\temp\jsr47min.properties
```

3. Spusťte klienta.

Výsledky

Když se vyskytne výjimka nebo problém, třída Debug Paho zapíše do nakonfigurovaného cíle souboru trasování v paměti.

Trasování se do souboru automaticky nezapíše, protože je generováno, a to pouze v případě, že je použit spouštěč typu push nebo třída ladění způsobí zápis do trasování. Ty mohou vyžadovat změny v kódu aplikace.

Každý řádek je zapsán do popisovače souboru, jak je vytvořen. Formát, ve kterém se zprávy odepisují, můžete řídit konfigurací FileHandler. Vlastní popisovač souboru je poskytnut s Paho, který zapisuje více než SimpleHandler a je menší než popisovač XMLHandler poskytnutý s prostředím JRE. Trasovací záznamy, které používají formátovač protokolu Paho, jsou v následujícím tvaru:

```
Level   Data and Time   Class   Method   Thread   clientID   Message
```

Příklad

K dispozici je pracovní soubor vlastností `jsr47min.properties` . Tento soubor obsahuje navrženou konfiguraci pro shromažďování trasování, která pomáhá řešit problémy související s klientem protokolu MQTT Paho. Nakonfiguruje trasování tak, aby se průběžně shromažďoval v paměti s minimálním dopadem na výkon. Když se vyskytne spouštěč nebo specifický požadavek na odeslání typu push, trasování v paměti se odešle do konfigurované obslužné rutiny cíle. Výchozí spouštěč typu push je zpráva se závažnou úrovní, která je přerušeným připojením. Při výchozím nastavení je trasování, které je shromážděno v paměti, zapisováno do zadaného souboru v tomto bodě. Standardně je tento soubor standardním `java.util.logging.FileHandler`. K odeslání trasování paměti do cíle můžete použít třídu paho Debug.

Úplné podrobnosti o JSR47 lze nalézt v dokumentaci Javadoc pro balík java.util.logging.

```
# Loggers
# ------------------------------------------
# A memory handler is attached to the Paho packages
# and the level specified to collect all trace related
# to Paho packages.  This will override any root/global
# level handlers if set.
org.eclipse.paho.client.mqttv3.handlers=java.util.logging.MemoryHandler
org.eclipse.paho.client.mqttv3.level=ALL
# It is possible to set more granular trace on a per class basis e.g.
#org.eclipse.paho.client.mqttv3.internal.ClientComms.level=ALL

# Handlers
# ------------------------------------------
# Note: the target handler that is associated with the Memory Handler is not a root handler
# and hence not returned when getting the handlers from root. It appears accessing
# target handler programmatically is not possible as target is a private variable in
# class MemoryHandler
java.util.logging.MemoryHandler.level=FINEST
java.util.logging.MemoryHandler.size=10000
java.util.logging.MemoryHandler.push=SEVERE
```

```
java.util.logging.MemoryHandler.target=java.util.logging.FileHandler
#java.util.logging.MemoryHandler.target=java.util.logging.ConsoleHandler

# --- FileHandler ---
# Override of global logging level
java.util.logging.FileHandler.level=ALL

# Naming style for the output file:
# (The output file is placed in the directory
# defined by the "user.home" System property.)
# See java.util.logging for more options
java.util.logging.FileHandler.pattern=%h/paho%u.log

# Limiting size of output file in bytes:
java.util.logging.FileHandler.limit=200000

# Number of output files to cycle through, by appending an
# integer to the base file name:
java.util.logging.FileHandler.count=3

# Style of output (Simple or XML):
java.util.logging.FileHandler.formatter=org.eclipse.paho.client.mqttv3.logging.SimpleLogFormatte
r
```

Jak pokračovat dále

Chcete-li shromažďovat trasování programově, je poskytnuta třída obslužného programu, která pomáhá shromažďovat informace ladění. Tato třída zahrnuje zprávy protokolu a trasování, které jsou popsány dříve, ale mohou shromažďovat informace, jako jsou systémové vlastnosti Java a hodnoty proměnných zevnitř klienta Paho.

Poskytovaná služba ladění je poskytována v rámci veřejné třídy Debug, která je součástí balíku `org.eclipse.paho.client.mqttv3.util`. Instance ladění může být získána pomocí metody getDebug() na asynchronních a synchronních objektech klienta MQTT.

Příklad:

```
MqttClient cl = new MqttClient();
Debug d = cl.getDebug();
```

Metoda dumpClientDebug () vypíše maximální množství ladicích informací. Musí být povoleno protokolovací zařízení, aby bylo možné zachytit úplné informace o ladění, které jsou zapsány do protokolu. Chcete-li zachytit úplné informace ladění, vyvolejte metodu výpisu paměti, pokud je známo, že se problém vyskytuje, například po výskytu určité výjimky.

Trasování klienta MQTT JavaScript

Klienta JavaScript můžete použít ke shromažďování trasování tím, že změníte webovou aplikaci klienta na volání metod na připojeném objektu klienta.

Informace o této úloze

Chcete-li shromáždit trasování, můžete použít následující metody:

- Příkaz `client.startTrace()` spustí trasování pro klienta.
- Produkt `client.stopTrace()` zastaví trasování pro klienta.
- Příkaz `client.getTraceLog()` vrací aktuální vyrovnávací paměť pro trasování.

Trasovací vyrovnávací paměť můžete odeslat na adresu IBM Software Support. Existuje řada způsobů, jak to udělat. Příklad ukazuje, jak je spuštěno trasování, potom výstup odeslaný na konzolu a zadanou e-mailovou adresu a nakonec je trasování zastavováno.

```
client = new Messaging.Client(location.hostname, Number(location.port), "clientId");
// Start the client tracing, the trace records capture the method calls and network
//flows from now on.
client.startTrace();

client.onConnectionLost = onConnectionLost;
```

```
client.connect({onSuccess:onConnect});

function onConnect() {
  console.log("onConnect, will now disconnect then email Trace");
  client.disconnect();
};
function onConnectionLost(responseObject) {
  if (responseObject.errorCode !== 0)
    console.log("ConnectionLost:"+responseObject.errorMessage);
    console.log(client.getTraceLog());
    window.location="mailto:helpdesk@"+location.hostname+
        "?Subject=Web%20Messaging%20Utility%20Trace&body="+
        client.getTraceLog().join("%0A");
    client.stopTrace();
 };
```

Ukázkový výstup:

```
Client.startTrace, "2013-10-03T10:58:10.531Z", "0.0.0.0",
Client.connect, {"keepAliveInterval":60,"cleanSession":true},, false,
Client._socket_send, {"type":1,"keepAliveInterval":60,"cleanSession":true,
    "clientId":"clientId"},
Client._on_socket_message, {},
Client._on_socket_message, {"type":2,"topicNameCompressionResponse":0,"returnCode":0},
Client.disconnect,Client._socket_send, {"type":14},
Client.getTraceLog, "2013-10-03T10:58:10.548Z",
Client.getTraceLog in flight messages,
```

V 7.5.0.2 Systémové požadavky pro použití šifrovacích sad SHA-2 s klienty MQTT

Pro Java 6 od IBM, SR13 a dále můžete použít šifrovací sady SHA-2 k zabezpečení vašich MQTT kanálů a klientských aplikací. Avšak šifrovací sady SHA-2 nejsou standardně povoleny, dokud Java 7 od IBM, SR4 a dále, takže ve starších verzích musíte uvést požadovanou sadu. Pokud spouštíte klienta MQTT s vlastním prostředím JRE, je třeba zajistit, aby podporoval šifrovací sady SHA-2 . Aby vaše klientské aplikace používaly šifrovací sady SHA-2 , klient musí také nastavit kontext SSL na hodnotu, která podporuje protokol TLS (Transport Layer Security) verze 1.2.

Pro prostředí Java 7 z produktu IBMverze SR4 je při výchozím nastavení povoleno použití šifer SHA-2 . Pokud pro prostředí Java 6 z produktů IBM, SR13 a pozdějších verzí služeb definujete kanál MQTT bez určení šifrovací sady, nebude kanál přijímat připojení od klienta s použitím šifrovací sady SHA-2 . Chcete-li použít šifrovací sady SHA-2 , musíte v definici kanálu uvést požadovanou sadu. Díky tomu server MQTT povolí tuto sadu před vytvořením připojení. Znamená to také, že se k tomuto kanálu mohou připojit pouze aplikace klienta používající uvedenou sadu.

Pro Klient MQTT pro produkt Java existuje podobné omezení. Pokud je kód klienta spuštěn na prostředí JRE Java 1.6 z produktu IBM, musí být požadované šifrovací sady SHA-2 explicitně povoleno. Chcete-li použít tyto sady, musí klient také nastavit kontext zabezpečení SSL na hodnotu, která podporuje verzi 1.2 protokolu TLS (Transport Layer Security). Příklad:

```
MqttConnectOptions mqttConnectOptions = new MqttConnectOptions();
java.util.Properties sslClientProps = new java.util.Properties();
sslClientProps.setProperty("com.ibm.ssl.keyStore", sslKeys.clientKeyStore);
sslClientProps.setProperty("com.ibm.ssl.keyStorePassword", sslKeys.clientStorePassword);
sslClientProps.setProperty("com.ibm.ssl.trustStore", sslKeys.clientKeyStore);
sslClientProps.setProperty("com.ibm.ssl.trustStorePassword", sslKeys.clientStorePassword);
sslClientProps.setProperty("com.ibm.ssl.protocol", "TLSv1.2");
sslClientProps.setProperty("com.ibm.ssl.enabledCipherSuites",
"SSL_RSA_WITH_AES_256_CBC_SHA256" );
mqttConnectOptions.setSSLProperties(sslClientProps);
```

Stejně jako v červnu 2013 je Internet Explorer 10 jediným prohlížečem, který pracuje s protokolem Klient systému zpráv MQTT pro produkt JavaScript a také podporuje protokol TLS 1.2 , takže je jediným prohlížečem, který můžete použít, pokud chcete vytvořit připojení SHA-2 s klientem JavaScript .

Seznam šifrovacích sad, které jsou momentálně podporovány, najdete v souvisejících odkazech.

Související pojmy

“Konfigurace klienta MQTT pro ověření klienta pomocí SSL” na stránce 104

Chcete-li ověřit klienta MQTT pomocí protokolu SSL, připojí se klient k kanálu telemetrie pomocí zabezpečení SSL. Musí určovat port TCP, který odpovídá kanálu telemetrie, který je konfigurován pro ověřování klientů SSL.

“Konfigurace klienta MQTT pro ověření kanálu pomocí zabezpečení SSL” na stránce 107
Chcete-li ověřit kanál telemetrie pomocí zabezpečení SSL, musí se klient připojit ke kanálu telemetrie pomocí zabezpečení SSL. Musí určovat port, který odpovídá kanálu telemetrie konfigurovanému pro zabezpečení SSL. Konfigurace musí obsahovat úložiště klíčů chráněné heslem, které obsahuje soukromě podepsaný digitální certifikát serveru.

V 7.5.0.1 Omezení v podpoře prohlížečů pro webové aplikace mobilního systému zpráv přes SSL

Mezi různými prohlížeči existují rozdíly ve schopnosti, na různých platformách. Porozumění těmto rozdílům pomáhá konfigurovat vaše aplikace, vydavatele certifikátů (CA) a certifikáty klientů pro připojení pomocí Klient systému zpráv MQTT pro produkt JavaScript přes SSL a WebSockets.

Mobilní posílání zpráv pomocí produktu JavaScript přes SSL je celkem nové, takže není překvapivé, že různé kombinace prohlížeče a platformy implementovaly schopnost různými způsoby a s různými oblastmi. Následující tabulka poskytuje přehled o tom, co v současné době funguje, a nefunguje pro každou kombinaci prohlížeče (Firefox, Chrome, Internet Explorera Safari) a platformy (Windows, Linux, Mac, iOSa Android).

Tabulka 6. Podpora zabezpečení SSL podle platformy a prohlížeče. Pro každou kombinaci prohlížeče a platformy tabulka uvádí, zda jsou podporována anonymní a neanonymní připojení SSL a rozsah, ve kterém prohlížeč pracuje se všemi CA (CA) a certifikáty klienta.

Prohlížeč	**Podpora SSL (A/N)**	**SSL pracuje s jakoukoli CA (A/N)**	**Další informace**
Firefox .	Anonymní SSL-Ano SSL bez anonymního uživatele-Ano	Anonymní SSL-Ano SSL bez anonymního uživatele-Ano	Přidejte CA a certifikát klienta do prohlížeče. Produkt Firefox používá své vlastní úložiště certifikátů. Chcete-li importovat certifikát CA, klepněte na volbu **Nástroje** > **Volby** > **Rozšířené** > **Šifrování** > **Zobrazit certifikáty** > **Oprávnění** > **Importovat** . Chcete-li importovat certifikát klienta, klepněte na volbu **Nástroje** > **Volby** > **Rozšířené** > **Šifrování** > **Zobrazit certifikáty** > **Vaše certifikáty** > **Importovat** . Chcete-li povolit zabezpečené připojení, zadejte `https://` do adresy URL. Firefox vám dává možnost vybrat si certifikát automaticky nebo se dotáže pokaždé, když se vás pokaždé zeptá. Produkt Firefox vám také poskytuje možnost použití SSL 3.0 nebo TLS 1.0; ujistěte se, že jsou vybrány obě volby.

Tabulka 6. Podpora zabezpečení SSL podle platformy a prohlížeče. Pro každou kombinaci prohlížeče a platformy tabulka uvádí, zda jsou podporována anonymní a neanonymní připojení SSL a rozsah, ve kterém prohlížeč pracuje se všemi CA (CA) a certifikáty klienta. *(pokračování)*

Prohlížeč	**Podpora SSL (A/N)**	**SSL pracuje s jakoukoli CA (A/N)**	**Další informace**
Chrome .	Anonymní SSL-Ano SSL bez anonymního uživatele-Ano	Anonymní SSL-Ano SSL bez anonymního uživatele-Ano	Použijte prohlížeč k přidání certifikátu CA a klienta do úložiště certifikátů operačního systému, které je sdíleno s jiným softwarem. Chcete-li importovat certifikát CA, klepněte na volbu **Nastavení** > **Zobrazit rozšířená nastavení** > **Správa certifikátů** > **Důvěryhodné kořenové certifikační autority** > **Importovat** . Chcete-li importovat certifikát klienta, klepněte na volbu **Nastavení** > **Zobrazit rozšířená nastavení** > **Správa certifikátů** > **Osobní** > **Importovat** . Chcete-li povolit zabezpečené připojení, zadejte `https://` do adresy URL. Chrome vás vyzve k zadání několika voleb; vyberte jednu z nich, v závislosti na tom, zda konfigurujete anonymní nebo neanonymní připojení.

Tabulka 6. Podpora zabezpečení SSL podle platformy a prohlížeče. Pro každou kombinaci prohlížeče a platformy tabulka uvádí, zda jsou podporována anonymní a neanonymní připojení SSL a rozsah, ve kterém prohlížeč pracuje se všemi CA (CA) a certifikáty klienta. *(pokračování)*

Prohlížeč	**Podpora SSL (A/N)**	**SSL pracuje s jakoukoli CA (A/N)**	**Další informace**
Internet Explorer.	Anonymní SSL-Ano SSL bez anonymního uživatele-Ano	Anonymní SSL-Ano SSL bez anonymního uživatele-Ano	Při vytvoření neanonymního připojení SSL se zobrazí výzva k výběru správného certifikátu klienta. Produkt Internet Explorer používá úložiště certifikátů produktu Windows , které je sdíleno s jiným softwarem. Chcete-li importovat certifikát CA, klepněte na volbu **Nástroje** > **Možnosti Internetu** > **Obsah** > **Certifikáty** > **Důvěryhodné kořenové certifikační autority** > **Importovat** . Chcete-li importovat certifikát klienta, klepněte na volbu **Nástroje** > **Možnosti Internetu** > **Obsah** > **Certifikáty** > **Osobní** > **Importovat** .
Safari .	Anonymní SSL-Ano SSL bez anonymního uživatele-Ano	Anonymní SSL-Ano SSL bez anonymního uživatele-Ano	Použijte prohlížeč k přidání certifikátu CA a klienta do úložiště certifikátů operačního systému, které je sdíleno s jiným softwarem.

Tabulka 6. Podpora zabezpečení SSL podle platformy a prohlížeče. Pro každou kombinaci prohlížeče a platformy tabulka uvádí, zda jsou podporována anonymní a neanonymní připojení SSL a rozsah, ve kterém prohlížeč pracuje se všemi CA (CA) a certifikáty klienta. *(pokračování)*

Prohlížeč	Podpora SSL (A/N)	SSL pracuje s jakoukoli CA (A/N)	Další informace
Modul plug-in Firefox na serveruAndroid	Anonymní SSL-Ano SSL bez anonymního uživatele-Ano	Anonymní SSL-Ano SSL bez anonymního uživatele-Ne	Neanonymní: Klientské certifikáty nepracují, protože nemůžete splnit požadavek na přidání vašeho CA do seznamu v produktu Firefox. Chcete-li importovat certifikát klienta, klepněte na nabídku **Nastavení** > **Zabezpečení** > **Úložiště pověření**. Je-li váš certifikát podepsán důvěryhodným CA v seznamu, můžete vytvořit zabezpečené připojení.
Modul plug-in Chrome na serveruAndroid	Anonymní SSL-Ano SSL bez anonymního uživatele-Ano	Anonymní SSL-Ano SSL bez anonymního uživatele-Ne	Neanonymní: Klientské certifikáty nepracují, protože nemůžete splnit požadavek na přidání vašeho CA do seznamu v produktu Chrome. **Poznámka:** Plán Google jej podporuje ve verzi 27 produktu Chrome. Jedná se o otevřený defekt od verze 18. Chcete-li importovat certifikát klienta, klepněte na nabídku **Nastavení** > **Zabezpečení** > **Úložiště pověření**. Je-li váš certifikát podepsán důvěryhodným CA v seznamu, můžete vytvořit zabezpečené připojení.

Tabulka 6. Podpora zabezpečení SSL podle platformy a prohlížeče. Pro každou kombinaci prohlížeče a platformy tabulka uvádí, zda jsou podporována anonymní a neanonymní připojení SSL a rozsah, ve kterém prohlížeč pracuje se všemi CA (CA) a certifikáty klienta. *(pokračování)*

Prohlížeč	Podpora SSL (A/N)	SSL pracuje s jakoukoli CA (A/N)	Další informace
Modul plug-in Safari na serveruiOS	Anonymní SSL-Ano SSL bez anonymního uživatele-Ano	Anonymní SSL-Ano SSL bez anonymního uživatele-Ne	Neanonymní: Zařízení nedůvěřuje certifikátu klienta ani v případě, že je certifikát CA nainstalován ve stejnou dobu. Produkt Safari používá úložiště certifikátů zařízení. Chcete-li provést import do tohoto úložiště, klepněte na nabídku **Nastavení** > **Obecné** > **Profil**a nadejte certifikát CA nebo klienta z webové stránky, nebo jej odešlete e-mailem.
Modul plug-in Chrome na serveruiOS	Anonymní SSL-Ano SSL bez anonymního uživatele-Ne	Anonymní SSL-Ne SSL bez anonymního uživatele-Ne	Anonymní: Pouze aplikace Apple mohou přistupovat ke kořenovému úložišti systému iOS . Proto musí produkt Chrome používat svůj vlastní seznam CA, ke kterému nemůžete přidávat. Neanonymní: Klientské certifikáty nepracují, protože nemůžete splnit požadavek na přidání vašeho CA do seznamu.

Související úlohy

“Připojení Klient systému zpráv MQTT pro produkt JavaScript přes SSL a WebSockets” na stránce 77
Webovou aplikaci bezpečně připojte k produktu IBM WebSphere MQ pomocí ukázkových HTML stránek Klient systému zpráv MQTT pro produkt JavaScript s SSL a WebSocket protocol.

Související informace

Mozilla: (SSL) Prohlížeč Firefox používá úložiště CA Android nebo jeho vlastní úložiště?
Chrom: Problém 134418-Implementace podpory klientských certifikátů
Nelze otevřít web https s nedůvěryhodným certifikátem na ie10

Vyřešení problému: Klient MQTT se nepřipojí

Vyřešte problém programu klienta MQTT, který selhal při pokusu o připojení ke službě telemetrie (MQXR).

Než začnete

Je problém na serveru, na klientovi nebo s připojením? Napsali jste vlastní klienta pro zpracování protokolu MQTT v3 nebo klientskou aplikaci MQTT pomocí klientů MQTT jazyka C nebo jazyka Java WebSphere ?

Spusťte aplikaci pro ověření dodávanou s produktem WebSphere MQ Telemetry na serveru a zkontrolujte, zda jsou správně spuštěny kanály telemetrie a služba telemetrie (MQXR). Poté přeneste aplikaci ověření na klienta a spusťte ji tam.

Informace o této úloze

Existuje několik důvodů, proč se klient MQTT nemusí připojit, nebo k němu můžete dojít k závěru, že k serveru telemetrie není připojen.

Postup

1. Zvažte, jaké chyby lze vyvodit z kódu příčiny, který služba telemetrie (MQXR) vrátila do produktu `MqttClient.Connect`. Jaký typ selhání připojení je to?

Volba	Popis
REASON_CODE_INVALID_PROTOCOL_VERSION	Ujistěte se, že adresa soketu odpovídá kanálu telemetrie, a že jste nepoužili stejnou adresu soketu pro jiného zprostředkovatele.
REASON_CODE_INVALID_CLIENT_ID	Zkontrolujte, zda identifikátor klienta není delší než 23 bajtů a že obsahuje pouze znaky z rozsahu: `A-Z, a-z, 0-9, './_%`
REASON_CODE_INVALID_DESTINATION	Zkontrolujte, že identifikátor klienta není stejný jako název správce front.
REASON_CODE_SERVER_CONNECT_ERROR	Zkontrolujte, zda je služba telemetrie (MQXR) a správce front normálně spuštěny. Použijte **netstat** k ověření, že adresa soketu není přidělena jiné aplikaci.

   Pokud jste místo použití jedné z knihoven poskytnutých produktem IBM WebSphere MQ Telemetry napsali knihovnu klienta protokolu MQTT, podívejte se na návratový kód produktu `CONNACK` .

   Z těchto tří chyb můžete odvodit, že klient se připojil ke službě telemetrie (MQXR), ale služba nalezla chybu.

2. Zvažte, jaké chyby lze vyvodit z kódů příčiny, které klient produkuje, když služba telemetrie (MQXR) neodpovídá.

Volba	Popis
REASON_CODE_CLIENT_EXCEPTION **REASON_CODE_CLIENT_TIMEOUT**	Vyhledejte soubor FDC na serveru, viz “Protokoly na straně serveru” na stránce 160. Když služba telemetrie (MQXR) zjistí, že klient vypršel časový limit, zapíše soubor FDC (Data Capture) prvního selhání. Zapisuje soubor FDC vždy, když se neočekávaně přeruší spojení.

   Služba telemetrie (MQXR) pravděpodobně na klienta neodpověděla a vypršení časového limitu na klientovi vyprší. Klient jazyka Java produktu WebSphere MQ Telemetry se zablokuje pouze v případě, že aplikace má nastaven neomezený časový limit. Klient vygeneruje jednu z těchto výjimek po vypršení časového limitu pro `MqttClient.Connect` s nediagnostikovaným problémem s připojením.

Pokud nenaleznete soubor FDC, který koreluje se selháním připojení, nemůžete odvodit, že se klient pokusil připojit k serveru:

a) Potvrďte, že klient odeslal požadavek na připojení.

Zkontrolujte požadavek protokolu TCPIP pomocí nástroje, jako je například **tcpmon**, dostupného na adrese https://java.net/projects/tcpmon .

b) Je adresa vzdáleného soketu používaná klientem shodná s adresou soketu definovanou pro kanál telemetrie?

Výchozí třída perzistence souboru v klientovi Java SE MQTT dodávaná s produktem IBM WebSphere MQ Telemetry vytvoří složku s názvem: *clientIdentifier*-tcp*hostNameport* nebo *clientIdentifier*-ssl*hostNameport* v pracovním adresáři klienta. Název složky určuje `název_hostitele` a `port` použitý při pokusu o připojení.; viz “Soubory protokolu na straně klienta” na stránce 162.

c) Je možné odeslat příkaz ping na adresu vzdáleného serveru?

d) Prokáže produkt **netstat** na serveru kanál telemetrie v portu, ke kterému se klient připojuje?

3. Zkontrolujte, zda služba telemetrie (MQXR) nalezla v požadavku klienta problém.

Služba telemetrie (MQXR) zapisuje chyby, které zjistí, do produktu `mqxr.log`, a správce front zapisuje chyby do produktu `AMQERR01.LOG`; viz

4. Pokuste se izolovat problém spuštěním jiného klienta.

- Spusťte ukázkovou aplikaci MQTT pomocí stejného kanálu telemetrie.
- Spusťte klienta grafického uživatelského rozhraní produktu **wmqttSample** , abyste ověřili připojení. Získejte produkt **wmqttSample** stažením balíku SupportPac IA92.

Poznámka: Starší verze produktu IA92 nezahrnují knihovnu klienta protokolu MQTT v3 Java.

Spusťte vzorové programy na platformě serveru, abyste vyloučili nejistotu týkající se síťového připojení, pak spusťte ukázky na platformě klienta.

5. Další věci ke kontrole:

a) Jsou desítky tisíc klientů MQTT pokoušeli se připojit zároveň?

Kanály telemetrie mají frontu na vyrovnávací paměť nevyřízených příchozích připojení. Spojení jsou zpracovávána více než 10 000 za sekundu. Velikost vyrovnávací paměti nevyřízených požadavků lze konfigurovat pomocí průvodce kanálem telemetrie v Průzkumníku IBM WebSphere MQ . Jeho výchozí velikost je 4096. Zkontrolujte, zda nevyřízené požadavky nebyly nakonfigurovány na nízkou hodnotu.

b) Je služba telemetrie (MQXR) a správce front stále spuštěny?

c) Byl klient připojen ke správci front vysoké dostupnosti, který přepnul jeho adresu TCPIP?

d) Je brána firewall selektivně filtrováním odchozích nebo návratových datových paketů?

Problém při řešení problému: Připojení klienta MQTT bylo zrušeno

Zjistěte, co způsobuje, že klient vyvolal neočekávané výjimky produktu `ConnectionLost` po úspěšném připojení a spuštění buď krátkodobě, nebo dlouho.

Než začnete

Klient MQTT se úspěšně připojil. Klient může být dlouho vzhůru. Pokud mezi sebou klienti začínají pouze krátkým intervalem, může být doba mezi úspěšně připojováním a uvolněným spojením krátká.

Není těžké rozlišit zrušené připojení od spojení, které bylo úspěšně vytvořeno, a pak později upuštěno. Zahozené připojení je definováno klientem MQTT, který volá metodu `MqttCallback.ConnectionLost` . Metoda je volána až po úspěšném navázání spojení. Příznakem se liší od `MqttClient.Connect` po vyvolání výjimky po přijetí negativního potvrzení nebo vypršení časového limitu.

Pokud aplikace klienta protokolu MQTT nepoužívá knihovny klienta MQTT dodané produktem IBM WebSphere MQ, tento symptom závisí na klientovi. V protokolu MQTT v3 je symptom nedostatek včasné odezvy na požadavek na server nebo selhání připojení TCP/IP.

Informace o této úloze

Klient MQTT volá `MqttCallback.ConnectionLost` s výjimkou typu throwable v odpovědi na jakékoli problémy na straně serveru, které se vyskytly po přijetí pozitivního potvrzení připojení. Když se klient MQTT vrátí z produktů `MqttTopic.publish` a `MqttClient.subscribe` , je požadavek přenesen na podproces klienta MQTT, který je odpovědný za odesílání a příjem zpráv. Chyby na straně serveru se nahlašují asynchronně předáním výjimky throwable do metody zpětného volání `ConnectionLost` .

Služba telemetrie (MQXR) vždy zapíše soubor zachycení dat prvního selhání, pokud dojde k jeho zrušení.

Postup

1. Byl spuštěn jiný klient, který použil stejný identifikátor `ClientIdentifier`?

   Je-li spuštěn druhý klient nebo je restartován stejný klient, použije se stejné `ClientIdentifier`, první připojení k prvnímu klientovi se zruší.
2. Má klient přístup k tématu, které nemá autorizaci pro publikování nebo odběr?

   Veškeré akce, které služba telemetrie provede jménem klienta, který vrací výsledek `MQCC_FAIL` ve službě, zruší připojení klienta.

   Kód příčiny se pro klienta nevrací.

   - Vyhledejte zprávy protokolu v souborech `mqxr.log` a `AMQERR01.LOG` pro správce front, ke kterému je klient připojen; viz “Protokoly na straně serveru” na stránce 160.
3. Bylo uvolněno připojení TCP/IP?

   Brána firewall může mít nízké nastavení časového limitu pro označení připojení TCPIP jako neaktivní a zrušení připojení.

   - Zkraťte neaktivní dobu připojení TCPIP pomocí voleb `MqttConnectOptions.setKeepAliveInterval`.

Řešení problému: Ztracené zprávy v aplikaci MQTT

Vyřešte problém ztráty zprávy. Je zpráva netrvalá, odeslána na nesprávné místo nebo nebyla nikdy odeslána? Nesprávně kódovaný klientský program může ztratit zprávy.

Než začnete

Jak jste si jistý, že ta zpráva, kterou jste poslal, byla ztracena? Můžete odvodit, že zpráva je ztracená, protože zpráva nebyla přijata? Je-li zpráva publikace, která je ztracena: zpráva odeslaná vydavatelem nebo zpráva odesílaná odběrateli? Nebo došlo ke ztrátě odběru a zprostředkovatel neodesílá publikování pro tento odběr odběrateli?

Pokud řešení zahrnuje distribuované publikování/odběr, použití klastrů nebo hierarchií publikování/odběru, existuje celá řada problémů s konfigurací, které by mohly vyústit ve ztrátu zprávy.

Pokud jste odeslali zprávu "Nejméně jednou" nebo "Nejvýše jednou" kvalitu služby, je pravděpodobné, že zpráva, kterou si myslíte, nebyla ztracena, nebyla doručena způsobem, který jste očekávali. Je nepravděpodobné, že byla zpráva nesprávně odstraněna ze systému. Je možné, že došlo k selhání při vytváření publikování nebo odběru, který jste očekávali.

Nejdůležitějším krokem při určování příčin problémů ztracených zpráv je potvrzení ztráty zprávy. Znovu vytvořte scénář a ztratíte další zprávy. Použijte "Nejméně jednou" nebo "Nejvýše jednou" kvalitu služby, abyste vyloučili všechny případy zahození zpráv do systému.

Informace o této úloze

Existují čtyři nohy pro diagnostiku ztracené zprávy.

1. Zprávy "Fire and forget" pracují tak, jak jsou navrženy. Zprávy "Fire and forget" jsou někdy systémem vyřazeny.
2. Konfigurace: nastavení publikování/odběru se správnými oprávněními v distribuovaném prostředí není jednoduché.
3. Chyby programování klienta: odpovědnost za doručení zprávy není výhradně odpovědností kódu, který je napsán společností IBM.
4. Pokud jste vyčerpali všechny tyto možnosti, můžete se rozhodnout zahrnout službu IBM .

Postup

1. Pokud byla ztracená zpráva "Fire and forget" ("Fire and forget"-quality of service), nastavte "At least once" nebo "At most once" ("At most once"). Pokuste se o ztrátu zprávy znovu.
   - Zprávy odeslané s "Fire and forget" quality of service are thrown away by IBM WebSphere MQ in a number of circumstances:
     - Ztráta komunikací a kanál byl zastaven.
     - Správce front byl vypnut.
     - Nadměrný počet zpráv.
   - Dodávka zpráv "Fire and forget" závisí na spolehlivosti TCP/IP. TCP/IP pokračuje v posílání datových paketů znovu, dokud není potvrzeno jejich doručení. Je-li relace TCP/IP přerušena, budou ztraceny zprávy s kvalitou služby "Fire and forget". Relace může být přerušena zavřením klienta nebo serveru, komunikačním problémem nebo bránou firewall odpojením relace.
2. Zkontrolujte, zda klient znovu spouští předchozí relaci, aby bylo možné odesílat nedoručené zprávy s kvalitou služby "Nejméně jednou" nebo "At most once" (Alespoň jednou) nebo "At most once" (At most once).
   a) Pokud klientská aplikace používá klienta Java SE MQTT, zkontrolujte, že nastavuje `MqttClient.CleanSession` na `false` .
   b) Používáte-li různé knihovny klienta, zkontrolujte, zda je relace restartována správně.
3. Zkontrolujte, zda aplikace klienta znovu spouští stejnou relaci, a ne se spuštěním jiné relace omylem.

   Chcete-li spustit stejnou relaci znovu, `cleanSession = false`a `Mqttclient.clientIdentifier` a `MqttClient.serverURI` , musí být stejná jako předchozí relace.
4. Dojde-li k předčasnému ukončení relace, zkontrolujte, zda je zpráva v úložišti perzistence k dispozici v klientu k odeslání znovu.
   a) Pokud klientská aplikace používá klienta Java SE MQTT, zkontrolujte, zda je zpráva uložena do složky perzistence, viz “Soubory protokolu na straně klienta” na stránce 162 .
   b) Používáte-li různé knihovny klienta, nebo jste implementovali vlastní mechanismus perzistence, zkontrolujte, zda pracuje správně.
5. Zkontrolujte, že nikdo neodstranil zprávu před jejím dodáním.

   Nedoručené zprávy čekající na doručení klientům MQTT jsou uloženy v produktu `SYSTEM.MQTT.TRANSMIT.QUEUE`. Zprávy čekající na doručení do serveru telemetrie jsou ukládány mechanizmem perzistence klienta. Další informace naleznete v tématu Trvání zpráv v klientech produktu MQTT.
6. Zkontrolujte, zda má klient odběr pro publikování, které očekává přijetí.

   Seznam odběrů pomocí programu Průzkumník produktu WebSphere MQ nebo pomocí příkazů **runmqsc** nebo PCF. Všechny odběry klienta MQTT mají název. Zobrazí se název formuláře: *`ClientIdentifier`*:*`Topic name`*
7. Zkontrolujte, zda má vydavatel oprávnění k publikování, a odběratel, abyste se mohli přihlásit k odběru tématu publikování.

```
dspmqaut -m qMgr -n topicName -t topic -p user ID
```

V systému klastrovaných publikování/odběr musí být odběratel autorizován pro téma ve správci front, k němuž je odběratel připojen. Odběratel není nutný k odběru informací o odběru tématu ve správci front, kde je publikování publikováno. Kanály mezi správci front musí být správně autorizovány pro předání odběru proxy a předávání této publikace.

Vytvořte stejný odběr a publikujte jej pomocí Průzkumníka IBM WebSphere MQ . Simulujte klienta aplikace a přihlašujte se k odběru pomocí obslužného programu klienta. Spusťte obslužný program z produktu IBM WebSphere MQ Explorer a změňte jeho ID uživatele tak, aby se shodovalo s ID uživatele přijatým klientskou aplikací.

8. Zkontrolujte, zda má odběratel oprávnění k vložení publikace do produktu SYSTEM.MQTT.TRANSMIT.QUEUE.

```
dspmqaut -m qMgr -n queueName -t queue -p user ID
```

9. Zkontrolujte, zda má aplikace typu point-to-point IBM WebSphere MQ oprávnění k umístění své zprávy na server SYSTEM.MQTT.TRANSMIT.QUEUE.

```
dspmqaut -m qMgr -n queueName -t queue -p user ID
```

Viz "Odeslání zprávy klientovi přímo" v tématu Konfigurace distribuovaných front pro odesílání zpráv na klienty MQTT.

Řešení problému: Služba telemetrie (MQXR) se nespustí

Vyřešte problém týkající se spuštění služby telemetrie (MQXR). Zkontrolujte instalaci produktu WebSphere MQ Telemetry a žádné soubory nebyly přesunuty, přesunuty nebo mají chybná oprávnění. Zkontrolujte cesty použité službou telemetrie (MQXR) a vyhledejte servisní programy telemetrie (MQXR).

Než začnete

Funkce produktu WebSphere MQ Telemetry je nainstalována. Průzkumník IBM WebSphere MQ má složku Telemetrie v adresáři **IBM WebSphere MQ > Správci front > *qMgrName* > Telemetry**. Pokud složka neexistuje, instalace se nezdařila.

Služba telemetrie (MQXR) musí být vytvořena, aby mohla být spuštěna. Pokud nebyla služba telemetrie (MQXR) vytvořena, spusťte příkaz **Definovat ukázkovou konfiguraci ...** ve složce Telemetry .

Pokud byla služba telemetrie (MQXR) spuštěna dříve, pak jsou ve složce Telemetry vytvořeny další složky **Kanály** a **Stav kanálu** . Služba Telemetrie, SYSTEM.MQXR.SERVICE, se nachází ve složce **Služby** . Je viditelný, pokud je klepnuto na přepínač Průzkumník k zobrazení systémových objektů.

Klepněte pravým tlačítkem myši na SYSTEM.MQXR.SERVICE , abyste spustili a ukončili službu, zobrazili jeho stav a zobrazili, zda má vaše ID uživatele oprávnění ke spuštění služby.

Informace o této úloze

Spuštění služby telemetrie SYSTEM.MQXR.SERVICE (MQXR) se nezdařilo. Selhání při spouštění souborů typu manifest se projevuje dvěma různými způsoby:

1. Příkaz pro spuštění se okamžitě nezdaří.
2. Spuštění příkazu je úspěšné a je okamžitě následováno zastavením služby.

Postup

1. Spustit službu

Výsledek

Služba se zastaví okamžitě. V okně se zobrazí chybová zpráva, například:

```
WebSphere MQ cannot process the request because the
executable specified cannot be started. (AMQ4160)
```

Příčina

Soubory chybí v instalaci, nebo jsou oprávnění pro instalované soubory nastavena chybně.
Funkce produktu IBM WebSphere MQ Telemetry je instalována pouze na jednom z dvojice vysoce dostupných správců front. Pokud se instance správce front přepne na rezervní databázi, pokusí se spustit příkaz SYSTEM.MQXR.SERVICE. Příkaz pro spuštění služby selže, protože služba telemetrie (MQXR) není nainstalována v rezervní databázi.

vyšetřování
Podívejte se do protokolů chyb, viz “Protokoly na straně serveru” na stránce 160.

Akce
Nainstalujte nebo odinstalujte a znovu nainstalujte funkci WebSphere MQ Telemetry.

2. Spusťte službu; počkejte 30 sekund; obnovte průzkumníka a zkontrolujte stav služby.

Výsledek
Služba se spustí a poté se zastaví.

Příčina
SYSTEM.MQXR.SERVICE spustil příkaz **runMQXRService** , ale příkaz selhal.

vyšetřování

Podívejte se do protokolů chyb, viz “Protokoly na straně serveru” na stránce 160.
Zjistěte, zda k problému dochází pouze s definovaným vzorkovým kanálem. Zálohujte a vymažte obsah adresáře *WMQ data directory*\Qmgrs*qMgrName*\mqxr\ . Spusťte ukázkového průvodce konfigurací a pokuste se spustit službu.

Akce
Hledejte oprávnění a problémy s cestami.

Vyřešení problému: přihlašovací modul JAAS , který není volán službou telemetrie

Zjistěte, zda váš přihlašovací modul JAAS není volán službou telemetrie (MQXR), a nakonfigurujte službu JAAS , chcete-li problém opravit.

Než začnete

Upravujete *WMQ installation directory*\mqxr\samples\LoginModule.java pro vytvoření své vlastní třídy ověření *WMQ installation directory*\mqxr\samples\samples\LoginModule.class. Případně jste napsali vlastní třídy ověření JAAS a umístili je do adresáře dle vašeho výběru. Po určitém počátečním testování s použitím služby telemetrie (MQXR) se domníváte, že vaše třída ověření není volána službou telemetrie (MQXR).

Poznámka: Ochrana proti možnosti, že by vaše ověřovací třídy mohly být přepsány údržbou, která se používá pro produkt WebSphere MQ. Použijte vlastní cestu pro třídy ověření, spíše než cestu v adresářovém stromu produktu WebSphere MQ .

Informace o této úloze

Úloha používá scénář k ilustraci způsobu řešení problému. Ve scénáři obsahuje balík s názvem security.jaas třídu ověření JAAS s názvem JAASLogin.class. Je uložen v cestě C:\WMQTelemetryApps\security\jaas. Nápovědu ke konfiguraci služby JAAS pro produkt IBM WebSphere MQ Telemetry naleznete v tématu Konfigurace služby JAAS kanálu telemetrie .
Příklad: “Příklad konfigurace JAAS” na stránce 184 je vzorová konfigurace.

Postup

1. V produktu `mqxr.log` se podívejte na výjimku vyvolanou produktem `javax.security.auth.login.LoginException`.

   Viz "Protokoly na straně serveru" na stránce 160 pro cestu k produktu `mqxr.log`a Obrázek 54 na stránce 186 pro příklad výjimky uvedené v protokolu.
2. Opravte konfiguraci služby JAAS porovnáním s uvedeným příkladem v produktu "Příklad konfigurace JAAS" na stránce 184.
3. Nahraďte svou třídu přihlášení ukázkou produktu `JAASLoginModule`po opětovné deklaraci do svého balíku ověření a implementace ji pomocí stejné cesty. Přepněte hodnotu `loggedIn` mezi `true` a `false`.

   Pokud problém zmizí, když `loggedIn` je `true`a vypadá to samé, když `loggedIn` je `false`, problém se nachází ve vaší třídě přihlášení.
4. Zkontrolujte, zda je problém spíše s autorizací než ověřením.
   a) Změňte definici kanálu telemetrie, aby provedla kontrolu autorizace pomocí pevného ID uživatele. Vyberte ID uživatele, které je členem skupiny `mqm` .
   b) Znovu spusťte aplikaci klienta.

      Pokud problém zmizí, řešení spočívá v tom, že ID uživatele je předáno k autorizaci. Co je předáváno jméno uživatele? Vytiskněte ji do souboru z přihlašovacího modulu. Zkontrolujte svá přístupová oprávnění pomocí programu Průzkumník IBM WebSphere MQ nebo **dspmqauth**.

Příklad konfigurace JAAS

Prostřednictvím průvodce **Nový kanál telemetrie** v produktu WebSphere MQ Explorer můžete konfigurovat kanál telemetrie. Klient se připojuje k portu 1884 a připojuje se k telemetrickým kanálu produktu `JAASMCAUser` . Obrázek 48 na stránce 184 ukazuje příklad souboru vlastností telemetrie vytvořeného pomocí průvodce telemetrie. Neupravujte tento soubor přímo. Kanál ověřuje pomocí služby JAASkonfiguraci s názvem `JAASConfig`. Jakmile je klient ověřen, použije ID uživatele `Admin` k autorizaci jeho přístupu k objektům produktu IBM WebSphere MQ .

```
com.ibm.mq.MQXR.channel/JAASMCAUser: \
com.ibm.mq.MQXR.Port=1884;\
com.ibm.mq.MQXR.JAASConfig=JAASConfig;\
com.ibm.mq.MQXR.UserName=Admin;\
com.ibm.mq.MQXR.StartWithMQXRService=true
```

Obrázek 48. WMQ Installation directory\data\qmgrs\qMgrName\mqxr\mqxr_win.properties

Konfigurační soubor JAAS má oddíl s názvem `JAASConfig` , který pojmenovává třídu Java `security.jaas.JAASLogin`, kterou má produkt JAAS použít k ověřování klientů.

```
JAASConfig {
  security.jaas.JAASLogin required debug=true;
};
```

Obrázek 49. WMQ Installation directory\data\qmgrs\qMgrName\mqxr\jaas.config

Když se produkt `SYSTEM.MQTT.SERVICE` spustí, přidá cestu do Obrázek 50 na stránce 185 ke své cestě ke třídě.

```
CLASSPATH=C:\WMQTelemtryApps;
```

Obrázek 50. WMQ Installation directory\data\qmgrs\qMgrName\service.env

Obrázek 51 na stránce 185 ukazuje dodatečnou cestu v produktu Obrázek 50 na stránce 185 přidaná k cestě ke třídě, která je nastavena pro službu telemetrie (MQXR).

```
CLASSPATH=;C:\IBM\MQ\Program\mqxr\bin\\..\lib\MQXRListener.jar;
C:\IBM\MQ\Program\mqxr\bin\\..\lib\WMQCommonServices.jar;
C:\IBM\MQ\Program\mqxr\bin\\..\lib\objectManager.utils.jar;
C:\IBM\MQ\Program\mqxr\bin\\..\lib\com.ibm.micro.xr.jar;
C:\IBM\MQ\Program\mqxr\bin\\..\..\java\lib\com.ibm.mq.jmqi.jar;
C:\IBM\MQ\Program\mqxr\bin\\..\..\java\lib\com.ibm.mqjms.jar;
C:\IBM\MQ\Program\mqxr\bin\\..\..\java\lib\com.ibm.mq.jar;
C:\WMQTelemtryApps;
```

Obrázek 51. Výstup cesty ke třídě ze souboru runMQXRService.bat

Výstup z produktu Obrázek 52 na stránce 185 ukazuje, že služba telemetrie (MQXR) byla spuštěna s definicí kanálu zobrazenou v části Obrázek 48 na stránce 184.

```
21/05/2010 15:32:12 [main] com.ibm.mq.MQXRService.MQXRPropertiesFile
AMQXR2011I: Property com.ibm.mq.MQXR.channel/JAASMCAUser value
com.ibm.mq.MQXR.Port=1884;
com.ibm.mq.MQXR.JAASConfig=JAASConfig;
com.ibm.mq.MQXR.UserName=Admin;
com.ibm.mq.MQXR.StartWithMQXRService=true
```

Obrázek 52. WMQ Installation directory\data\qmgrs\qMgrName\errors\mqxr.log

Když se aplikace klienta připojí k kanálu JAAS , pokud `com.ibm.mq.MQXR.JAASConfig=JAASWrongConfig` neodpovídá názvu sekce JAAS v souboru `jaas.config` , připojení selže a klient vyvolá výjimku s návratovým kódem `0`; viz Obrázek 53 na stránce 186. Druhá výjimka `Client is not connected (32104)` byla vyvolána, protože se klient pokusil o odpojení, když se nepřipojil.

```
C:\WMQTelemetryApps>java com.ibm.mq.id.PubAsyncRestartable
Starting a clean session for instance "Admin_PubAsyncRestartab"
Publishing "Hello World Fri May 21 17:23:23 BST 2010" on topic "MQTT Example"
for client instance: "Admin_PubAsyncRestartab" using QoS=1 on address tcp://localhost:1884"
Userid: "Admin", Password: "Password"
Delivery token "528752516" has been received: false
Connection lost on instance "Admin_PubAsyncRestartab" with cause "MqttException"
MqttException (0) - java.io.EOFException
        at com.ibm.micro.client.mqttv3.internal.CommsReceiver.run(CommsReceiver.java:118)
        at java.lang.Thread.run(Thread.java:801)
Caused by: java.io.EOFException
        at java.io.DataInputStream.readByte(DataInputStream.java:269)
        at
com.ibm.micro.client.mqttv3.internal.wire.MqttInputStream.readMqttWireMessage(MqttInputStream.ja
va:56)
        at com.ibm.micro.client.mqttv3.internal.CommsReceiver.run(CommsReceiver.java:90)
        ... 1 more
Client is not connected (32104)
        at
com.ibm.micro.client.mqttv3.internal.ExceptionHelper.createMqttException(ExceptionHelper.java:33
)
        at com.ibm.micro.client.mqttv3.internal.ClientComms.internalSend(ClientComms.java:100)
        at com.ibm.micro.client.mqttv3.internal.ClientComms.sendNoWait(ClientComms.java:117)
        at com.ibm.micro.client.mqttv3.internal.ClientComms.disconnect(ClientComms.java:229)
        at com.ibm.micro.client.mqttv3.MqttClient.disconnect(MqttClient.java:385)
        at com.ibm.mq.id.PubAsyncRestartable.main(PubAsyncRestartable.java:49)
```

Obrázek 53. Došlo k výjimce při připojování com.ibm.mq.id.PubAsyncRestartable

`mqxr.log` obsahuje další výstup zobrazený v Obrázek 53 na stránce 186.

Chyba byla zjištěna pomocí JAAS , která vyvolává `javax.security.auth.login.LoginException` s příčinou `No LoginModules configured for JAAS`. Příčinou může být chybný název konfigurace, jako například Obrázek 54 na stránce 186, nesprávné konfigurační jméno. Může to být také výsledek jiných problémů JAAS , který narazil na načtení konfigurace JAAS .

Pokud obslužný program JAASnenahlásí žádnou výjimku, služba JAAS úspěšně načte třídu `security.jaas.JAASLogin` pojmenovanou ve stanze `JAASConfig` .

```
21/05/2010 12:06:12 [ServerWorker0] com.ibm.mq.MQXRService.MQTTCommunications
AMQXR2050E: Unable to load JAAS config: JAASWrongConfig.
The following exception occurred javax.security.auth.login.LoginException:
No LoginModules configured for JAAS
```

Obrázek 54. `mqxr.log` -chyba při načítání konfigurace JAAS

Řešení problému: Spuštění nebo spuštění démona

Informace o odstraňování problémů s démonem najdete v protokolu démona IBM WebSphere MQ Telemetry pro zařízení konzoly zařízení, zapnutí trasování nebo použití tabulky projevů v tomto tématu.

Postup

1. Zkontrolujte protokol konzoly.

   Je-li démon spuštěn na popředí, jsou zprávy konzoly zapsány do okna terminálu. Pokud byl démon spuštěn na pozadí, konzola je místo, kam jste přesměrovali `stdout` .
2. Restartujte démona.

   Změny konfiguračního souboru se neaktivují, dokud se démon nerestartuje.
3. Konzultujte Tabulka 7 na stránce 187:

Tabulka 7. Tabulka projevů

Problém	Doporučené řešení

Když spustíte démona na systému Windows, zobrazí se následující zpráva:
`Systém nemůže provést uvedený program`
, nebo
`Spuštění aplikace se nezdařilo protože jeho konfigurace na straně po straně je nesprávná.`
	Nainstalujte balík Microsoft Visual C++ 2008 Redistributable Package.

Dva nebo více démonů nebo serverů s podporou MQTT jsou navzájem propojeny mostem nebo mostem a procesor vykazuje nadměrné zatížení.	Existuje pravděpodobně smyčka zprávy s jednou či více zprávami opakovaně předávanými z jednoho serveru do jiného. Prověřte parametry tématu v konfiguračních souborech. Pokud je to možné, použijte specifičtější témata. Široké zástupné znaky v obou směrech jsou nejčastějším důvodem smyček připojení.
Most se nemůže připojit ke vzdálenému serveru schopnému MQTT, ke kterému se mohou připojit další klienti MQTT.	Je možné, že vzdálený server není kompatibilní s pokusy o určení, zda je vzdálený server také démon WebSphere MQ Telemetry pro zařízení. Chcete-li zakázat speciální zpracování za účelem odstranění smyček zpráv, zkuste nastavit volbu **`try_private`** na hodnotu `off` .
Tato zpráva se vytiskne, když je konfigurován most: `Varování: Připojení nebylo první paket na soketu 1888, byl přijat CONNACK.`	Pravděpodobně jste konfigurovali most, aby se vrátil zpět na lokální démona. Zpětná smyčka není podporována.

Řešení problému: Klienti MQTT se nepřipojují k démonu

Klienti se k démonu nepřipojují, nebo se démon nepřipojuje k jiným démonům nebo k telemetrickým kanálu WebSphere MQ .

Informace o této úloze

Trasovat každý paket MQTT odeslaný a přijatý démonem.

Postup

Nastavte parametr **`trace_output`** na hodnotu `protocol` v konfiguračním souboru démona nebo odešlete příkaz démonovi pomocí souboru `amqtdd.upd` .

Příklad použití souboru `amqtdd.upd` naleznete v tématu Přenos zpráv mezi démonem IBM WebSphere MQ Telemetry pro zařízení a IBM WebSphere MQ .

Při použití nastavení protokolu démon vytiskne zprávu na konzolu s popisem každého paketu MQTT, který odesílá a přijímá.

Poznámky

Tyto informace byly vyvinuty pro produkty a služby poskytované v USA.

Společnost IBM nemusí nabízet produkty, služby nebo funkce uvedené v tomto dokumentu v jiných zemích. Informace o produktech a službách, které jsou ve vaší oblasti aktuálně dostupné, získáte od místního zástupce společnosti IBM. Odkazy na produkty, programy nebo služby společnosti IBM v této publikaci nejsou míněny jako vyjádření nutnosti použití pouze uvedených produktů, programů či služeb společnosti IBM. Místo toho lze použít jakýkoli funkčně ekvivalentní produkt, program nebo službu, které neporušují žádná práva k duševnímu vlastnictví IBM. Ověření funkčnosti produktu, programu nebo služby pocházející od jiného výrobce je však povinností uživatele.

Společnost IBM může vlastnit patenty nebo nevyřízené žádosti o patenty zahrnující předměty popsané v tomto dokumentu. Vlastnictví tohoto dokumentu neposkytuje licenci k těmto patentům. Dotazy týkající se licencí můžete posílat písemně na adresu:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Odpovědi na dotazy týkající se licencí pro dvoubajtové znakové sady (DBCS) získáte od oddělení IBM Intellectual Property Department ve vaší zemi, nebo tyto dotazy můžete zasílat písemně na adresu:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

Následující odstavec se netýká Velké Británie nebo kterékoliv jiné země, kde taková opatření odporují místním zákonům: SPOLEČNOST INTERNATIONAL BUSINESS MACHINES CORPORATION TUTO PUBLIKACI POSKYTUJE TAKOVOU, "JAKÁ JE" , BEZ JAKÝCHKOLIV ZÁRUK, VYJÁDŘENÝCH VÝSLOVNĚ NEBO VYPLÝVAJÍCÍCH Z OKOLNOSTÍ, VČETNĚ, A TO ZEJMÉNA, ZÁRUK NEPORUŠENÍ PRÁV TŘETÍCH STRAN, PRODEJNOSTI NEBO VHODNOSTI PRO URČITÝ ÚČEL VYPLÝVAJÍCÍCH Z OKOLNOSTÍ. Některé právní řády u určitých transakcí nepřipouštějí vyloučení záruk výslovně vyjádřených nebo vyplývajících z okolností, a proto se na vás toto omezení nemusí vztahovat.

Uvedené údaje mohou obsahovat technické nepřesnosti nebo typografické chyby. Údaje zde uvedené jsou pravidelně upravovány a tyto změny budou zahrnuty v nových vydáních této publikace. Společnost IBM může kdykoli bez upozornění provádět vylepšení nebo změny v produktech či programech popsaných v této publikaci.

Veškeré uvedené odkazy na webové stránky, které nespravuje společnost IBM, jsou uváděny pouze pro referenci a v žádném případě neslouží jako záruka funkčnosti těchto webů. Materiály uvedené na tomto webu nejsou součástí materiálů pro tento produkt IBM a použití uvedených stránek je pouze na vlastní nebezpečí.

Společnost IBM může použít nebo distribuovat jakékoli informace, které jí sdělíte, libovolným způsobem, který společnost považuje za odpovídající, bez vyžádání vašeho svolení.

Vlastníci licence k tomuto programu, kteří chtějí získat informace o možnostech (i) výměny informací s nezávisle vytvořenými programy a jinými programy (včetně tohoto) a (ii) oboustranného využití vyměňovaných informací, mohou kontaktovat informační středisko na adrese:

IBM Corporation
Koordinátor spolupráce softwaru, oddělení 49XA
148 00 Praha 4-Chodby

148 00 Praha 4-Chodov
U.S.A.

Poskytnutí takových informací může být podmíněno dodržením určitých podmínek a požadavků zahrnujících v některých případech uhrazení stanoveného poplatku.

IBM poskytuje licencovaný program popsaný v těchto informacích a veškeré dostupné licencované materiály na základě podmínek smlouvy IBM Customer Agreement, IBM International Program License Agreement nebo jiné ekvivalentní smlouvy mezi námi.

Jakékoli údaje o výkonnosti obsažené v této publikaci byly zjištěny v řízeném prostředí. Výsledky získané v jakémkoli jiném operačním prostředí se proto mohou výrazně lišit. Některá měření mohla být prováděna na vývojových verzích systémů a není zaručeno, že tato měření budou stejná i na běžně dostupných systémech. Některá měření mohla být navíc odhadnuta pomocí extrapolace. Skutečné výsledky mohou být jiné. Čtenáři tohoto dokumentu by měli zjistit použitelné údaje pro své specifické prostředí.

Informace týkající se produktů jiných výrobců pocházejí od dodavatelů těchto produktů, z jejich veřejných oznámení nebo z jiných veřejně dostupných zdrojů. Společnost IBM tyto produkty netestovala a nemůže potvrdit správný výkon, kompatibilitu ani žádné jiné výroky týkající se produktů jiných výrobců než IBM. Otázky týkající se kompatibility produktů jiných výrobců by měly být směrovány dodavatelům těchto produktů.

Veškerá tvrzení týkající se budoucího směru vývoje nebo záměrů společnosti IBM se mohou bez upozornění změnit nebo mohou být zrušena a reprezentují pouze cíle a plány společnosti.

Tyto údaje obsahují příklady dat a sestav používaných v běžných obchodních operacích. Aby byla představa úplná, používají se v příkladech jména osob a názvy společností, značek a produktů. Všechna tato jména a názvy jsou fiktivní a jejich podobnost se jmény, názvy a adresami používanými ve skutečnosti je zcela náhodná.

LICENČNÍ INFORMACE:

Tyto informace obsahují ukázkové aplikační programy ve zdrojovém jazyce ilustrující programovací techniky na různých operačních platformách. Tyto ukázkové programy můžete bez závazků vůči společnosti IBM jakýmkoli způsobem kopírovat, měnit a distribuovat za účelem vývoje, používání, odbytu či distribuce aplikačních programů odpovídajících rozhraní API pro operační platformu, pro kterou byly ukázkové programy napsány. Tyto příklady nebyly plně testovány za všech podmínek. Společnost IBM proto nemůže zaručit spolehlivost, upotřebitelnost nebo funkčnost těchto programů.

Při prohlížení těchto dokumentů v elektronické podobě se nemusí zobrazit všechny fotografie a barevné ilustrace.

Informace o programovacím rozhraní

Informace programátorských rozhraní, je-li poskytnuta, vám pomohou vytvořit aplikační software pro použití s tímto programem.

Tato příručka obsahuje informace o zamýšlených programovacích rozhraních, které umožňují zákazníkům psát programy za účelem získání služeb produktu IBM WebSphere MQ.

Tyto informace však mohou obsahovat i diagnostické údaje a informace o úpravách a ladění. Informace o diagnostice, úpravách a vyladění jsou poskytovány jako podpora ladění softwarových aplikací.

Důležité: Nepoužívejte tyto informace o diagnostice, úpravách a ladění jako programátorské rozhraní, protože se mohou měnit.

Ochranné známky

IBM, logo IBM , ibm.comjsou ochranné známky společnosti IBM Corporation, registrované v mnoha jurisdikcích po celém světě. Aktuální seznam ochranných známek IBM je k dispozici na webu na stránce "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml. Ostatní názvy produktů a služeb mohou být ochrannými známkami společnosti IBM nebo jiných společností.

Microsoft a Windows jsou ochranné známky společnosti Microsoft Corporation ve Spojených státech a případně v dalších jiných zemích.

UNIX je registrovaná ochranná známka skupiny The Open Group ve Spojených státech a případně v dalších jiných zemích.

Linux je registrovaná ochranná známka Linuse Torvaldse ve Spojených státech a případně v dalších jiných zemích.

Tento produkt obsahuje software vyvinutý v rámci projektu Eclipse Project (http://www.eclipse.org/).

Java a všechny ochranné známky a loga založené na termínu Java jsou ochranné známky nebo registrované ochranné známky společnosti Oracle anebo příbuzných společností.

Číslo položky:

(1P) P/N: