7.5

Mobil İleti Sistemi ve M2M

Not

Bu bilgileri ve desteklediği ürünü kullanmadan önce, “Özel notlar” sayfa 187 bölümündeki bilgileri okuyun.

Bu basım, yeni basımlarında tersi belirtilmediği sürece, IBM® WebSphere MQ ' ın 7. yayın düzeyi 5 'i ve sonraki tüm yayın ve değişiklik düzeyleri için geçerlidir.

İçindekiler

MQTT ' a Giriş

MQ telemetri aktarımını kullanarak mobil uygulamalar arasında ileti gönderme hakkında bilgi edinin (MQTT). Protokol, kablosuz ve düşük bant genişlikli ağlarda kullanılmak üzere tasarlanmıştır. MQTT kullanan bir mobil uygulama, MQTT kitaplığını çağırarak ileti gönderir ve alır. İletiler bir MQTT ileti sistemi sunucusu aracılığıyla değiştirilir. MQTT istemcisi ve sunucusu, mobil uygulama için güvenilir bir şekilde ileti sağlama ve ağ yönetimi maliyetini düşük tutma karmaşıklıklarını ele alır.

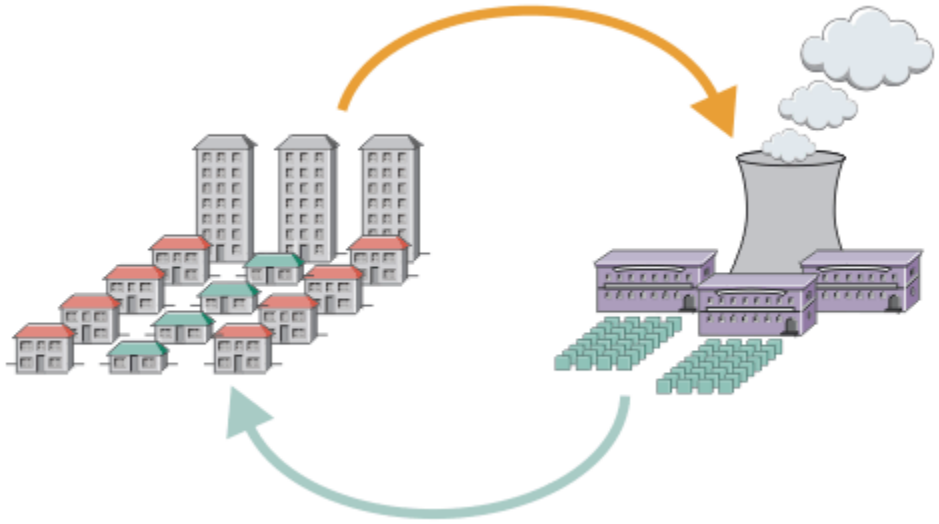
MQTT uygulamaları, akıllı telefonlar ve tabletler gibi mobil aygıtlarda çalışır. MQTT ayrıca, algılayıcılardan veri almak ve bunları uzaktan denetlemek için telemetri için de kullanılır. Mobil aygıtlar ve algılayıcılar için MQTT , güvenli bir teslimatla yüksek düzeyde ölçeklenebilir bir yayınlama/abone olma iletişim kuralı sunar. MQTT iletilerini göndermek ve almak için uygulamanıza bir MQTT istemci kitaplığı eklersiniz.

MQTT istemcisi kitaplığı küçüktür. Kitaplık, MQTT sunucusuna bağlı diğer MQTT uygulamalarıyla ileti gönderip alan bir posta kutusu gibi davranır. MQTT uygulamaları, yanıt bekleyen bir sunucuya bağlı kalmak yerine ileti göndererek pil ömrünü muhafaza eder. Kitaplık, MQTT version 3.1 iletişim kuralını çalıştıran bir MQTT sunucusu aracılığıyla diğer aygıtlara ileti gönderir. Belirli bir istemciye ileti gönderebilir ya da birçok aygıtı bağlamak için yayınlama/abone olma ileti sistemini kullanabilirsiniz.

MQTT istemci kitaplıkları, mobil aygıtlar ve algılayıcılar için uygulamaları MQTT iletişim kuralını kullanarak bir MQTT sunucusuna bağlar.

IBM MessageSight ve IBM WebSphere MQ , MQTT sunucularıdır. Büyük hacimli MQTT istemci uygulamalarını bağlayabilir ve MQTT ve IBM WebSphere MQ ağlarını birbirine bağlayabilirler. Bkz. “MQTT Server sunucuları ile çalışmaya başlama” sayfa 134. IBM WebSphere MQ ve IBM MessageSight , mobil aygıtlarda ve algılayıcılarda çalışan dış web uygulamaları ile kuruluş içinde çalışan diğer yayınlama/abone olma ve ileti sistemi uygulamaları arasında bir köprü oluşturabilir. Köprü, mobil aygıtları ve sensörleri içeren "akıllı çözümler" oluşturulmasını kolaylaştırır.

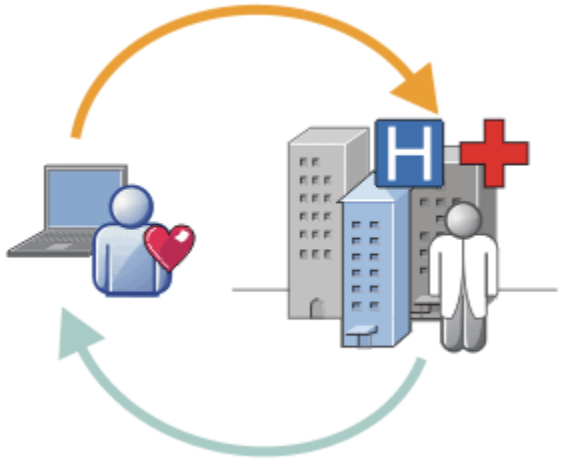
Akıllı çözümler, mobil ve sensör cihazlarında çalışan uygulamalara internet üzerinden sağlanan bilgi zenginliği elde eder. Telemetriye dayanan akıllı uygulamalara örnek olarak akıllı elektrik ve akıllı sağlık hizmetleri verilebilir.



- Hizmet sağlayıcıya gönderilen enerji kullanım verilerini içeren bir MQTT iletisi.
- Bir telemetri uygulaması, enerji kullanım verilerinin analizine dayalı olarak denetim komutları gönderir.
- Daha fazla bilgi için bkz. Telemetry scenario: Home energy monitoring and control.

Şekil 1. Akıllı elektrik ölçümü

Şekil 2. Akıllı durum izleme



- Bir telemetri uygulaması sağlık verilerinizi hastanenize ve doktorunuza gönderir.
- MQTT ileti uyarıları ya da geribildirim, sağlık verilerinizin analizine dayalı olarak gönderilebilir.
- Daha fazla bilgi için bkz. Telemetry scenario: Home patient monitoring.

MQTT protocoliçin kendi uygulamanızı yazarak küçük aygıtlara MQTT oluşturabilirsiniz. IBM , bunu yapmanıza yardımcı olmak için MQTTüzerinde çalışan uygulamaları destekleyen istemci kitaplıkları sağlar. Bkz. “MQTT istemcilerini kullanmaya başlama” sayfa 9. IBM , iOS uygulamaları ve Android uygulamalar V 7.5.0.1 için istemci kitaplıkları ve JavaScript platformdan bağımsız web uygulamaları için tarayıcı istemcisisağlar. V 7.5.0.1 JavaScript istemci sayfaları, WebSocketsüzerinden MQTT iletişim kuralıyla IBM MessageSight ve IBM WebSphere MQ ' e bağlanır. IBM , Linux® ve Windowsüzerinde MQTT C ve Java için örnek uygulamalar sağlar.

C ve Java kitaplıkları iOS, Android, Windowsve bazı UNIX and Linux platformlarında çalışır. MQTT istemci kitaplığına ilişkin C kaynak kodunu diğer platformlara da bağlayabilirsiniz. C ve Java için MQTT istemci kitaplıkları, Eclipse Paho projesinden bir açık kaynak lisansıyla kullanılabilir. Bkz. Eclipse Paho. MQTT protocol belirtimi açıktır ve MQTT.orgadresinden edinilebilir.

MQTT protocol

MQTT protocol , müşterilerin küçük olması ve ağ bant genişliğini verimli bir şekilde kullanması açısından hafiftir. MQTT iletişim kuralı, güvenli teslimat ve yangın ve unut aktarımlarını destekler. Protokolde, ileti teslimi uygulamadan ayrışmıştır. Bir uygulamadaki ayrışmanın kapsamı, MQTT istemcisi ve MQTT sunucusunun yazış şekline bağlıdır. Ayrılmış teslim, bir uygulamayı herhangi bir sunucu bağlantısından ve ileti beklemekten kurtarmıştır. Etkileşim modeli e-posta gibi, ancak uygulama programlama için optimize edilmiştir.

MQTT V3.1 iletişim kuralı yayınlanır; bkz. MQTT V3.1 Protokol Belirtimi. Belirtim, protokolle ilgili bir dizi ayırt edici özelliği tanımlar:

- Bu bir yayınlama/abone olma protokolüdür.

  Birden çoğa ileti dağıtımı sağlamanın yanı sıra, yayınlama/abone olma ayrışmaları uygulamaları da sağlar. Birçok istemcisi olan uygulamalarda her iki özellik de kullanışlıdır.
- Hiçbir şekilde ileti içeriğine bağımlı değildir.
- Temel ağ bağlanırlığı sağlayan TCP/IP üzerinden çalışır.
- İleti teslimatı için üç hizmet niteliğine sahiptir:

  "En çok bir kez"
  İletiler, temel Internet Protocol ağının en iyi çabalarına göre teslim edilir. İleti kaybı oluşabilir.

  Örneğin, iletişim ortamı algılayıcı verileriyle bu hizmet kalitesini kullanın. Bir sonraki kısa bir süre sonra yayımlanmışsa, bireysel bir okuma kaybolup kaybolmaması önemli değildir.

"En az bir kez"
İletilerin ulaşacağına emin olabilirsiniz, ancak yinelemeler oluşabilir.

"Tam olarak bir kez"
Mesajların tam olarak bir kez ulaşacağına emin olabilirsiniz.

Örneğin, faturalama sistemleriyle bu hizmet kalitesini kullanın. Yinelenen ya da kaybolan iletiler, rahatsızlığa ya da yanlış ücretlere neden olabilir.

- Ağ üzerindeki ileti akışını yönetme şeklinden ekonomiktir. Örneğin, sabit uzunluklu üstbilgi yalnızca 2 bayt uzunluğundadır ve ağ trafiğini azaltmak için protokol değiş tokuşları en aza indirilir.
- Bir istemcinin MQTT sunucusuyla bağlantısının olağandışı kesildiğini abonelere bildiren bir "Son Will and Testament" özelliği vardır. Bkz. “Son irade ve vasiyet yayını” sayfa 127.

MQTT version 3.1 , IBM IBM WebSphere MQ ve IBM MessageSighttarafından desteklenir. MQTT TCP/IP üzerinden uygulanır. Protokolün başka bir sürümü (MQTT-S), TCP/IP dışı ağlar için kullanılabilir. Bkz. MQTT-S version 1.2 belirtimi.

MQTT toplulukları

IBM , IBM MessageSight ve IBM WebSphere MQiçin uygulamalar yazan MQTT geliştiricileri için IBM Developer İleti sistemi Topluluk ' yi çalıştırıyor.

MQTT.org , MQTT iletişim kuralı uygulamaları ve uzantıları hakkında bilgi edinmek ve bunları tartışmak için iyi bir yerdir.

MQTT , Eclipse Teknoloji Projesialtındaki bir açık kaynak Eclipse projesidir. Paho topluluğu, açık kaynaklı istemciler ve sunucular geliştiriyor. Bkz. Eclipse Paho.

MQTT ' a Giriş

MQ telemetri aktarımını kullanarak mobil uygulamalar arasında ileti gönderme hakkında bilgi edinin (MQTT). Protokol, kablosuz ve düşük bant genişlikli ağlarda kullanılmak üzere tasarlanmıştır. MQTT kullanan bir mobil uygulama, MQTT kitaplığını çağırarak ileti gönderir ve alır. İletiler bir MQTT ileti sistemi sunucusu aracılığıyla değiştirilir. MQTT istemcisi ve sunucusu, mobil uygulama için güvenilir bir şekilde ileti sağlama ve ağ yönetimi maliyetini düşük tutma karmaşıklıklarını ele alır.

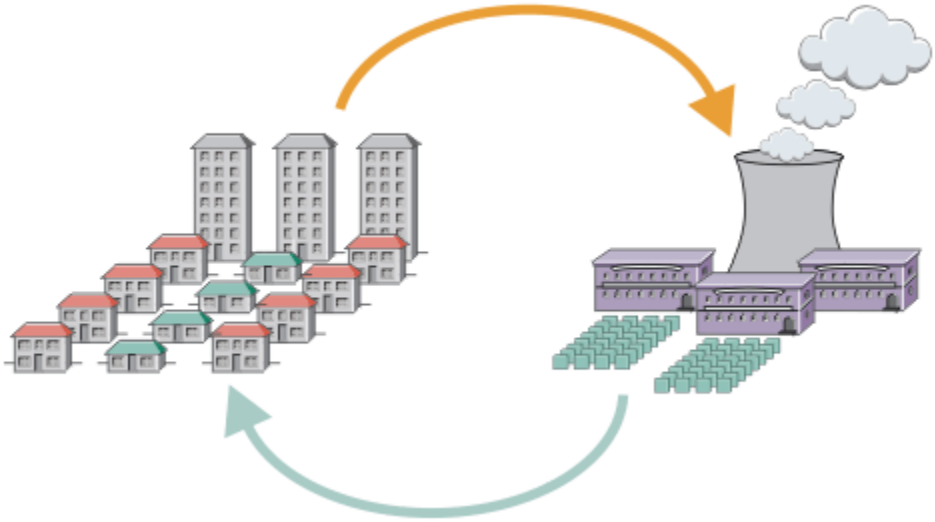
MQTT uygulamaları, akıllı telefonlar ve tabletler gibi mobil aygıtlarda çalışır. MQTT ayrıca, algılayıcılardan veri almak ve bunları uzaktan denetlemek için telemetri için de kullanılır. Mobil aygıtlar ve algılayıcılar için MQTT , güvenli bir teslimatla yüksek düzeyde ölçeklenebilir bir yayınlama/abone olma iletişim kuralı sunar. MQTT iletilerini göndermek ve almak için uygulamanıza bir MQTT istemci kitaplığı eklersiniz.

MQTT istemcisi kitaplığı küçüktür. Kitaplık, MQTT sunucusuna bağlı diğer MQTT uygulamalarıyla ileti gönderip alan bir posta kutusu gibi davranır. MQTT uygulamaları, yanıt bekleyen bir sunucuya bağlı kalmak yerine ileti göndererek pil ömrünü muhafaza eder. Kitaplık, MQTT version 3.1 iletişim kuralını çalıştıran bir MQTT sunucusu aracılığıyla diğer aygıtlara ileti gönderir. Belirli bir istemciye ileti gönderebilir ya da birçok aygıtı bağlamak için yayınlama/abone olma ileti sistemini kullanabilirsiniz.

MQTT istemci kitaplıkları, mobil aygıtlar ve algılayıcılar için uygulamaları MQTT iletişim kuralını kullanarak bir MQTT sunucusuna bağlar.

IBM MessageSight ve IBM WebSphere MQ , MQTT sunucularıdır. Büyük hacimli MQTT istemci uygulamalarını bağlayabilir ve MQTT ve IBM WebSphere MQ ağlarını birbirine bağlayabilirler. Bkz. “MQTT Server sunucuları ile çalışmaya başlama” sayfa 134. IBM WebSphere MQ ve IBM MessageSight , mobil aygıtlarda ve algılayıcılarda çalışan dış web uygulamaları ile kuruluş içinde çalışan diğer yayınlama/abone olma ve ileti sistemi uygulamaları arasında bir köprü oluşturabilir. Köprü, mobil aygıtları ve sensörleri içeren "akıllı çözümler" oluşturulmasını kolaylaştırır.

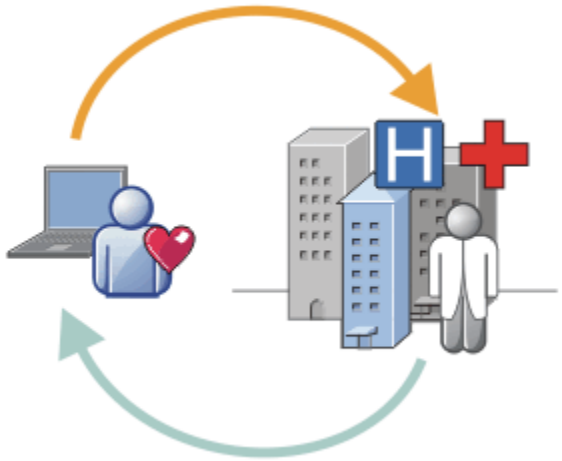
Akıllı çözümler, mobil ve sensör cihazlarında çalışan uygulamalara internet üzerinden sağlanan bilgi zenginliği elde eder. Telemetriye dayanan akıllı uygulamalara örnek olarak akıllı elektrik ve akıllı sağlık hizmetleri verilebilir.



- Hizmet sağlayıcıya gönderilen enerji kullanım verilerini içeren bir MQTT iletisi.
- Bir telemetri uygulaması, enerji kullanım verilerinin analizine dayalı olarak denetim komutları gönderir.
- Daha fazla bilgi için bkz. Telemetry scenario: Home energy monitoring and control.

Şekil 3. Akıllı elektrik ölçümü

Şekil 4. Akıllı durum izleme



- Bir telemetri uygulaması sağlık verilerinizi hastanenize ve doktorunuza gönderir.
- MQTT ileti uyarıları ya da geribildirim, sağlık verilerinizin analizine dayalı olarak gönderilebilir.
- Daha fazla bilgi için bkz. Telemetry scenario: Home patient monitoring.

MQTT protocoliçin kendi uygulamanızı yazarak küçük aygıtlara MQTT oluşturabilirsiniz. IBM , bunu yapmanıza yardımcı olmak için MQTTüzerinde çalışan uygulamaları destekleyen istemci kitaplıkları sağlar. Bkz. “MQTT istemcilerini kullanmaya başlama” sayfa 9. IBM , iOS uygulamaları ve Android uygulamalar V 7.5.0.1 için istemci kitaplıkları ve JavaScript platformdan bağımsız web uygulamaları için tarayıcı istemcisisağlar. V 7.5.0.1 JavaScript istemci sayfaları, WebSocketsüzerinden MQTT iletişim kuralıyla IBM MessageSight ve IBM WebSphere MQ ' e bağlanır. IBM , Linux ve Windowsüzerinde MQTT C ve Java için örnek uygulamalar sağlar.

C ve Java kitaplıkları iOS, Android, Windowsve bazı UNIX and Linux platformlarında çalışır. MQTT istemci kitaplığına ilişkin C kaynak kodunu diğer platformlara da bağlayabilirsiniz. C ve Java için MQTT istemci kitaplıkları, Eclipse Paho projesinden bir açık kaynak lisansıyla kullanılabilir. Bkz. Eclipse Paho. MQTT protocol belirtimi açıktır ve MQTT.orgadresinden edinilebilir.

MQTT protocol

MQTT protocol , müşterilerin küçük olması ve ağ bant genişliğini verimli bir şekilde kullanması açısından hafiftir. MQTT iletişim kuralı, güvenli teslimat ve yangın ve unut aktarımlarını destekler. Protokolde, ileti teslimi uygulamadan ayrışmıştır. Bir uygulamadaki ayrışmanın kapsamı, MQTT istemcisi ve MQTT sunucusunun yazış şekline bağlıdır. Ayrılmış teslim, bir uygulamayı herhangi bir sunucu bağlantısından ve ileti beklemekten kurtarmıştır. Etkileşim modeli e-posta gibi, ancak uygulama programlama için optimize edilmiştir.

MQTT V3.1 iletişim kuralı yayınlanır; bkz. MQTT V3.1 Protokol Belirtimi. Belirtim, protokolle ilgili bir dizi ayırt edici özelliği tanımlar:

- Bu bir yayınlama/abone olma protokolüdür.

  Birden çoğa ileti dağıtımı sağlamanın yanı sıra, yayınlama/abone olma ayrışmaları uygulamaları da sağlar. Birçok istemcisi olan uygulamalarda her iki özellik de kullanışlıdır.
- Hiçbir şekilde ileti içeriğine bağımlı değildir.
- Temel ağ bağlanırlığı sağlayan TCP/IP üzerinden çalışır.
- İleti teslimatı için üç hizmet niteliğine sahiptir:

  "En çok bir kez"
  İletiler, temel Internet Protocol ağının en iyi çabalarına göre teslim edilir. İleti kaybı oluşabilir.

  Örneğin, iletişim ortamı algılayıcı verileriyle bu hizmet kalitesini kullanın. Bir sonraki kısa bir süre sonra yayımlanmışsa, bireysel bir okuma kaybolup kaybolmaması önemli değildir.

  "En az bir kez"
  İletilerin ulaşacağına emin olabilirsiniz, ancak yinelemeler oluşabilir.

  "Tam olarak bir kez"
  Mesajların tam olarak bir kez ulaşacağına emin olabilirsiniz.

  Örneğin, faturalama sistemleriyle bu hizmet kalitesini kullanın. Yinelenen ya da kaybolan iletiler, rahatsızlığa ya da yanlış ücretlere neden olabilir.
- Ağ üzerindeki ileti akışını yönetme şeklinden ekonomiktir. Örneğin, sabit uzunluklu üstbilgi yalnızca 2 bayt uzunluğundadır ve ağ trafiğini azaltmak için protokol değiş tokuşları en aza indirilir.
- Bir istemcinin MQTT sunucusuyla bağlantısının olağandışı kesildiğini abonelere bildiren bir "Son Will and Testament" özelliği vardır. Bkz. “Son irade ve vasiyet yayını” sayfa 127.

MQTT version 3.1 , IBM IBM WebSphere MQ ve IBM MessageSighttarafından desteklenir. MQTT TCP/IP üzerinden uygulanır. Protokolün başka bir sürümü (MQTT-S), TCP/IP dışı ağlar için kullanılabilir. Bkz. MQTT-S version 1.2 belirtimi.

MQTT toplulukları

IBM , IBM MessageSight ve IBM WebSphere MQiçin uygulamalar yazan MQTT geliştiricileri için IBM Developer İleti sistemi Topluluk ' yi çalıştırıyor.

MQTT.org , MQTT iletişim kuralı uygulamaları ve uzantıları hakkında bilgi edinmek ve bunları tartışmak için iyi bir yerdir.

MQTT , Eclipse Teknoloji Projesialtındaki bir açık kaynak Eclipse projesidir. Paho topluluğu, açık kaynaklı istemciler ve sunucular geliştiriyor. Bkz. Eclipse Paho.

MQTT istemcilerini kullanmaya başlama

MQTT istemci kitaplığını kullanan bir örnek MQTT istemci uygulaması oluşturarak ve çalıştırarak bir mobil ya da makineden makineye (M2M) uygulaması geliştirmeye başlayabilirsiniz. Örnek uygulamalar ve ilişkili istemci kitaplıkları, IBMiçindeki Mobil İleti Sistemi ve M2M Client Pack içinde bulunur. Java, JavaScriptve C dillerinde yazılan uygulamaların ve istemci kitaplıklarının sürümleri vardır. Bu uygulamaları, Apple' den Android aygıtları ve ürünleri de dahil olmak üzere birçok platformda ve aygıtta çalıştırabilirsiniz.

Başlamadan önce

Uygulamanızı oluşturmak ve çalıştırmak için, hedef aygıt veya platform ve kullanılan programlama dili için uygulamalar oluşturma konusunda biraz deneyime gereksinim duyarsınız. Küçük bir deneyim genellikle örnek bir uygulamayı seçtiğiniz aygıt ya da platformda çalışır durumda yapmak için yeterlidir.

IBM WebSphere MQ ya da IBM MessageSightgibi kurumsal düzeyde güçlü bir MQTT sunucusu kullanıyorsanız, örnek uygulamanızdan var olan kurumsal uygulamalarınızla bilgi alışverişi yapabilirsiniz.

Bu görev hakkında

Hedefleriniz şunlardır:

1. İstemci uygulamasını bağlayabileceğiniz bir MQTT sunucusu seçin.
2. Mobil İleti Sistemi ve M2M Client Pack' ı karşıdan yükleyin.
3. Hedef aygıtınız ya da platformunuz için istemci paketinden örnek uygulamalar oluşturun.
4. Örnekleri MQTT sunucusuna bağlayarak, örneklerin beklendiği gibi davrandığını doğrulayın.

Aygıtınıza ya da platformunuza ilişkin örnek uygulamaları oluşturmanın ve test etmenin bir sonucu olarak, daha sonra kendi istemci uygulamalarınızı oluşturmak için kullanabileceğiniz bir çalışma geliştirme ortamı yaratırsınız.

Mobil İleti Sistemi ve M2M Client Pack , MQTT SDK ' yı içerir. Bu SDK size aşağıdaki kaynakları sağlar:

- Örnek MQTT istemci uygulamaları Java, JavaScriptve C içinde yazılmıştır.
- Bu istemci uygulamalarını destekleyen ve bunların çoğu platformda ve aygıtta çalıştırılmasını sağlayan MQTT istemci kitaplıkları.

SDK, C için MQTT istemcisiiçin kaynak kodu da içerir. Bu kaynak kodu, diğer platformlar için C için MQTT istemci kitaplıkları oluşturmak üzere uyarlayabilirsiniz. Bunun için yardım almak üzere bkz. “C kitaplıkları için MQTT istemcisi oluşturulması” sayfa 29. C için MQTT istemcisi kaynak kodu, Eclipse Pahotarafından sağlanan bir açık kaynak lisansıyla da kullanılabilir.

Yordam

Aşağıdaki makaleler, masaüstü bilgisayarda ya da Android ya da Appleiçin mobil bir aygıtta örnek bir MQTT uygulaması oluşturmak ve çalıştırmak için platforma özgü adımlar boyunca size yol gösterir:

- “Javaiçin MQTT istemcisiyle çalışmaya başlama” sayfa 11
- “Android üzerinde Javaiçin MQTT istemcisi ile çalışmaya başlama” sayfa 17
- V 7.5.0.1 “MQTT messaging client for JavaScriptile çalışmaya başlama” sayfa 23
- “C için MQTT istemcisiyle çalışmaya başlama” sayfa 25
- “iOSüzerinde C için MQTT istemcisi ile çalışmaya başlama” sayfa 45

Sonraki adım

Yeni bir MQTT uygulaması geliştirmek için aşağıdaki becerilere sahip olmanız ya da edinmeniz gerekir:

- Aygıt ya da platform için gerekli olan dilde programlama.
- Hedef aygıt ya da platform için programlama.
- Yayınlama/abone olma uygulamalarının tasarlanması.
- MQTT programlama modeli için programlar tasarlama.
- Seçtiğiniz mobil aygıtta çalıştırılacak programlar tasarlama.
- Programların güvenliğini sağlamak için SSL ve JAAS kullanılması.

MQTT bir ileti sistemi ve kuyruğa alma sistemi olduğundan, MQTT istemcisini başka bir aygıta ya da uygulamaya bağlamak için herhangi bir ağ programlama becerisine gerek yoktur. MQTT istemci kitaplıkları, uygulamanıza ilişkin ağ bağlantılarını yönetir.

MQTT istemcinizi var olan kurumsal uygulamalarla bütünleştirmek için iki seçeneğiniz vardır. MQTT yayınlama/abone olma konularını bir IBM WebSphere MQ ya da JMS uygulamasıyla paylaşabilir ya da kendi bütünleştirme bağdaştırıcınızı başka bir MQTT istemcisi olarak yazabilirsiniz.

Bugün görüşülmesi gereken bilgi kaynakları şunlardır:

- WebSphere MQ Telemetryiçin uygulama geliştirilmesi
- MQTT.org
- Eclipse Paho

İlgili kavramlar

“MQTT Server sunucuları ile çalışmaya başlama” sayfa 134

Javaiçin MQTT istemcisiyle çalışmaya başlama

' den bir istemci kitaplığı kullanır ve either IBM MessageSight or IBM WebSphere MQ as the MQTT serveristemcisini kullanarak Java örnek uygulamaları için MQTT Client istemcisiyle çalışır. Örnek uygulamalar, MQTT yazılım geliştirme araç takımından (SDK) IBM' den istemci kitaplığı kullanır. `SampleAsyncCallBack` örnek uygulaması, Android uygulamaları ve olay odaklı diğer işletim sistemleri için MQTT uygulamaları yazmak için kullanılan bir modeldir.

Başlamadan önce

- You can run an Javaiçin MQTT istemcisi app on any platform with JSE 1.5 or above that is "Java Uyumlu". Bkz. IBM Mobil İleti Sistemi ve M2M Client Packiçin sistem gereksinimleri.
- İstemciniz ve sunucu arasında bir güvenlik duvarı varsa, bunun MQTT trafiğini engellediğinden emin olun.

Bu görev hakkında

The purpose of the task is to check that you can build and run an MQTT client for Java sample application, connect it to IBM WebSphere MQ or IBM MessageSight as the MQTT version 3 server, and exchange messages.

Örnek uygulamayı Eclipse Workbench 'ten ya da bir komut satırından çalıştırmak için bu görevi izleyin. Örnekteki adımlar Windowsiçindir. Küçük değişikliklerle, örnek uygulamayı JSE 1.5 ya da üstünü destekleyen herhangi bir platformda çalıştırabilirsiniz.

You can run applications on the same server as IBM WebSphere MQ, where the environment for running applications that connect to IBM WebSphere MQ is configured for you. Follow the task “Configuring the MQTT service from the command line” sayfa 138 to install and configure IBM WebSphere MQ with the IBM WebSphere MQ Telemetry option on Windows or Linux. When the environment is installed and configured, run the sample application, `MQTTV3Sample`, to verify the installation.

Yordam

1. İstemci uygulamasını bağlayabileceğiniz bir MQTT sunucusu seçin.

   Sunucu, MQTT version 3.1 iletişim kuralını desteklemelidir. IBM WebSphere MQ ve IBM MessageSightde içinde olmak üzere, tüm MQTT sunucuları IBM ' dan bunu yapar. Bkz. “MQTT Server sunucuları ile çalışmaya başlama” sayfa 134.
2. İsteğe bağlı: MQTT sunucusunu yapılandırın.
   - On IBM WebSphere MQ, you must complete one or other the following tasks to set up a queue manager and configure its telemetry (MQXR) service:
     - “Configuring the MQTT service from the command line” sayfa 138

– “Configuring the MQTT service with IBM WebSphere MQ Explorer” sayfa 140

- Diğer sunucularda sunucu belgelerine bakın. Really Small Message Brokeriçin herhangi bir yapılandırma adımı gerekli değildir. Bkz. Really Small Message Broker.

3. Mobil İleti Sistemi ve M2M Client Pack dosyasını yükleyin ve MQTT SDK ' yı kurun.

   Kuruluş programı yok, yalnızca karşıdan yüklenen dosyayı genişletiyorsunuz.

   a. Mobil İleti Sistemi ve M2M Client Pack' ı karşıdan yükleyin.

   b. SDK ' yı kurabildiğiniz bir klasör oluşturun.

      `MQTT`klasörünün adını vermek isteyebilirsiniz. Bu klasöre giden yol, burada *`sdkroot`*olarak adlandırılır.

   c. Sıkıştırılmış Mobil İleti Sistemi ve M2M Client Pack dosyasının içeriğini *`sdkroot`*' e genişletin. Genişletme, *`sdkroot`*\SDK' ta başlayan bir dizin ağacı oluşturur.

4. Bir Java geliştirme seti (JDK) Sürüm 6 ya da sonraki bir sürümü kurun.

   Because you are developing a Java app for Android, the JDK must come from Oracle. JDK 'i Java SE Yüklemeleri' den alabilirsiniz.

5. Java örnek uygulamaları için bir ya da daha fazla MQTT istemcisini derleyin ve çalıştırın:

   - “Compile and run the Paho sample programs from the command line” sayfa 13
   - “Compile and run all the MQTT client sample Java apps from Eclipse” sayfa 14
   - “Android üzerinde Javaiçin MQTT istemcisi ile çalışmaya başlama” sayfa 17

   Aşağıdaki MQTT istemcisi örnek Java uygulamaları SDK ' da yer almaktadır:

   `MQTTV3Sample`
   Örnek, IBM WebSphere MQ ile de birlikte gönderilir ve `com.ibm.micro.client.mqttv3.jar` paketine bağlanır.

   `Sample`
   `Sample` , Paho paketinde ve `org.eclipse.paho.client.mqttv3` paketindeki bağlantılarda yer alıyor. `MQTTV3Sample`' a benzer; her MQTT işlemi tamamlanıncaya kadar bekler.

   `SampleAsyncWait`
   `SampleAsyncWait` , `org.eclipse.paho.client.mqttv3` paketinedir. Bu, zamanuyumsuz MQTT API ' yı kullanır; bir işlem tamamlanıncaya kadar farklı bir iş parçacığıda bekler. Ana iş parçacığı, MQTT işleminin tamamlanması için bekleyen iş parçacığıda uyumlulaştırılıncaya kadar başka işler de yapabilir.

   `SampleAsyncCallBack`
   `SampleAsyncCallBack` , `org.eclipse.paho.client.mqttv3` paketinedir. Bu, zamanuyumsuz MQTT API ' yı çağırır. Zamanuyumsuz API, MQTT ' un bir çağrıyı işlemeyi tamamlamayı beklemez; bu, uygulamaya geri döner. Uygulama diğer görevlerle birlikte devam eder, daha sonra işlenmek üzere bir sonraki olayın gelmesini bekler. MQTT , bir olay bildiriminin işlenmeyi tamamlayınca uygulamaya geri gönderilmesini sağlar. Olay odaklı MQTT arabirimi, Android ' in hizmet ve etkinlik programlama modeline ve diğer olay odaklı işletim sistemlerine uygundur.

   Örnek olarak, `mqttExerciser` örneğinin hizmet ve etkinlik programlama modeli kullanılarak MQTT 'i Android ' a nasıl bütünleştirdiği hakkında bir bakın.

   `mqttExerciser`
   `mqttExerciser` , Androidiçin örnek bir programdır. Farklı bir şekilde oluşturulduğundan ve çalıştırıldığından, ayrı olarak açıklanır. Bkz. “Android üzerinde Javaiçin MQTT istemcisi ile çalışmaya başlama” sayfa 17.

Sonuçlar

You compiled and ran the MQTT Java sample applications that are connected to IBM WebSphere MQ or IBM MessageSight as the MQTT server.

Sonraki adım

Javadoc başvuru bilgilerini araştır; bkz.step/ “Compile and run all the MQTT client sample Java apps from Eclipse” sayfa 14. Adım “3” sayfa 15 . Diğer bir seçenek olarak, Mobil İleti Sistemi ve M2M Client Packdizinindeki SDK\clients\java\doc\javadoc dizininde bulunan Javadoc html dosyalarını açabilirsiniz.

Compile and run the Paho sample programs from the command line

Compile and run the Paho sample application Sample.java from the command line. Örnek MQTT SDK ' de yer alıyor. Örnek, org.eclipse.paho.client.mqttv3 paketindeki MQTT Paho istemci kitaplıklarıyla oluşturulmuştur. Bir MQTT yayınlayıcısını ve aboneyi gösterir. Aynı dizinde yer alan diğer iki Paho örneği de aynı şekilde oluşturulabilir ve çalıştırılabilir. They differ by calling the MQTT library asynchronously.

Başlamadan önce

Bir MQTT sunucusunu yapılandırmak ve Mobil İleti Sistemi ve M2M Client Packdosyasını karşıdan yüklemek için ana görevdeki “1” sayfa 11 - “4” sayfa 12 adımlarını gerçekleştirin.

Bu görev hakkında

Compile and run Sample.java from the SDK client samples subdirectory SDK\clients\java\samples. The Java code is in the directory, SDK\clients\java\samples\org\eclipse\paho\sample\mqttv3app.

Yordam

1. Seçilen altyapınızda Sample derlemek ve çalıştırmak için istemci örnekleri dizininde bir komut kütüğü yaratın.

   The following script compiles and runs the sample on 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
   ```

   Şekil 5. Compile and run Sample.java
2. Komut dosyasını çalıştırın.

   Sonuçlar:

   ```
   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
   ```

   Şekil 6. MQTTV3Sample Abone

```
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
```

Şekil 7. MQTTV3Sample Yayıncı

Abone uygulamasını çalışır durumda bırakırsanız, aynı aboneyi yeniden çalıştıramazsınız. Yeni abonenin istemci tanıtıcısı, eski aboneye benzer. Aynı istemci tanıtıcısına sahip iki MQTT istemcisini aynı anda çalıştıramazsınız. Farklı aboneleri aynı anda çalıştırabilmeniz için, istemci tanıtıcısını ayarlamak için `-i` seçeneğini ayarlayabilirsiniz.

Aynı istemciyi yeniden çalıştırırsanız, istemciyi `temizoturum` ile başlatmak ve false olarak ayarlamak için `-c` seçeneğinden yararlanabilirsiniz. Bu seçenekle, kesintiye uğrayan istemci oturumlarının davranışını keşfedebilirsiniz.

3. Enter tuşuna basarak ya da pencereyi kapatarak aboneyi sona erdirin.

Sonraki adım

`SDK\clients\java\samples\org\eclipse\paho\sample\mqttv3app` alt dizininde diğer örnekleri derlemek ve çalıştırmak için komut dosyaları oluşturun. Komut dosyasını Şekil 5 sayfa 13 ' ta kopyalayın ve `Sample` dosyasını `SampleAsyncWait` ya da `SampleAsyncCallBack`ile değiştirin. Diğer örnekler, zamanuyumlu `Sample` programının zamanuyumsuz sürümleridir.

`SampleAsyncWait`
`SampleAsyncWait` , `org.eclipse.paho.client.mqttv3` paketinedir. Bu, zamanuyumsuz MQTT API ' yı kullanır; bir işlem tamamlanıncaya kadar farklı bir iş parçacığıda bekler. Ana iş parçacığı, MQTT işleminin tamamlanması için bekleyen iş parçacığıda uyumlulaştırılıncaya kadar başka işler de yapabilir.

`SampleAsyncCallBack`
`SampleAsyncCallBack` , `org.eclipse.paho.client.mqttv3` paketinedir. Bu, zamanuyumsuz MQTT API ' yı çağırır. Zamanuyumsuz API, MQTT ' un bir çağrıyı işlemeyi tamamlamayı beklemez; bu, uygulamaya geri döner. Uygulama diğer görevlerle birlikte devam eder, daha sonra işlenmek üzere bir sonraki olayın gelmesini bekler. MQTT , bir olay bildiriminin işlenmeyi tamamlayınca uygulamaya geri gönderilmesini sağlar. Olay odaklı MQTT arabirimi, Android ' in hizmet ve etkinlik programlama modeline ve diğer olay odaklı işletim sistemlerine uygundur.

Örnek olarak, `mqttExerciser` örneğinin hizmet ve etkinlik programlama modeli kullanılarak MQTT 'i Android ' a nasıl bütünleştirdiği hakkında bir bakın.

Zamanuyumsuz örnekler, MQTT istemcisi için beklerken MQTT uygulama bloklarının nasıl azaltılacağını gösterir. Bir mobil ortamda yanıt verisini ve pil ömrünü artırmak için ana iş parçacığıdaki engelleyici çağrıları ortadan kaldırmak önemlidir.

Örnekler, MQTT istemcisinde zamanuyumsuz arabirimleri çağırmak için iki kalıp gösterir.

1. MQTT , ağ etkileşimleri beklerken`SampleAsyncWait` engellenmez.
2. `SampleAsyncCallBack` , MQTT istemcisinin herhangi bir işlemi tamamlamayı beklemesini engellemez. Bir JavaScript sayfası, bir tarayıcıdan MQTT istemcisini çağırdığında, ikincinin gerekli olduğunu kabul eder. JavaScript sayfaları engellenmemelidir. İşlemlere verilen yanıtlar ana tarayıcı iş parçacığa geri gönderilmelidir; bu durumda, bildirimi işlemek için yazdığınız MQTT olay işleyicisini çağırır.

Compile and run all the MQTT client sample Java apps from Eclipse

Mobil İleti Sistemi ve M2M Client Pack' de bulunan MQTT istemcisi örnek Java uygulamalarını derleyin ve çalıştırın. Bir MQTT yayıncısı ve abonesi gösterirler.

Bu görev hakkında

Compile and run the MQTT Java samples, `Sample`, and `MQTTV3Sample` in Eclipse. `Sample` , *sdkroot*`\SDK\clients\java\samples\org\eclipse\paho\sample\mqttv3app` SDK istemcileri alt dizininde ve `MQTTV3Sample.java` , *sdkroot*`\SDK\clients\java\samples`dizininde yer alıyor.

Yordam

1. Eclipse IDE for Java Developersdosyasını karşıdan yükleyin.
2. Eclipseiçinde `MQTT Samples` olarak adlandırılan bir Java projesi oluşturun.
   a) **Dosya** > **Yeni** > **Java projesi** ve `MQTT Samples`yazın. **İleri**'yi tıklatın.

   JRE ' nin doğru ya da sonraki sürümlerde olup olmadığını denetleyin.JSE , sürüm 1.5 ya da sonraki bir sürümde olmalıdır.

   b) **Java Ayarları** penceresinde **Ek kaynak klasörleri bağla**seçeneğini tıklatın.

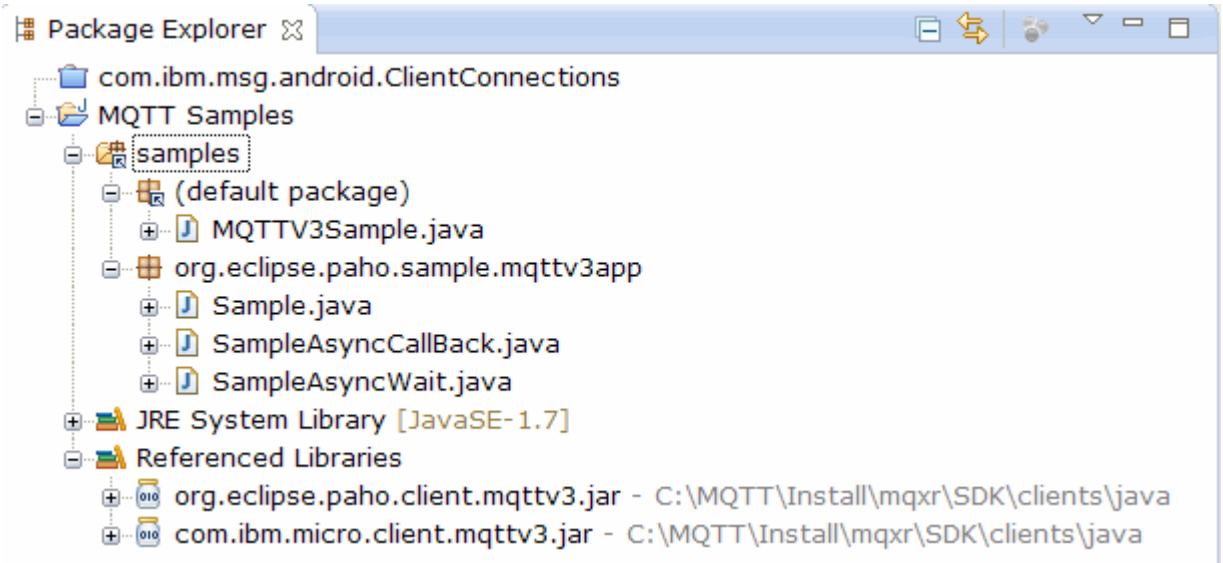
   c) Browse to the directory where you installed the MQTT Java SDK folder to. *sdkroot*`\SDK\clients\java\samples` klasörünü seçin ve **Tamam** > **İleri** > **Son**seçeneklerini tıklatın.

   d) **Java Ayarları** penceresinde **Kitaplıklar** > **Dış Jar Ekle** öğelerini tıklatın.

   e) Browse to the directory where you installed the MQTT Java SDK folder to. *sdkroot*`\SDK\clients\java` klasörünü bulun ve `org.eclipse.paho.client.mqttv3.jar` ve `com.ibm.micro.client.mqttv3.jar` dosyalarını seçin; **Aç** > **Son**seçeneklerini tıklatın.

   The `MQTTV3Sample.java` sample links to `com.ibm.micro.client.mqttv3.jar`, and the samples in the paho directory tree link to `org.eclipse.paho.client.mqttv3.jar`. The `com.ibm.micro.client.mqttv3.jar` is retained so that existing MQTT applications continue to build and run without change.

   `MQTT Samples` projesi bazı uyarılarla oluşturulur, ancak hata yok.



Şekil 8. Java istemcisi içinMQTT istemcisi

3. İsteğe bağlı: MQTT istemcisi Javadoc' u kurun.

   MQTT istemcisi Javadoc kurulu olduğu için, Java düzenleyicisi beliren yardımda MQTT sınıflarını tanımlar.

   a) Java projenizdeki **Paket Gezgini** > **Gönderme Yapılan Kitaplıklar** ' ı açın. `org.eclipse.paho.client.mqttv3.jar` **> Özellikler**seçeneğini sağ tıklatın.

   b) Özellikler gezgininde **Javadoc Yeri**seçeneğini tıklatın.

   c) **Javadoc Yeri** sayfasında **Javadoc URL 'si** > **Göz At** ' ı tıklatın ve `SDK\clients\java\doc\javadoc` klasörünü bulun **> Tamam**.

d) **Doğrula** > **Tamam**düğmesini tıklatın.

Belgeleri görüntülemek için bir tarayıcı açmanız istenir.

e) `com.ibm.micro.client.mqttv3.jar` dosyası için bu yordamı yineleyin.

4. `mqttv3app.Sample` uygulamasını çalıştırmak için bir yayınlayıcı ve abone yürütme ortamı yapılanışı yaratın.

a) **Örnek** sınıfını sağ tıklatın, **Olarak çalıştır** > **Yapılandırmaları çalıştır**seçeneğini tıklatın.

b) **Java Uygulaması** > **Yeni** seçeneğini sağ tıklatın ve `SampleSubscriber`adını yazın.

c) Bağımsız değişkenler sekmesini tıklatın ve program bağımsız değişkenlerini **Uygula**tarafından takip edilen bağımsız değişkenlere yazın.

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

d) `-a subscribe` değiştirgesini atlayarak bir `SamplePublisher` yapılanışı yaratmak için son adımı yineleyin.

5. Run the `mqttv3app.Sample` subscriber followed by the publisher.

a) **Çalıştır** > **Çalıştırma yapılandırmaları** seçeneklerini tıklatın.

b) **SampleSubscriber** > **Run**öğelerini tıklatın.

Konsol görünümünü açın. Abone bir yayın bekliyor.

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

c) **SamplePublisher** > **Çalıştır**öğelerini tıklatın.

Konsol görünümünü açın. Yayınlayıcı tarafından oluşturulan yayını görürsünüz:

```
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) Konsol görünümlerini abone konsoluna geçin.

Anahtar konsolları simgesi  ' dir.

Aboneyi şu yayını aldı:

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

6. İsteğe bağlı: `MQTTV3Sample`için bir yayınlayıcı ve abone yürütme ortamı yapılanışı yaratın.

a) **MQTTV3Sample** sınıfını farenin sağ düğmesiyle tıklatın, **Bu şekilde çalıştır** > **Yapılandırmaları çalıştır**öğelerini seçin.

b) **Java Uygulaması** > **Yeni** seçeneğini sağ tıklatın ve `MQTTV3SampleSubscriber`adını yazın.

c) Bağımsız değişkenler sekmesini tıklatın ve program bağımsız değişkenlerini **Uygula**tarafından takip edilen bağımsız değişkenlere yazın.

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

d) `-a subscribe` değiştirgesini atlayarak bir `MQTTV3SamplePublisher` yapılanışı yaratmak için son adımı yineleyin.

7. İsteğe bağlı: Run the `MQTTV3Sample` subscriber followed by the publisher.

a) **Çalıştır** > **Çalıştırma yapılandırmaları** seçeneklerini tıklatın.

b) **MQTTV3SampleSubscriber** > **Çalıştır**öğelerini tıklatın.

Konsol görünümünü açın. Abone bir yayın bekliyor.

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

c) **MQTTV3SamplePublisher** > **Çalıştır**öğelerini tıklatın.

Konsol görünümünü açın. Yayınlayıcı tarafından oluşturulan yayını görebilirsiniz.

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

d) Konsol görünümlerini abone konsoluna geçin.

Anahtar konsolları simgesi  ' dir.

Aboneyi şu yayını aldı:

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

Android üzerinde Javaiçin MQTT istemcisi ile çalışmaya başlama

MQTT sunucusuyla ileti alışverişi yapan bir Androidiçin MQTT istemcisi örnek Java uygulaması kurabilirsiniz. Uygulama, IBM' in MQTT SDK 'sından bir istemci kitaplığı kullanır. Uygulamayı kendiniz oluşturabilir ya da önceden oluşturulmuş bir örnek uygulamayı karşıdan yükleyebilirsiniz.

Başlamadan önce

- Desteklenen ve MQTT istemci altyapılarına ilişkin bilgi için bkz. IBM Mobil İleti Sistemi ve M2M Client Packiçin sistem gereksinimleri.
- İstemciniz ve sunucu arasında bir güvenlik duvarı varsa, bunun MQTT trafiğini engellediğinden emin olun.
- MQTT istemcisi örnek uygulaması Ice Cream Sandwich (Android 4.0) ve üstü üzerinde çalışır. Bu Android sürümü, tabletlerde daha net bir görüntü çözünürlüğü de sağlar.

Bu görev hakkında

Androidiçin MQTT istemcisi örnek Java uygulaması , "`mqttExerciser`"olarak adlandırılır. Bu uygulama, MQTT SDK ' dan bir istemci kitaplığı kullanır ve iletileri bir MQTT sunucusuyla değiştirir.

Örnek uygulamayı kendiniz oluşturabilir ve daha sonra Eclipse ' dan `mqttExerciser.apk`olarak dışa aktarabilir ya da Mobil İleti Sistemi ve M2M Client Packdosyasının *`sdkroot`*`\SDK\clients\android\samples\apks` klasöründe `mqttExerciser.apk` dosyası olarak kullanılabilen önceden oluşturulmuş örnek uygulamayı kullanabilirsiniz. Uygulamayı kendiniz oluşturmayı seçerseniz, oluşturacağınız geliştirme ortamı, mobil ileti sistemini uygulamalara dahil edecek şekilde uyarlanmıştır for Android. Bu, kendi uygulamalarınıza mobil ileti sistemi eklemeye başladığınızda size yardımcı olacaktır.

Yordam

1. İstemci uygulamasını bağlayabileceğiniz bir MQTT sunucusu seçin.

   Sunucu, MQTT version 3.1 iletişim kuralını desteklemelidir. IBM WebSphere MQ ve IBM MessageSightde içinde olmak üzere, tüm MQTT sunucuları IBM ' dan bunu yapar. Bkz. “MQTT Server sunucuları ile çalışmaya başlama” sayfa 134.
2. Doğru araçları alın.

Bir Java geliştirme seti (JDK) Sürüm 6 ya da sonraki bir sürümü kurun. Because you are developing a Java app for Android, the JDK must come from Oracle. JDK 'i Java SE Yüklemeleri' den alabilirsiniz.

Bir Eclipse geliştirme ortamına da gereksinim duyarsınız. Bu, Eclipse 3.6.2 (Helios) ya da daha büyük bir değer olmalıdır. Eclipse , JDK ile eşleşmesi için en az 6 Java derleyicisi düzeyine sahip olmalıdır. Tüm bunları Eclipse Foundation' dan alabilirsiniz.

Son olarak, Android SDK gerekir. Bunu Android SDKolanağından alabilirsiniz.

3. Mobil İleti Sistemi ve M2M Client Pack dosyasını yükleyin ve MQTT SDK ' yı kurun.

   Kuruluş programı yok, yalnızca karşıdan yüklenen dosyayı genişletiyorsunuz.

   a. Mobil İleti Sistemi ve M2M Client Pack' ı karşıdan yükleyin.

   b. SDK ' yı kurabildiğiniz bir klasör oluşturun.

      `MQTT`klasörünün adını vermek isteyebilirsiniz. Bu klasöre giden yol, burada *`sdkroot`*olarak adlandırılır.

   c. Sıkıştırılmış Mobil İleti Sistemi ve M2M Client Pack dosyasının içeriğini *`sdkroot`*' e genişletin. Genişletme, *`sdkroot`*\SDK' ta başlayan bir dizin ağacı oluşturur.

4. İsteğe bağlı: mqttExerciser örnek uygulamasını for Androidoluşturun.

   Eclipse ve Android araçlarını yapılandırın ve MQTT SDK ' dan `mqttExerciser` projesini içe aktarın ve oluşturun.

   Not: Bunu şu anda yapmak istemiyorsanız, MQTT SDK ' nın *`sdkroot`*`\SDK\clients\android\samples\apks` klasöründe `mqttExerciser.apk` dosyası olarak kullanılabilen önceden oluşturulmuş örnek uygulamayı kullanabilirsiniz.

   a) Eclipse geliştirme ortamını JDK ' den JRE ile başlatın.

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

   b) Android SDK ' dan bir paket ve platform kümesi seçin ve kurun.

      Google tarafından önerilen platformların ve paketlerin listesi için Platformların ve Paketlerin Eklenmesi başlıklı konuya bakın.

      Not: SDK platformu Android API düzeyi 16 ya da üstü olmalıdır. Daha önceki API düzeyleriyle proje başarıyla derlenemiyor.

   c) Android Development Tools (ADT) eklentisini Eclipseolanağına ekleyin.

   d) Örnek `mqttExerciser` uygulama projesini Eclipseiçine aktarın ve hataları düzeltin.

      i) *`sdkroot`*`\SDK\clients\android\samples\mqttExerciser`yolundaki MQTT SDK ' dan örnek uygulama projesini içe aktarın.

         Sorunlar görünümünde birçok oluşturma hatası listelenir. Sonraki birkaç adımda oluşturma hatalarını çözün.

      ii) `org.eclipse.paho.client.mqttv3.jar` kitaplığını Android projesindeki **libs** klasörüne kopyalayın. Windows Örneğin, Windowsüzerinde bu, *`sdkroot`*`\SDK\clients\java` klasörünün altındadır. Bir **Dosya İşlemi** penceresi görüntülenir. **Dosyaları kopyala** seçimini kabul edin ve **Tamam**düğmesini tıklatın.

      iii) Proje klasörünü sağ tıklatın, `com.ibm.msg.android`; tıklatın **Android araçları ...** > **Destek Kitaplığı Ekle ...**. Lisans koşullarını okuyup kabul edin ve **Kur**düğmesini tıklatın.

      iv) Proje klasörünü sağ tıklatın, `com.ibm.msg.android`; tıklatın **Android araçları ...** > **Düzeltme Projesi Özellikleri**.

      v) Çalışma alanında bir üst sınıf yönteminin geçersiz kılınmasına atıfta bulunan yaklaşık 84 hata varsa, derleyici uyumluluk düzeyi büyük olasılıkla 1.5 ya da daha düşük bir değere ayarlanır. Android SDK sürüm 16, derleyici uyumluluk düzeyinin 1.5' ten büyük olmamasını bekler. Geri kalan hataları düzeltmek için aşağıdaki adımları tamamlayın:

a) Android SDK ve ilgili Eclipse eklentilerini Android SDK sürüm 17 'ye denetleyin ve (gerekirse) güncelleyin.

b) **com.ibm.msg.android** proje klasörünü sağ tıklatın ve **Özellikler** > **Java Derleyicisi**seçeneğini belirleyin. Derleyici uyumluluk düzeyini denetleyin, en az 1.6olarak ayarlayın ve çalışma alanını yeniden oluşturun.

Proje, bazı uyarılarla oluşturulmuştur ve hata oluşmaz.

5. MQTT istemcisi örnek Java uygulaması aygıtını Android aygıtına kurun ve başlatın.

   `developer.android.com` sayfasına bakın Uygulamanızı çalıştırma.

   Uygulamayı kendiniz bir Eclipse projesi olarak oluşturdıysanız, uygulamayı Eclipseadresinden başlatabilirsiniz.

   Uygulama paketi (APK) dosyası `mqttExerciser.apk`varsa, Android Debug Bridge (ADB) kuruluş komutunu kullanarak paketi Eclipse dışında kurabilirsiniz. Bu komut, APK dosyasının konumunu bağımsız değişken olarak alır. Önceden oluşturulmuş örnek uygulamayı kullanıyorsanız, konum *`sdkroot`*`\SDK\clients\android\samples\apks\mqttExerciser.apk`olur.

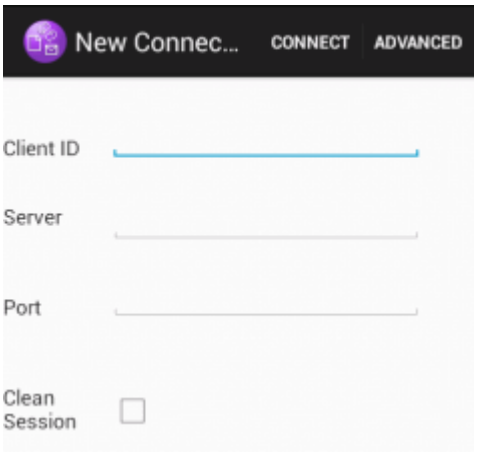
6. Bir konuya bağlanmak, abone olmak ve konuyu yayınlamak için mqttExerciser örnek uygulamasını for Android kullanın.

   a) Androidiçin MQTT istemcisi örnek Java uygulaması' ı açın.

   Bu pencere Android aygıtınızda açık:

   b) Bir MQTT sunucusuna bağlanın.

   i) Yeni bir MQTT bağlantısı açmak için **+** işaretini tıklatın.



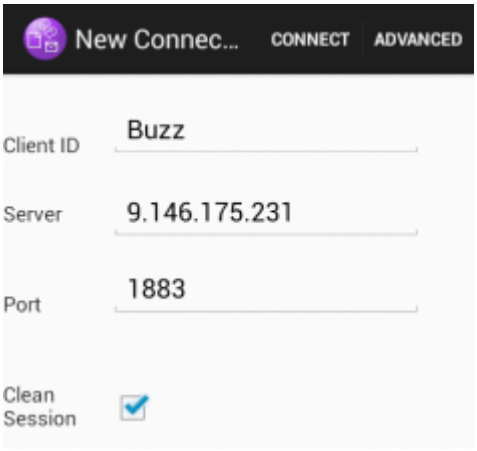
   ii) **İstemci tanıtıcısı** alanına benzersiz bir tanıtıcı girin. Sabırlı olun, tuş vuruşu yavaş olabilir.

   iii) **Sunucu** alanına, MQTT sunucunuzun IP adresini girin.

   Bu, ilk ana adımda seçtiğiniz sunucudur. IP adresi `127.0.0.1`olmamalıdır

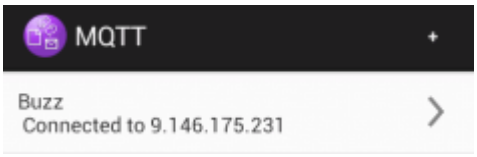
   iv) MQTT bağlantısının kapı numarasını girin.

   Normal bir MQTT bağlantısı için varsayılan kapı numarası şudur: 1883.



v) **Bağlan**'ı tıklayın.

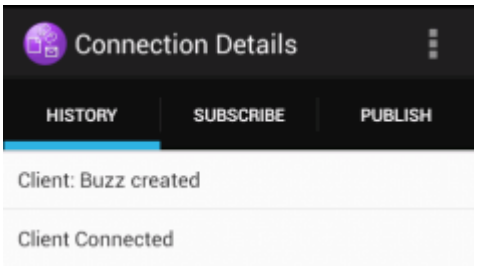
Bağlantı başarılı olursa, bu pencerenin ardından bir "Bağlanıyor" iletisi görürsünüz:



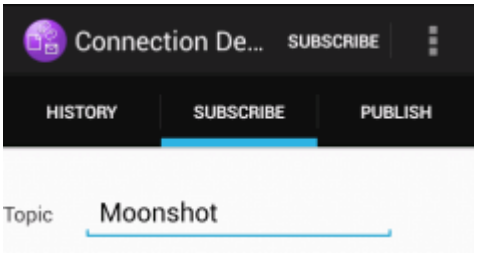
c) Bir konuya abone olun.

i) **Connected** (Bağlı) iletisini tıklatın.

Bağlantı Ayrıntıları penceresi, geçmişin listelendiği şekilde açılır:



ii) **Abone Ol** sekmesini tıklatın ve bir konu dizgisi girin.

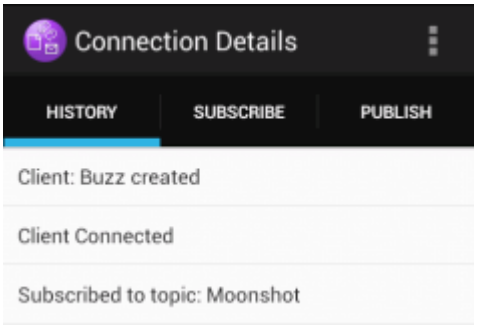


iii) **Abone Ol** işlemini tıklatın.

"Abone olunan" iletisi kısa bir süre için görüntülenir.

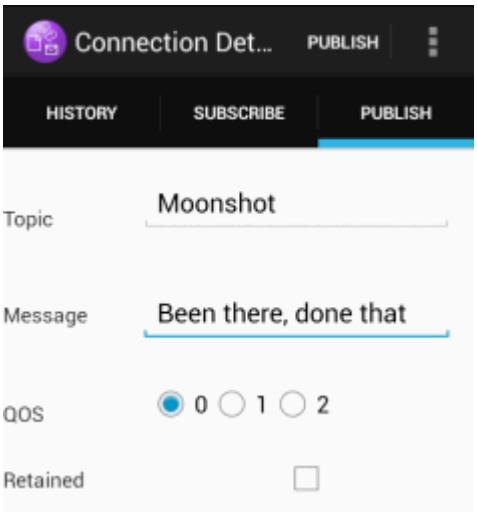
iv) **Geçmiş** sekmesini tıklatın.

Geçmiş artık şu aboneliği içerir:



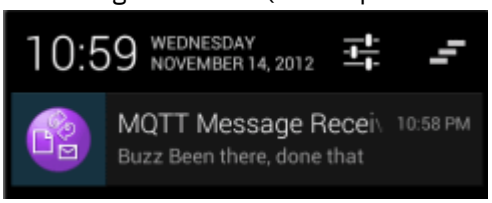
d) Şimdi aynı konuya yayınlayın.

i) **Yayınla** sekmesini tıklatın ve abone olmak için girdiğiniz konu dizgisini girin. Bir ileti girin.

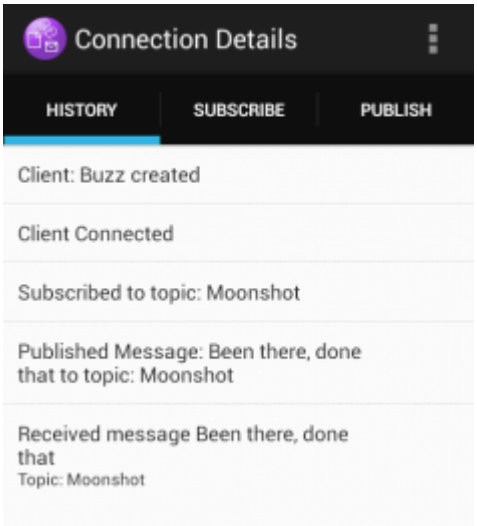


ii) **Yayınla** işlemini tıklatın.

Kısa bir süre için iki ileti görüntülenir: "Yayınlandı" , ardından "Abone Olundu". Yayın, durum alanında görüntülenir (durum penceresini açmak için ayırıcı çubuğunu aşağı çekin).



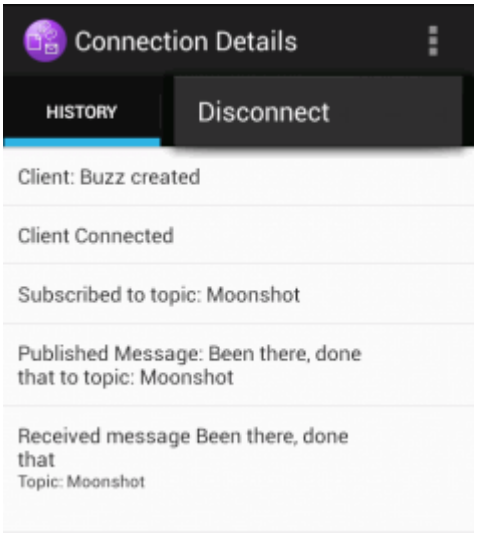
iii) Tüm geçmişi görüntülemek için **Geçmiş** sekmesini tıklatın.



e) İstemci yönetim ortamının bağlantısını kesin.

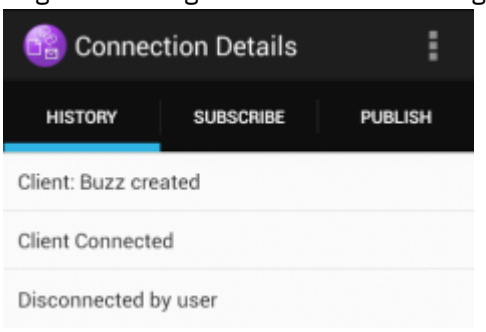
i) Eylem çubuğundaki menü simgesini tıklatın.

Androidiçin MQTT istemcisi örnek Java uygulaması , MQTT **Bağlantı Ayrıntıları** penceresine bir **Bağlantıyı Kes** düğmesi ekler.

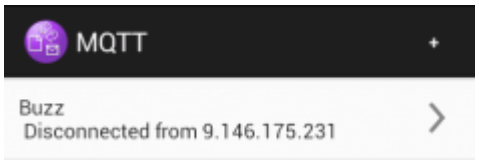


ii) **Bağlantıyı Kes**seçeneğini tıklatın.

Bağlı durum bağlantısı kesildi olarak değişir:



f) MQTT istemcisi örnek Java uygulaması oturumları listesine dönmek için **Geri** düğmesini tıklatın.



- Yeni bir MQTT istemcisi örnek Java uygulaması oturumu başlatmak için artı işaretini tıklatın.
- Bağlantısı kesilen istemciyi tıklatarak yeniden bağlayın.
- Başlatma panosuna dönmek için **Geri** düğmesini tıklatın.

g) Çalışan uygulamaları listelemek için görev düğmesini tıklatın. MQTT istemcisi örnek Java uygulamasıdosyasını bulun. Kapatmak için simgeyi ekrandan kaydırın.

Sonraki adım

Örnek uygulamayı kendiniz oluştururursanız, ileti alışverişi için MQTT kitaplıklarını çağıran kendi Android uygulamalarınızı geliştirmeye başlamaya hazırsınız. Android uygulamalarınızı `mqttExerciser`içindeki sınıflarda modelleyebilirsiniz. Örneği incelemek için, `mqttExerciser` projesindeki `com.ibm.msg.android` ve `com.ibm.msg.android.service` içindeki sınıflar için Javadoc oluşturun.

İlgili bilgiler

Projeleri ADT ile Eclipse içinden yönetme

MQTT messaging client for JavaScriptile çalışmaya başlama

İleti alışverişi istemcisi örnek giriş sayfasını görüntüleyerek ve bağlarının bulunduğu kaynaklara göz atarak MQTT messaging client for JavaScript ile çalışmaya başlayabilirsiniz. To display this home page, you configure an MQTT server to accept connections from the MQTT ileti sistemi istemcisi örnek JavaScript sayfaları, then you type the URL that you have configured on the server into a web browser. MQTT messaging client for JavaScript otomatik olarak aygıtınızda başlar ve ileti alışverişi istemcisi örnek ana sayfası görüntülenir. Bu sayfa, yardımcı programlar, programlama arabirimi belgeleri, eğitmen ve diğer yararlı bilgiler için bağlantılar içerir.

Başlamadan önce

Gelişmiş kullanım için ya da üretimde kullanım için, ileti alışverişi istemcisi örnek ana sayfasını yeniden taramak ya da kaldırmak isteyebilirsiniz. Örnek koddan kaynaklanan kullanıcı arabirimlerinin, herhangi bir erişilebilirlik standardıyla ya da erişilebilirlik koşullarıyla uyumlu olması için garanti verilmediğini lütfen unutmayın.

MQTT messaging client for JavaScript' ı desteklemek için bir MQTT sunucusu gerekir. Bu sunucu, WebSocketsüzerinde MQTT V3.1 iletişim kuralını desteklemelidir. IBM MessageSight, and IBM WebSphere MQ Version 7.5.0, Fix Pack 1 and later versions, support the MQTT protocol over WebSockets. Bkz. “MQTT Server sunucuları ile çalışmaya başlama” sayfa 134. Ücretsiz 90 günlük ücretsiz bir değerlendirme için IBM WebSphere MQ 'i kurmak üzere “kurmaIBM WebSphere MQ” sayfa 136' e bakın.

WebSocket protocol kısa bir süre önce oluşturuldu. İstemciniz ile sunucu arasında bir güvenlik duvarı varsa, bunun WebSockets trafiğini engellmediğini denetleyin. Benzer bir şekilde, tarayıcınız henüz WebSocket protocol' i desteklemiyorsa[1]İstemci yardımcı programını ya da ileti alışverişi istemcisi örnek ana sayfasından kullanılabilir olan eğitmenleri kullanamazsınız. Çizelge 1 sayfa 23 tablosu, en son sürümleri test edilmiş ve ileti sistemi istemcisiyle çalışmak üzere gösterilen tarayıcıları listeler.

Çizelge 1. MQTT messaging client for JavaScriptile kullanım için desteklenen tarayıcılar

Android	**iOS**	**Linux**	**Windows**
Firefox for Android 19.0 ve sonraki Chrome for Android 25.0 ve sonraki	Safari 6.0 ve üstü Chrome 14.0 ve sonraki	Firefox 6.0 ve üstü Chrome 14.0 ve sonraki	Firefox 6.0 ve üstü Chrome 14.0 ve sonraki yayın düzeyi

Bu görev hakkında

Bu görevdeki adımların çoğu, MQTT sunucusunu yapılandırmakta bulunur. JavaScript 'in ileti alışverişi istemcisine erişmek için gereken tüm tüm bilgiler, WebSocket protocol' u destekleyen bir tarayıcı işletebilmektedir.

IBM WebSphere MQ'ta, örnek kanalları yaratarak IBM WebSphere MQ Telemetry ' i etkinleştirmek için aşağıdaki adımları izleyin. 1883 numaralı kapıdaki örnek varsayılan MQTT WebSockets kanalına bağlanın. İleti alışverişi istemcisi örnek ana sayfası URL adresi IBM WebSphere MQüzerinde `http://`*`hostname`*`:1883` olur.

IBM MessageSight' ta, aracı kurun ve ayarlayın, ileti alışverişi merkezini bağlantıları kabul edecek şekilde yapılandırın ve bir MQTT WebSockets uç noktası oluşturun.

[1] Özellikle, RFC 6455 (WebSocket) standardını desteklemiyorsa.

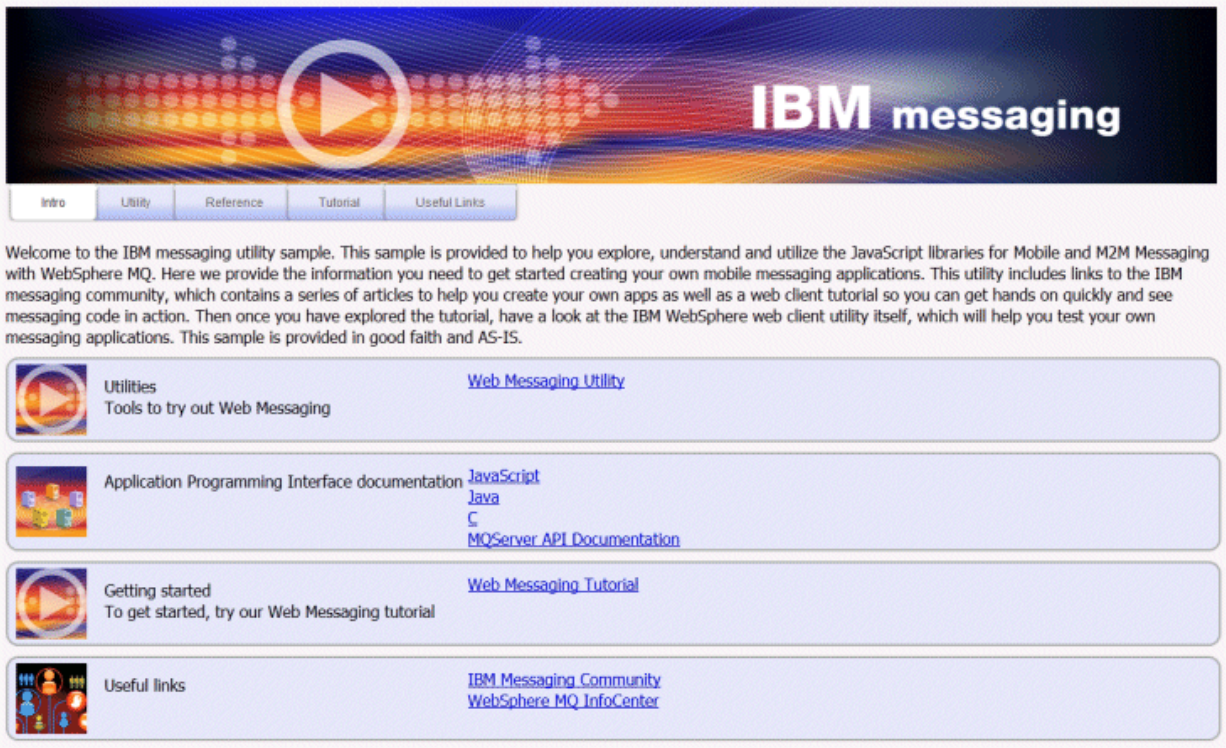
Yordam

1. Mobil İleti Sistemi ve M2M Client Packdosyasını karşıdan yükleyin ve istemci uygulamasını bağlayabileceğiniz bir MQTT sunucusu seçin.

   Bkz. “MQTT istemcilerini kullanmaya başlama” sayfa 9.
2. Configure your MQTT server to accept connections from the MQTT messaging client for JavaScript sample HTML pages.
   - IBM WebSphere MQ'ta:
     - MQTTiçin yapılandırılmış bir IBM WebSphere MQ kuyruk yöneticisiniz varsa, kanal tanımlamasındaki protokolü hem MQTT hem de HTTP ' yi destekleyecek şekilde değiştirin. Bkz. **ALTER CHANNEL**.
     - Bir IBM WebSphere MQ kuyruk yöneticisi oluşturmak ve örnek MQTT WebSockets uç noktasını yapılandırmak için aşağıdaki görevlerden birini gerçekleştirin:
       - “Configuring the MQTT service from the command line” sayfa 138
       - “Configuring the MQTT service with IBM WebSphere MQ Explorer” sayfa 140
3. Aygıtınızda bir web tarayıcısı açın.
4. İleti alışverişi istemcisi örnek ana sayfasının URL adresini yazın.
   - IBM WebSphere MQüzerinde, bu `http://hostname:1883`
   - IBM MessageSightüzerinde, bu `http://hostname:port`

     Burada *hostname* , istemcinin bağlanacağı uç nokta olarak IBM MessageSight aygıtınızda yapılandırdığınız Ethernet yuvasının DNS adı ya da IP adresidir ve *port* , istemciye ilişkin uç noktaya atadığınız TCP/IP kapı numarasıdır.

İleti alışverişi istemcisi örnek ana sayfası görüntülenir.



Şekil 9. MQTT messaging client for JavaScript örnek ana sayfası

Sonuçlar

WebSocketsiçin bir MQTT kanalı yapılandırdınız.

In the sample home page of the MQTT messaging client for JavaScript, click **Web Messaging Yardımcı Programı** to try out different functions in the messaging client API. Örneğin, kuyruk yöneticisine bağlanabilirsiniz, iletilere abone olabilir, sonra bazı iletiler yayınlayabilirsiniz. JavaScript için MQTT ileti sistemi istemcisini çağıran bir web sayfasının nasıl oluşturulacağı hakkında bilgi edinmek için **Web Messaging Öğretici Programı** simgesini de tıklatabilirsiniz.

İlgili kavramlar

"MQTT messaging client for JavaScript ve web uygulamaları" sayfa 115

"How to program messaging apps in JavaScript" sayfa 119

İlgili görevler

"MQTT messaging client for JavaScript ile SSL ve WebSocketsarasındaki bağlantı kurulması" sayfa 76

Connect your web app securely to IBM WebSphere MQ by using the MQTT messaging client for JavaScript sample HTML pages with SSL and the WebSocket protocol.

C için MQTT istemcisiyle çalışmaya başlama

C kaynağını derleyebileceğiniz herhangi bir platform üzerinde C için örnek MQTT istemcisiyle çalışır ve çalışır. Örnek MQTT Client for C ' yi either IBM MessageSight or IBM WebSphere MQ as the MQTT serverile çalıştırabildiğinizi doğrulayın.

Başlamadan önce

- İstemciniz ve sunucu arasında bir güvenlik duvarı varsa, bunun MQTT trafiğini engellediğinden emin olun.
- C platformları için desteklenen ve referans MQTT istemcisi için. Bkz. IBM Mobil İleti Sistemi ve M2M Client Packiçin sistem gereksinimleri.

Bu görev hakkında

Follow this task to compile and run the sample MQTT client for C on Windows from the command line or from Microsoft Visual Studio 2010. Microsoft Visual Studio 2010, istemciyi komut satırı örneğinde derlemek için de kullanılır. Komut satırı komut dosyalarını, diğer platformlarda derlemek ve çalıştırmak için komut satırı komut dosyalarını değiştirin.

Yordam

1. İstemci uygulamasını bağlayabileceğiniz bir MQTT sunucusu seçin.

   Sunucu, MQTT version 3.1 iletişim kuralını desteklemelidir. IBM WebSphere MQ ve IBM MessageSightde içinde olmak üzere, tüm MQTT sunucuları IBM ' dan bunu yapar. Bkz. "MQTT Server sunucuları ile çalışmaya başlama" sayfa 134.
2. Oluşturmadığınız platforma bir C geliştirme ortamı kurun.

   Bu konudaki örneklerdeki makefiles, aşağıdaki araçları içerir:

   - iOS For iOS, on Apple Mac with OS X 10.8.2 with the iOS development tools from Xcode.
   - Linux Linuxiçin, gcc sürüm 4.4.6 , Red Hat® Enterprise Linux sürümünden 6.2.

     The minimum supported level of the `glibc` C library is 2.12, and of the Linux kernel is 2.6.32.
   - Windows Microsoft Windows, Visual Studio sürüm 10.0için.
3. Mobil İleti Sistemi ve M2M Client Pack dosyasını yükleyin ve MQTT SDK ' yı kurun.

   Kuruluş programı yok, yalnızca karşıdan yüklenen dosyayı genişletiyorsunuz.

   a. Mobil İleti Sistemi ve M2M Client Pack' ı karşıdan yükleyin.

b. SDK ' yı kurabildiğiniz bir klasör oluşturun.

   MQTTklasörünün adını vermek isteyebilirsiniz. Bu klasöre giden yol, burada *sdkroot*olarak adlandırılır.

c. Sıkıştırılmış Mobil İleti Sistemi ve M2M Client Pack dosyasının içeriğini *sdkroot*' e genişletin. Genişletme, *sdkroot*\SDK' ta başlayan bir dizin ağacı oluşturur.

4. İsteğe bağlı: Follow the steps in “C kitaplıkları için MQTT istemcisi oluşturulması” sayfa 29.

   Bu adımı, Mobil İleti Sistemi ve M2M Client Pack hedef altyapınıza ilişkin C istemci kitaplığını içermiyorsa kullanın.

5. MQTT istemcisi örnek C uygulamasını derleyin ve çalıştırın `MQTTV3Sample.c`.

   - Komut satırından “Compile and run the MQTT client sample C app from the command line” sayfa 26içindeki adımları izleyin.
   - Bir IDE 'den “MQTT istemcisi örnek C uygulamasını Microsoft Visual Studio 'dan derleyin ve çalıştırın” sayfa 27' taki adımları izleyin.

Compile and run the MQTT client sample C app from the command line

Compile and run the MQTT client sample C app from the command line. Örnek MQTT SDK ' de yer alıyor. Bir MQTT yayınlayıcısını ve aboneyi gösterir.

Başlamadan önce

Bir C geliştirme ortamı kurun; örneğin, örnekte kullanıldığı gibi Microsoft Visual Studio 2010.

Bu görev hakkında

C örneğini (`MQTTV3Sample`SDK istemcileri altdizininde, *sdkroot*`\SDK\clients\c\samples`) derleyin ve çalıştırın.

Yordam

Seçilen altyapınızda `Sample` derlemek ve çalıştırmak için istemci örnekleri dizininde bir komut kütüğü yaratın.

The following script compiles and runs the sample on a Windows 32-bit platform, built with Microsoft Visual Studio 2010. Komut dosyasını *sdkroot*`\SDK\clients\c\samples` alt dizininden çalıştırın.

```
@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
```

Sonuçlar

Yayıncı ve abonenin komut pencerelerine çıkış yazılması:

```
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 . . .
```

Şekil 10. Yayınlayıcıdan çıkış

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

Şekil 11. Aboneden çıkış

MQTT istemcisi örnek C uygulamasını Microsoft Visual Studio 'dan derleyin ve çalıştırın

Compile and run the MQTT client sample C app from the Microsoft Visual Studio. Örnek, Mobil İleti Sistemi ve M2M Client Pack' de yer alıyor. Bir MQTT yayınlayıcısını ve aboneyi gösterir.

Başlamadan önce

Örnek, Microsoft Visual Studio 2010 olanağını kullanır. Diğer C geliştirme ortamlarını Windows ve diğer altyapılarda da kullanabilirsiniz; örneğin, Eclipse IDE for C/C++ Developers.

Bu görev hakkında

Compile and run the C sample, `MQTTV3Sample` with Microsoft Visual Studio 2010. `MQTTV3Sample.c` , SDK istemcileri altdizinidir, *sdkroot*`\SDK\clients\c\samples`.

Yordam

1. Microsoft Visual Studio olanağını başlatın.
2. Var olan koddan yeni bir proje yaratır.
   a) **Dosya** > **Yeni** > **Var Olan Koddan Proje**seçeneklerini tıklatın.
   b) Yaratılacak proje tipi olarak **Görsel C++** öğesini seçin.
   c) **İleri**'yi tıklatın.
3. Specify the parameters in the **Proje Konumu ve Kaynak Dosyaları** window.
   a) **Göz At** ' ı tıklatın ve *sdkroot*`\SDK\clients\c\samples` dizinini bulun.
   b) `MQTTV3Sample`projesinin adını yazın.
   c) **İleri**'yi tıklatın.
4. **Proje tipi** listesinde **Konsol uygulaması projesi** seçeneğini belirleyin. **Son**düğmesini tıklatın.
5. Yalnızca hata ayıklama yapılandırmasını yapılandırın.

   Varsayılan olarak, Microsoft Visual Studio hem bir yayın düzeyi hem de hata ayıklama yapılandırması yaratır. Eğitmende, hata ayıklama yapılandırmasını yapılandırıyorsunuz. Oluşturma hatalarını bastırmak için, yayın yapılandırması için **Oluştur** seçeneğini temizleyin.

   a) **Proje** > **MQTTV3Sample Özellikler** > **Yapılanış Yöneticisi**öğelerini tıklatın. **Etkin çözüm yapılandırması** olarak **Serbest bırak** seçeneğini belirleyin ve **Oluştur**seçeneğini temizleyin.
   b) **Etkin çözüm yapılandırması** > **Kapat**olarak **Hata Ayıkla** seçeneğini belirleyin.

      Aşağıdaki tüm adımlarda Hata Ayıklama yapılanışını değiştirdiğinizi denetleyin.
6. **MQTTV3Sample Özellik Sayfaları**içindeki **C/C++** ayarlarını değiştirin.
   a) **MQTTV3Sample Özellik Sayfaları** penceresinde, **Yapılandırma özellikleri** > **C/C++** > **Genel**' i açın.
   b) Genel özellikler listesinde **Additional Include Directories**(Ek İçerme Dizinleri) seçeneğini tıklatın ve dizin yolunuzu *sdkroot*`\SDK\clients\c\include` olarak ekleyin ve **Apply**(Uygula) düğmesini tıklatın.
7. **Linker** ayarlarını değiştirin
   a) **Yapılandırma özellikleri** > **Linker** > **Genel**' ı açın.

b) Genel özellikler listesinde **Additional Library Directories**(Ek Kitaplık Dizileri) seçeneğini tıklatın ve dizin yolunuzu *sdkroot*\SDK\clients\c\windows_ia32dizinine ekleyin.

c) Linker özellikleri listesinde **Command line**(Komut satırı) seçeneğini tıklatın. **Ek seçenekler** veri girdisi alanına `mqttv3c.lib` yazın ve **Uygula**seçeneğini tıklatın.

8. Remove the `MQTTV3SSample.c` source file from the project.

a) **Solution Explorer** (Çözüm Gezgini) penceresindeki **MQTTV3sample** > **Source Files** (Kaynak Dosyaları) klasörünü açın.

b) **MQTTV3SSample.c** > **Projeden Dışlama**öğelerini sağ tıklatın.

9. `MQTTV3Sample` projesini oluşturun.

a) Çözüm Gezgini 'nde **MQTTV3sample** projesini farenin sağ düğmesiyle tıklatın ve **Oluştur**' u tıklatın.

Oluşturma işlemi hatasız tamamlanır.

10. Hem bir Subscriber hem de Publisher yürütme ortamı olarak **`MQTTV3Sample`** komutunu çalıştırmak için iki yeni proje ekleyin.

Publisher ve Abone projeleri, **`MQTTV3Sample`**komutunu çalıştırmak için gereken komutları içermekte. Bunlar oluşturulmaz ve kod içermez.

a) **Solution Explorer**(Çözüm Gezgini) içinde **Solution ` MQTTV3Sample `** > **Add** > **New project**öğelerini sağ tıklatın.

b) **Ad** alanına `Subscriber` yazın. **Win32 Konsol Uygulaması** seçeneğini seçili olarak bırakın. **Tamam**'ı tıklatın.

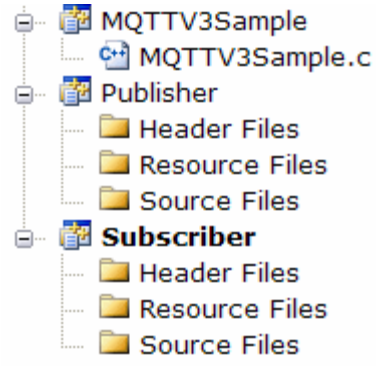
Win32 Application Wizard (Uygulama Sihirbazı) başlatılır.

c) **Win32 Application Wizard**(Uygulama Sihirbazı) içinde **Next**(İleri) düğmesini tıklatın. **Boş proje** > **Son**seçeneğini işaretleyin.

d) Bir Publisher projesi eklemek için bu adımları yineleyin.

e) Abone projesini farenin sağ düğmesiyle tıklatın ve **Ayarla StartUp projesi olarak belirle**öğesini seçin.

Çözüm Gezgini penceresi Şekil 12 sayfa 28içinde gösterilir.



Şekil 12. MQTTV3Sample çözümü

11. Abone özelliği sayfalarını yapılandırın.

a) Right-click **Abone** in the Solution Explorer **Özellikler** > **Yapılandırma Özellikleri** > **Özellikler** > **Hata ayıklama**

Pencere başlığının **Abone Özellik Sayfaları**' na bakın.

b) **Ortam**' ı tıklatın. `path=%path%;`*sdkroot*`\SDK\clients\c\windows_ia32` yazın ve **Uygula**düğmesini tıklatın.

sdkroot ' u ortamınıza uygun olarak değiştirin.

c) Click **Komut**, and replace `$(TargetPath)` by the path to the `MQTTV3Sample` module

Örneğin, *sdkroot*`\SDK\clients\c\samples\debug\MQTTV3Sample`

sdkroot ' u ortamınıza uygun olarak değiştirin.

d) **Komut Bağımsız Değişkenleri** 'ne ve `-a subscribe -b localhost -p 1883` yazıp **Uygula**' yı tıklatın.

12. Publisher özellik sayfalarını yapılandırın.

    İpucu: You can switch the property pages project by clicking the projects in the **Çözüm Gezgini** window.

    a) Yayınlayıcıya ilişkin Abone adımlarını yineleyin.

    Komut bağımsız değişkeni şöyledir: `-b localhost -p 1883`

13. Publisher ve Abone projelerini oluşturan oluşturma işlemini durdurun.

    a) Projelerin herhangi birinin özellik sayfalarında **Configuration Manager** seçeneğini tıklatın ve Yayıncı ve Abone için hem Yayın hem de Hata Ayıklama yapılandırmalarında **Oluştur** seçeneğini temizleyin. **Kapat**'ı tıklatın.

14. Örneği çalıştırın.

    a) Aboneyi başlatmak için **F5** düğmesini tıklatın.

    b) Çözüm Gezgini 'nde **PublisherYayınlayıcı** ' ı sağ tıklatın, **Hata Ayıkla** > **Yeni bir yönetim ortamı başlat**

Sonuçlar

Yayınlayıcı ve abone çıkışı komut pencerelerinde. Visual Studio yayınlayıcı penceresini kapatır. Aşağıdaki şekilde gösterilen abone penceresine bakın ve daha sonra, abone penceresini kapatın.

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

Şekil 13. Aboneden çıkış

Sonraki adım

Zamanuyumsuz yayınlayıcı ve aboneyi oluşturun ve çalıştırın. Örnekler, *sdkroot*`\SDK\clients\c\samples`içinde `MQTTV3ASample.c` ve `MQTTV3ASSample.c` ' dir.

C kitaplıkları için MQTT istemcisi oluşturulması

C kitaplıkları için MQTT istemcisini oluşturmak için bu adımları izleyin. Bu konuda, bir dizi platformun derleme ve bağlantı anahtarları ve iOS ve Windowsüzerindeki kitaplıkların oluşturulmasına ilişkin örnekler yer alır.

Başlamadan önce

1. C istemcisi kitaplığını yalnızca gerektiğinde oluşturun. (Software Development Kit) SDK içindeki önceden oluşturulmuş istemci kitaplıklarını, hedef altyapınızla eşleşen bir `SDK\clients\c` alt dizininde bağlayın.
2. MQTT istemcisi örnek C uygulamasısunucusu ile oluşturdugunuz kitaplığı sınamak için bir MQTT sunucusu yapılandırın. Bkz. “MQTT Server sunucuları ile çalışmaya başlama” sayfa 134. MQTT istemcisi örnek uygulamalarından birini çalıştırarak sunucu yapılandırmasını doğrulayın.
3. C kitaplığının güvenli bir sürümünü oluşturuyorsanız (Secure Sockets Layer) destekler SSL, OpenSSL kitaplığını da oluşturmalısınız. Bkz. “OpenSSL paketini oluşturma” sayfa 43.

   Önemli: OpenSSL paketinin karşıdan yüklenmesi ve yeniden dağıtılması, katı içe aktarma ve dışa aktarma düzenlemelerine ve açık kaynak lisanslama koşullarına tabidir. Paketi karşıdan yüklemeye karar vermeden önce kısıtlamalara ve uyarılara dikkat edin.

Bu görev hakkında

OpenSSL kitaplığını karşıdan yüklemek ve oluşturmak için “OpenSSL paketini oluşturma” sayfa 43 içindeki yönergeleri izleyin. C kitaplığı için MQTT istemcisinin güvenli bir sürümünü oluşturmak için OpenSSL ' i oluşturmanız gerekir. OpenSSL ' in MQTT kitaplığı için güvenli olmayan bir sürümünü oluşturmasını zorunlu kılmasınız. Bu adımlarda, iOS ve Windowsiçin kitaplığın oluşturulmasına ilişkin örnekler yer alır.

C geliştirme kitaplığı araçlarını ve MQTT yazılım geliştirme araç takımını (SDK) oluşturma platformunuza yükleyerek, C kitaplığı için MQTT istemcisini oluşturun. Hedef altyapınıza ilişkin kitaplığı oluşturmak için bir makefile yazın ve “Farklı platformlar içinMQTT oluşturma seçenekleri” sayfa 31içinde belgelenmiş olan seçenekleri ekleyin. Bir makefile oluşturmak ve çalıştırmak için platforma özgü adımlar burada verilmiştir:

- iOS “Building the MQTT client libraries for C on an Apple Mac for use with iOS devices” sayfa 32
- Windows “Windowsüzerinde MQTT kitaplıklarının oluşturulması” sayfa 38

Yordam

1. Oluşturmadığınız platforma bir C geliştirme ortamı kurun.

   Bu konudaki örneklerdeki makefiles, aşağıdaki araçları içerir:

   - iOS For iOS, on Apple Mac with OS X 10.8.2 with the iOS development tools from Xcode.
   - Linux Linuxiçin, gcc sürüm 4.4.6 , Red Hat Enterprise Linux sürümünden 6.2.

     The minimum supported level of the `glibc` C library is 2.12, and of the Linux kernel is 2.6.32.
   - Windows Microsoft Windows, Visual Studio sürüm 10.0için.
2. Mobil İleti Sistemi ve M2M Client Pack dosyasını yükleyin ve MQTT SDK ' yı kurun.

   Kuruluş programı yok, yalnızca karşıdan yüklenen dosyayı genişletiyorsunuz.

   a. Mobil İleti Sistemi ve M2M Client Pack' ı karşıdan yükleyin.
   b. SDK ' yı kurabildiğiniz bir klasör oluşturun.

      `MQTT`klasörünün adını vermek isteyebilirsiniz. Bu klasöre giden yol, burada *`sdkroot`*olarak adlandırılır.
   c. Sıkıştırılmış Mobil İleti Sistemi ve M2M Client Pack dosyasının içeriğini *`sdkroot`*' e genişletin. Genişletme, *`sdkroot`*`\SDK`' ta başlayan bir dizin ağacı oluşturur.
3. C kitaplıkları için MQTT istemcisine ilişkin kaynak kodu açın.

   Kaynak kodu sıkıştırılmış dosya: *`sdkroot`*`\SDK\clients\c\source.zip`.
4. İsteğe bağlı: Build OpenSSL.

   Bkz. “OpenSSL paketini oluşturma” sayfa 43.
5. C kitaplıkları için MQTT istemcisini oluşturun.

   Kitaplıkların oluşturulmasına ilişkin komutlar ve seçenekler “Farklı platformlar içinMQTT oluşturma seçenekleri” sayfa 31içinde listelenir.

   Follow the steps in the following examples to write a makefile to build the MQTT client for C libraries for your target platform.

   - “Building the MQTT client libraries for C on an Apple Mac for use with iOS devices” sayfa 32
   - “Windowsüzerinde MQTT kitaplıklarının oluşturulması” sayfa 38

Farklı platformlar içinMQTT oluşturma seçenekleri

Aşağıdaki tabloda, çeşitli platformlarda C kitaplıkları için MQTT istemcisi oluşturmak üzere derleyici ve oluşturma seçenekleri listelenmektedir.

Çizelge 2. Farklı platformlar içinMQTT oluşturma seçenekleri

<table>
<tr><th>Altyapı</th><th>Compiler</th><th>Compiler Options</th><th>Linker Options</th><th>Extra Options</th></tr>
<tr><td>AIX</td><td rowspan="5">gcc</td><td rowspan="2">-fPIC -Os -Wall
-DREVERSED
-I MQTTCLIENT_DIR</td><td>-Wl,-G</td><td></td></tr>
<tr><td>Linux s390x</td><td rowspan="5">-shared -Wl,-soname,
libmqttv3c.so</td><td rowspan="2">-m
64</td></tr>
<tr><td>Linux
x86-64</td><td rowspan="4">-fPIC -Os -Wall -I
MQTTCLIENT_DIR</td></tr>
<tr><td>Linux
x86-32</td><td>-m
32</td></tr>
<tr><td>Linux ARM
(glibc)</td><td>arm-linux-gcc</td><td rowspan="2"></td></tr>
<tr><td>Linux ARM
(uclibc)</td><td>arm-unknown-
linux
-uclibcgnueabi-
gcc</td></tr>
<tr><td>Windows 32
bit</td><td>cl</td><td>/D "WIN32" /D
"_UNICODE" /D
"UNICODE"
/D
"_CRT_SECURE_NO_WAR
NINGS"
/
nologo /c /O2 /W3 /
Fd /MD /TC</td><td>/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)</td><td></td></tr>
</table>

Çizelge 2. Farklı platformlar içinMQTT oluşturma seçenekleri (devamı var)

Altyapı	Compiler	Compiler Options	Linker Options	Extra Options
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`			
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 *Building the MQTT client libraries for C on an Apple Mac for use with iOS devices*

iOS aygıtlarıyla daha sonraki kullanım için Apple Macüzerinde C için MQTT istemci kitaplıklarını oluşturmak üzere bir makefile yazmak için bu adımları izleyin.

Başlamadan önce

1. Install build tools, develop, and run the makefile on Apple Mac with OS X 10.8.2, or later.
2. Install the command-line tools for Xcode, which include the **make** program. Komut satırı araçlarını Xcodekodundan yükleyin.

Bu görev hakkında

Bir ARMv7 ya da ARMv7s işlemcisi çalıştıranCfor iPhone ya da iPad için MQTT istemci kitaplıklarını oluşturan bir makefile ve bir i386-64 bit işlemcisine çalışan iPhone benzetimcisini oluşturun. Bkz. iOS aygıtlarının listesi.

İpucu: “MQTTios.mak makefile listeleme” sayfa 36 , tüm makefile 'ı listeler.

1. Listeyi bir dosyaya kopyalayıp yapıştırın.
2. Convert the leading character of each line that follows a target to a tab; see step “8” sayfa 34.
3. Run it with the command listed in step “9” sayfa 36 of the procedure.

Yordam

1. iOS geliştirme araçlarını karşıdan yükleyin ve kurun.
   a. Yönetici ayrıcalıklarına sahip bir kullanıcı kimliğiyle oturum açın.
   b. Apple Mac ürününüzün 10.8.2 sürümünde ya da sonraki sürümlerinden olup olmadığını denetleyin.
   c. Go to the website Xcode to download Xcode from the Mac app store.
   d. Xcodekomutunu, komut satırı ortamını ve benzetimcisini kurun.

   Mac app store, benzetimcisinin birden çok sürümünü sunarsa, uygulamanız için hedeflediğiniz iOS düzeyiyle uyumlu sürümü seçin.
2. Makefile `MQTTios.mak`yarat

   Öneylem ekle:

   ```
   # Oluşturma çıkışı, yürürlükteki dizinde üretilir.
   # MQTTCLIENT_DIR, MQTT istemci kaynak kodunu içeren temel dizini göstermelidir.
   # Varsayılan MQTTCLIENT_DIR yürürlükteki dizindir
   # Varsayılan TOOL_DIR, /Applications/Xcode.app/Contents/Developer/Platforms
   # Varsayılan OPENSSL_DIR is sdkroot/openssl, sdkroot/sdk/clients/c/mqttv3c/src ile göreli
   # OPENSSL_DIR, OpenSSL oluşturmasını içeren temel dizini göstermelidir.
   # Örnek: make -f MQTTios.mak MQTTCLIENT_DIR=sdkroot/sdk/clients/c/mqttv3c/src tüm
   ```
3. MQTT kaynak kodunun konumunu ayarlayın.

   Makefile 'ı MQTT kaynak dosyalarıyla aynı dizinde çalıştırın ya da `MQTTCLIENT_DIR` komut satırı parametresini ayarlayın:

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

   Aşağıdaki satırları makefile (makedfile) dosyasına ekleyin:

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

   The example sets VPATH to the directory where **make** searches for source files that are not explicitly identified; for example all the header files that are required in the build.
4. İsteğe bağlı: OpenSSL kitaplıklarının yerini ayarlayın.

   Bu adım, C kitaplıklarına ilişkin MQTT istemcisinin SSL sürümlerini oluşturmak için gereklidir.

   MQTT SDK ' yı genişletmiş olduğunuz aynı dizine OpenSSL kitaplıklarının varsayılan yolunu ayarlayın. Otherwise, set `OOPENSSL_DIR` as a command-line parameter.

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

   İpucu: *OpenSSL* , tüm OpenSSL alt dizinlerini içeren OpenSSL dizinidir. Dizin ağacını genişletdiğiniz yerden taşımak zorunda kalabilirsiniz, çünkü bu ağaç gereksiz boş üst dizinler içeriyor.
5. Geliştirme araçları dizinlerini ayarlayın.

   Xcode 'yi farklı bir yere kurdıysanız, komut satırında `TOOL_DIR` ' i ayarlayın.

   ```
   ifndef TOOL_DIR
      TOOL_DIR = /Applications/Xcode.app/Contents/Developer/Platforms endif
   IPHLIE_SDK = iPhoneOS.platform/Developer/SDKs/iPhoneOS6.0.sdk
   ```

```
IPHESESIM_SDK = iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator6.0.sdk
SDK_ARM = ${TOOL_DIR}/${IPHONE_SDK}
SDK_i386 = ${TOOL_DIR}/${IPHONESIM_SDK}
```

6. Her bir MQTT kitaplığını oluşturmak için gerekli olan tüm kaynak dosyaları seçin. Ayrıca, oluşturmak için MQTT kitaplığının adını ve konumunu ayarlayın.

   Tüm MQTT kaynak dosyalarını listelemek için aşağıdaki satırı makefile 'a ekleyin:

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

   Kaynak dosyalar, zamanuyumlu ya da zamanuyumsuz bir kitaplık mı oluşturuyorsanız ve kitaplığın SSL ' yi içerip içermediğine bağlı olarak değişir.

   Oluşturmak için hedeflere bağlı olan bu satırlardan bir ya da daha fazlasını ekleyin. Paylaşılan kitaplıklar `darwin_x86_64` dizininde yaratılır.

   - Zamanuyumlu, güvenli olmayan:

     ```
     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
     ```

   - Zamanuyumlu, güvenli:

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

   - Zamanuyumsuz, güvenli olmayan:

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

   - Zamanuyumsuz güvenli:

     ```
     MQTTLIB_AS = mqttv3as
     KAYNAK_DOSYA_AS = ${ süzgeç-çıkış ${MQTTCLIENT_DIR}/MQTTClient.c, ${ALL_SOURCE_FILES}}
     MQTTLIB_DARWIN_AS = darwin_x86_64/lib${MQTTLIB_AS} .a
     ```

7. Derleyici ve derleyici seçeneklerini tanımlayın.

   Farklı platformlar için MQTT oluşturma seçenekleri' nde gösterilen farklı platformlara ilişkin seçeneklere bakın.

   a) Set Gnu project C and C++ (**gcc**) as the compiler.

      Farklı aygıtlar ve iPhone benzetimcisi için kitaplığı oluşturmak üzere üç çapraz derleyici seçin:

      ```
      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) Derleyici seçeneklerini ekleyin.

      ```
      CCFLAGS = -Os -Wall-fomit-çerçeve-işaretçisi
      ```

   c) İçerme yollarını ekleyin.

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

8. Oluşturma hedeflerini tanımlayın.

   İpucu: Bir hedefin somutlamasını tanımlayan her ardışık satır bir sekme karakteriyle başlamalıdır.

a) **all** hedefini tanımlayın.

"all" hedefi tüm kitaplıkları oluşturur.

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

İlk önce listeleyerek varsayılan hedeftir.

a) Build the synchronous, unsecured library, `libmqttv3c.a`.

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

`rm *.o` deyimi, her kitaplık için yaratılmış tüm nesne dosyalarını siler. `lipo` , üç kitaplığı tek bir dosyada birleştirir.

b) Zamanuyumsuz, güvenli olmayanlibrarykitaplığını oluşturun. `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 -syslibroot ${SDK_ARM} -o $@.armv7s *.o
    rm *.o
    ${CC_i386} ${CCFLAGS_SO_i386} -c ${SOURCE_FILES_A} -DMQTT_ASYNC -DUSE_NAMED_SEMAPHOES
-DNOSIGPIPE
    libtool -static -syslibroot ${SDK_i386} -o $@.i386 *.o
    rm *.o
    lipo -create $@.armv7 $@.armv7s $@.i386 -output $@
```

`rm *.o` deyimi, her kitaplık için yaratılmış tüm nesne dosyalarını siler. `lipo` , üç kitaplığı tek bir dosyada birleştirir.

c) Zamanuyumlu, güvenli kitaplık oluşturun, `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
    lipo -create $@.armv7 $@.armv7s $@.i386 -output $@
```

`rm *.o` deyimi, her kitaplık için yaratılmış tüm nesne dosyalarını siler. `lipo` , üç kitaplığı tek bir dosyada birleştirir.

d) Zamanuyumsuz, güvenli kitaplığı oluşturun, `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_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_AS} -DMQTT_ASYNC -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_AS} -DMQTT_ASYNC -DOPENSSL
-DUSE_NAMED_SEMAPHOES -DNOSIGPIPE
    libtool -static -syslibroot ${SDK_ARM} -L${OPENSSL_DIR}/i386 -lssl -lcrypto -o
$@.i386 *.o
    rm *.o
    lipo -create $@.armv7 $@.armv7s $@.i386 -output $@
```

`rm *.o` deyimi, her kitaplık için yaratılmış tüm nesne dosyalarını siler. `lipo` , üç kitaplığı tek bir dosyada birleştirir.

e) **`clean`** hedefini tanımlayın.

"clean" hedefi, makefile tarafından oluşturulan tüm dosyaları ve dizinleri kaldırır.

```
.Temiz.
temizle:
    -rm -f *.obj
    -rm -f -r darwin_x86_64
```

9. Makefile 'ı çalıştırın.

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

Sonuçlar

Aşağıdaki dosyalar *sdkroot/sdk/clients/c/mqttv3c/src*/darwin_x86_64 dizininde oluşturulur.

```
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
```

`MQTTios.mak` makefile listeleme

```
# Oluşturma çıkışı, yürürlükteki dizinde üretilir.
# MQTTCLIENT_DIR, MQTT istemci kaynak kodunu içeren temel dizini göstermelidir.
# Varsayılan MQTTCLIENT_DIR yürürlükteki dizindir
# Varsayılan TOOL_DIR, /Applications/Xcode.app/Contents/Developer/Platforms
# Varsayılan OPENSSL_DIR is sdkroot/openssl, sdkroot/sdk/clients/c/mqttv3c/src ile göreli
# OPENSSL_DIR, OpenSSL oluşturmasını içeren temel dizini göstermelidir.
# Örnek: make -f MQTTios.mak MQTTCLIENT_DIR=sdkroot/sdk/clients/c/mqttv3c/src tüm

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 = ${ joker ${MQTTCLIENT_DIR}/* .c }
ifndef TOOL_DIR
   TOOL_DIR = /Applications/Xcode.app/Contents/Developer/Platforms endif
IPHLIE_SDK = iPhoneOS.platform/Developer/SDKs/iPhoneOS6.0.sdk
IPHESESIM_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 = ${ süzgeç-${MQTTCLIENT_DIR}/MQTTAsync.c, ${ALL_SOURCE_FILES}}
MQTTLIB_DARWIN_S = darwin_x86_64/lib${MQTTLIB_S} .a
MQTTLIB_A = mqttv3a
KAYNAK_DOSYA_A = ${ süzgeç-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
KAYNAK_DOSYA_AS = ${ süzgeç-çıkış ${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-çerçeve-işaretçisi
CCFLAGS_SO_ARM = ${CCFLAGS} -isysroot ${SDK_ARM} -I${OPENSSL_DIR}/include -L${SDK_ARM}/usr/lib/
system
CCFLAGS_SO_i386 = ${CCFLAGS} -isysroot ${SDK_i386} -I${OPENSSL_DIR}/include -L$
{SDK_i386}/usr/lib/system

# targets
tümü: ${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_SEMAPHOES -DNOSIGPIPE
    libtool -static -syslibroot ${SDK_ARM} -o $@.armv7 *.o
    rm *.o
    ${CC_armv7s} ${CCFLAGS_SO_ARM} -c ${SOURCE_FILES} -DUSE_NAMED_SEMAPHOES -DNOSIGPIPE
    libtool -static -syslibroot ${SDK_ARM} -o $@.armv7s *.o
    rm *.o
    ${CC_i386} ${CCFLAGS_SO_i386} -c ${SOURCE_FILES} -DUSE_NAMED_SEMAPHOES -DNOSIGPIPE
    libtool -static -syslibroot ${SDK_i386} -o $@.i386 *.o
    rm *.o
    lipo -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 -syslibroot ${SDK_ARM} -o $@.armv7s *.o
    rm *.o
    ${CC_i386} ${CCFLAGS_SO_i386} -c ${SOURCE_FILES_A} -DMQTT_ASYNC -DUSE_NAMED_SEMAPHOES
-DNOSIGPIPE
    libtool -static -syslibroot ${SDK_i386} -o $@.i386 *.o
    rm *.o
    lipo -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
    lipo -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_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_AS} -DMQTT_ASYNC -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_AS} -DMQTT_ASYNC -DOPENSSL
-DUSE_NAMED_SEMAPHOES -DNOSIGPIPE
    libtool -static -syslibroot ${SDK_ARM} -L${OPENSSL_DIR}/i386 -lssl -lcrypto -o $@.i386 *.o
    rm *.o
    lipo -create $@.armv7 $@.armv7s $@.i386 -output $@

.Temiz.
temizle:
    -rm -f *.obj
    -rm -f -r darwin_x86_64
```

Windows *Windowsüzerinde MQTT kitaplıklarının oluşturulması*

Bir makefile yazmak için, Windowsüzerinde C için MQTT istemci kitaplıklarını oluşturmak üzere bu adımları izleyin.

Başlamadan önce

1. Gerekiyorsa, oluşturma iş istasyonunuza Gnu için yazılan makefiles ile uyumlu bir **Make** sürümünü kurun; tersi durumda, Gnu dosyasını karşıdan yükleyin ve oluşturun. Bkz. Gnu Make (Yap). The website, Windows için yap, provides an installable version of **Make** for Windows.
2. Ayrıca, makefile örneğindeki temiz hedefi kullanmak için Windows için Linux komutlarına gerek duyarsınız. You can obtain Linux commands for Windows from websites such as Cygwin.

Bu görev hakkında

Create a makefile that builds MQTT client libraries for C for Windows 32 bit.

İpucu: “MQTTwin.mak makefile listeleme” sayfa 42 , tüm makefile 'ı listeler.

1. Listeyi bir dosyaya kopyalayıp yapıştırın.
2. Convert the leading character of each line that follows a target to a tab; see step “7” sayfa 40.
3. Run it with the command listed in step “9” sayfa 41 of the procedure.

Yordam

1. Makefile `MQTTwin.mak`yarat

   Öneylem ekle:

   ```
   # Oluşturma çıkışı, yürürlükteki dizinde üretilir.
   # MQTTCLIENT_DIR, MQTT istemci kaynak kodunu içeren temel dizini göstermelidir.
   # Varsayılan MQTTCLIENT_DIR yürürlükteki dizindir
   # Varsayılan OPENSSL_DIR is sdkroot\openSSLise, sdkroot\sdk\clients\c\mqttv3c\src ile göreli
   # OPENSSL_DIR, OpenSSL oluşturmasını içeren temel dizini göstermelidir.
   # Örnek: make -f MQTTwin.mak MQTTCLIENT_DIR=sdkroot/sdk/clients/c/mqttv3c/src
   # Oluşturma ortamını ayarlayın; örneğin:
   #    %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. MQTT kaynak kodunun konumunu ayarlayın.

   Makefile 'ı MQTT kaynak dosyalarıyla aynı dizinde çalıştırın ya da `MQTTCLIENT_DIR` komut satırı parametresini ayarlayın:

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

   Aşağıdaki satırları makefile (makedfile) dosyasına ekleyin:

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

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

The example sets VPATH to the directory where **make** searches for source files that are not explicitly identified; for example all the header files that are required in the build.

3. İsteğe bağlı: OpenSSL kitaplıklarının yerini ayarlayın.

   Bu adım, C kitaplıklarına ilişkin MQTT istemcisinin SSL sürümlerini oluşturmak için gereklidir.

   MQTT SDK ' yı genişletmiş olduğunuz aynı dizine OpenSSL kitaplıklarının varsayılan yolunu ayarlayın. Otherwise, set 0OPENSSL_DIR as a command-line parameter.

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

   İpucu: *OpenSSL* , tüm OpenSSL alt dizinlerini içeren OpenSSL dizinidir. Dizin ağacını genişletdiğiniz yerden taşımak zorunda kalabilirsiniz, çünkü bu ağaç gereksiz boş üst dizinler içeriyor.

4. Her bir MQTT kitaplığını oluşturmak için gerekli olan tüm kaynak dosyaları seçin. Ayrıca, oluşturmak için MQTT kitaplığının adını ve konumunu ayarlayın.

   Tüm MQTT kaynak dosyalarını listelemek için aşağıdaki satırı makefile 'a ekleyin:

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

   Kaynak dosyalar, zamanuyumlu ya da zamanuyumsuz bir kitaplık mı oluşturuyorsanız ve kitaplığın SSL ' yi içerip içermediğine bağlı olarak değişir.

   Oluşturmak için hedeflere bağlı olan bu satırlardan bir ya da daha fazlasını ekleyin. Paylaşılan kitaplıklar ve bildirgeler, `windows_ia32` dizininde oluşturulur.

   - Zamanuyumlu, güvenli olmayan:

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

   - Zamanuyumlu, güvenli:

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

   - Zamanuyumsuz, güvenli olmayan:

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

   - Zamanuyumsuz güvenli:

```
MQTTLIB_AS = mqttv3as
MQTTDLL_AS = windows_ia32/${MQTTLIB_AS} .dll
KAYNAK_DOSYA_AS = ${ süzgeç-çıkış ${MQTTCLIENT_DIR}/MQTTClient.c, ${ALL_SOURCE_FILES}}
MANIFEST_S = mt -manifest ${MQTTDLL_S}.bildirge -outputresource: ${MQTTDLL_S}\; 2
```

5. Derleyici ve derleyici seçeneklerini tanımlayın.

   Farklı platformlar için MQTT oluşturma seçenekleri' nde gösterilen farklı platformlara ilişkin seçeneklere bakın.

   a) Derleyici olarak Microsoft Visual C++ olarak ayarlayın.

```
CC = cl
```

b) Ön işlemci seçeneklerini ekleyin.

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

c) Derleyici seçeneklerini ekleyin.

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

d) İçerme yollarını ekleyin.

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

e) İsteğe bağlı: Güvenli bir kitaplık oluşturuyorsanız, bir ön işlemci seçeneği ekleyin.

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

f) İsteğe bağlı: Güvenli bir kitaplık oluşturuyorsanız, OpenSSL üstbilgi dosyalarını ekleyin.

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

İpucu: Üstbilgi dosyaları `${OPENSSL_DIR}/inc32/openssl`dosyasıdır, ancak `ssl.h` dosyası "`openssl/ssl.h`"ile birlikte bulunur.

6. Linker 'ı ve linker seçeneklerini ayarlayın.

a) Set Microsoft Visual C++ as the linker.

```
LD = bağlantı
```

b) Linker seçeneklerini ekleyin.

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

c) Kitaplık yollarını ekleyin.

```
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) Ara çıkış dosyalarını ekleyin.

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

e) İsteğe bağlı: Güvenli bir kitaplık oluşturuyorsanız, OpenSSL kitaplıklarını ekleyin.

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

f) İsteğe bağlı: Güvenli bir kitaplık oluşturuyorsanız, OpenSSL kitaplık yolunu ekleyin.

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

7. Dört oluşturma hedefini tanımlayın.

a) **`all`** hedefini tanımlayın.

İpucu: Bir hedefin somutlamasını tanımlayan her ardışık satır bir sekme karakteriyle başlamalıdır.

"all" hedefi tüm kitaplıkları oluşturur.

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

b) Build the synchronous, unsecured library, `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}
```

The statement `-rm ${CURDIR}/MQTTAsync.obj` deletes any `MQTTAsync.obj` created for an earlier target. `MQTTAsync.obj` ve `MQTTClient.obj` karşılıklı olarak birbirini dışlar.

c) Zamanuyumsuz, güvenli olmayanlibrarykitaplığını oluşturun. `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} *.oj /out: ${MQTTDLL_A}
    ${MANIFEST_A}
```

The statement `-rm ${CURDIR}/MQTTClient.obj` deletes any `MQTTClient.obj` created for an earlier target. `MQTTAsync.obj` ve `MQTTClient.obj` karşılıklı olarak birbirini dışlar.

d) Zamanuyumlu, güvenli kitaplık oluşturun, `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} ${LIBPDB} ${LIBMAP} ${WINLIBS_S} ${LIBPATH_S} *.obj /out:
${MQTTDLL_S}
    ${MANIFEST_S}
```

The statement `-rm ${CURDIR}/MQTTAsync.obj` deletes any `MQTTAsync.obj` created for an earlier target. `MQTTAsync.obj` ve `MQTTClient.obj` karşılıklı olarak birbirini dışlar.

e) Zamanuyumsuz, güvenli kitaplığı oluşturun, `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 }
```

The statement `-rm $(CURDIR}/MQTTClient.obj` deletes any `MQTTClient.obj` created for an earlier target. `MQTTAsync.obj` ve `MQTTClient.obj` karşılıklı olarak birbirini dışlar.

f) **`clean`** hedefini tanımlayın.

"clean" hedefi, makefile tarafından oluşturulan tüm dosyaları ve dizinleri kaldırır.

```
.Temiz.
temizle:
    -rm -f *.obj
    -rm -f -r windows_ia32
```

8. Makefile 'ı çalıştırmak için Windows yolunu ayarlayın.

   Kuruluşunuzla eşleştirilecek parçaları italik olarak ayarlayın.

   a) Microsoft Visual Studio ortamını ayarlayın.

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

   b) Set the `Path` variable to include the make program and the Linux command environment.

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

9. Makefile 'ı çalıştırın.

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

İpucu: Dosya ayırıcı karakteri, ters eğik çizgi değil, ters eğik çizgi karakteri olmalıdır.

Sonuçlar

Aşağıdaki dosyalar *sdkroot**SDK**clients**c**mqttv3c**src*\windows_ia32 dizininde oluşturulur.

```
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
```

MQTTwin.mak makefile listeleme

```
# Oluşturma çıkışı, yürürlükteki dizinde üretilir.
# MQTTCLIENT_DIR, MQTT istemci kaynak kodunu içeren temel dizini göstermelidir.
# Varsayılan MQTTCLIENT_DIR yürürlükteki dizindir
# Varsayılan OPENSSL_DIR is sdkroot\openSSLise, sdkroot\sdk\clients\c\mqttv3c\src ile göreli
# OPENSSL_DIR, OpenSSL oluşturmasını içeren temel dizini göstermelidir.
# Örnek: make -f MQTTwin.mak MQTTCLIENT_DIR=sdkroot/sdk/clients/c/mqttv3c/src
# Oluşturma ortamını ayarlayın; örneğin:
#    %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 = ${ joker ${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}}
BILDIRGE = mt -manifest ${MQTTDLL}.bildirge -outputresource: ${MQTTDLL}\; 2
MQTTLIB_S = mqttv3cs
MQTTDLL_S = windows_ia32/${MQTTLIB_S} .dll
SOURCE_FILES_S = ${ süzgeç-${MQTTCLIENT_DIR}/MQTTAsync.c, ${ALL_SOURCE_FILES}}
MANIFEST_S = mt -manifest ${MQTTDLL_S}.bildirge -outputresource: ${MQTTDLL_S}\; 2
MQTTLIB_A = mqttv3a
MQTTDL_A = windows_ia32/${MQTTLIB_A} .dll
KAYNAK_DOSYA_A = ${ süzgeç-out ${MQTTCLIENT_DIR}/MQTTClient.c ${MQTTCLIENT_DIR}/SSLSocket.c, $
{ALL_SOURCE_FILES}}
MANIFEST_S = mt -manifest ${MQTTDLL_S}.bildirge -outputresource: ${MQTTDLL_S}\; 2
MQTTLIB_AS = mqttv3as
MQTTDLL_AS = windows_ia32/${MQTTLIB_AS} .dll
KAYNAK_DOSYA_AS = ${ süzgeç-çıkış ${MQTTCLIENT_DIR}/MQTTClient.c, ${ALL_SOURCE_FILES}}
MANIFEST_S = mt -manifest ${MQTTDLL_S}.bildirge -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 = bağlantı
LINKFLAGS = /nologo /machine:x86 /bildirge /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}
LIBPPDB = /pdb: ${@:.dll=.pdb}
LIBMAP = /map: ${@:.dll=.map}
WINLIBS_S = ${WINLIBS} crypt32.lib ssleay32.lib libeay32.lib
LIBPATH_S = /LIBPATH: ${OPENSSL_DIR}/lib

# targets
tümü: ${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} *.oj /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} ${LIBPDB} ${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 }

.Temiz.
temizle:
    -rm -f *.obj
    -rm -f -r windows_ia32
```

OpenSSL paketini oluşturma

C, `mqttv3cs` ve `mqttv3as`için güvenli MQTT istemci kitaplıklarını oluşturmadan önce OpenSSL paketini oluşturun. The build creates the libraries that are required to build a secure version of the MQTT client for C library, and the OpenSSL certificate management tool.

Başlamadan önce

1. iOS The iOS makefile customization is for target devices that run iOS6. The customization might be different for earlier or later versions of iOS.
2. Windows Windows makefile uyarlaması, 32 bit pencereler içindir.

Bu görev hakkında

OpenSSL paketini ve herhangi bir önkoşul yazılıma yükleyin ve kurun. Customize the OpenSSL makefiles, and build OpenSSL libraries for your target platform. Windows ve Linuxüzerinde, OpenSSL anahtar oluşturma ve yönetim aracı da oluşturur.

Yordam

1. OpenSSL paketini kurun.
   a) Download the OpenSSL package from OpenSSL

      Önemli: OpenSSL paketinin karşıdan yüklenmesi ve yeniden dağıtılması, katı içe aktarma ve dışa aktarma düzenlemelerine ve açık kaynak lisanslama koşullarına tabidir. Paketi karşıdan yüklemeye karar vermeden önce kısıtlamalara ve uyarılara dikkat edin.

b) Sıkıştırılmış dosya içeriğini *sdkroot*olarak genişletin.

En son paketin karşıdan yükleme konumunu bulmak için OpenSSL sitesindeki **News** (Haberler) sekmesinin altına bakın. Paket, `tar.gz`uzantılı bir tar dosyası olarak sıkıştırılır. Paket genişletildiğinde, en üst düzey klasör (`openssl`*version*) yaratır; örneğin, *openssl-1.0.1c*. The examples refer to the path to the folder as %openssl% on Windows and $openssl on iOS; for example, on Windows, %openssl% is *sdkroot*\openssl-1.0.1c.

İpucu: OpenSSL paketini ayıklayarak yaratılan dizin yolunu denetleyin. Some packages have duplicate levels of the `openssl`*version* folder.

2. Windows

İsteğe bağlı: Windows' ta perl 'i karşıdan yükleyin ve kurun. Bkz. perl.org.

Örnek için, perl ActivePerl Karşıdan Yüklemeler' den aşağı yüklendi.

3. iOS

İsteğe bağlı: iOS' ta üç dizin daha oluşturun.

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

iOS için, üç farklı donanım platformu için OpenSSL paketini oluşturmanız gerekir.

4. Donanımınıza ve işletim sisteminize ilişkin OpenSSL paketini oluşturmak için OpenSSL makedfile 'ı oluşturun.

a) %openssl% ya da $openssl dizininde bir komut penceresi açın.

b) Run the **Configure** perl command with the appropriate parameters.

- 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

iOS' ta, oluşturulan OpenSSL makefile 'ı farklı Apple aygıtları için özelleştirin.

a) Oluşturulan makedo dosyasının üç kopyasını çıkarın, $openssl/Makefile

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

b) Her bir makefile içindeki "CC=gcc" deyimini değiştirin.

CC=gcc deyimi, 62 numaralı satırdır ya da bu civarda. Bu değeri aşağıdaki komutlara çevirin:

$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) Her bir makefile içindeki "CFLAG=..." deyimini değiştirin.

Deyim 63 numaralı hatta ya da civarda (okunabilirliği için üç satıra bölüner):

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

Gösterilen SDK'lerin konumu Xcode kuruluş seçimlerinize bağlıdır. SDD ' lerin sürümü, makefile for. (Makefl) dosyasını oluşturmakta olduğunuz işletim sistemi düzeyine bağlıdır.

iPhone benzetimi

iPhone simulator makefile $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

iOS makefiles, $openssl/Makefile_arm7 ve $openssl/Makefile_arm7s' dir.

```
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. Oluşturulan makefile komutunu çalıştırın.

- Windows

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

- iOS iOS'ta:

```
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
```

Sonuçlar

Oluşturma, C için MQTT istemci kitaplığının güvenli sürümlerini oluşturmak için gerekli paylaşılan kitaplıkları, lib ve üstbilgi dosyalarını oluşturur.

iOSüzerinde C için MQTT istemcisi ile çalışmaya başlama

Learn how to get iOS applications to exchange messages with an MQTT server. iOS aygıtlarında (yani, iPhone ve iPad) kullanım için,Ciçin MQTT istemci kitaplığını, MQTT yazılım geliştirme setinin bir parçası olarak sağlanan kaynak koddan oluşturmanız gerekir.

Başlamadan önce

1. iOS Dev Merkezi ' ne bağlantı oluşturun ve iOSiçin uygulamaların nasıl geliştirileceğini biliniz.

2. Xcode tümleşik geliştirme ortamını (IDE) çalıştırmak için, OS X 10.8.2ile bir Apple Mac edinin ya da daha sonrarunintegrated' u çalıştırın.
3. (İsteğe bağlı) Windows ya da Linuxüzerinde bir C geliştirme ortamı yapılandırın. It is useful to build and run the MQTT client sample C apps on Windows or Linux before you develop an MQTT iOS app. Diğer bir seçenek olarak, örnek kaynak kodunu, örnekleri oluşturmadan inceler.
4. For supported and reference MQTT client platforms for C, see IBM Mobil İleti Sistemi ve M2M Client Packiçin sistem gereksinimleri.
5. İstemciniz ve sunucu arasında bir güvenlik duvarı varsa, bunun MQTT trafiğini engellediğinden emin olun.

Bu görev hakkında

Yordam, aşağıdaki adımlar boyunca size yol gösterir:

1. Learn about programming for MQTT by studying, building, and running the MQTT client sample apps and MQTT client libraries for C.
2. Install the Xcode development environment for iOS on Apple Mac.
3. Do the task “C kitaplıkları için MQTT istemcisi oluşturulması” sayfa 29 to build the MQTT client for C libraries for iOS devices.

Yordam

1. İstemci uygulamasını bağlayabileceğiniz bir MQTT sunucusu seçin.

   Sunucu, MQTT version 3.1 iletişim kuralını desteklemelidir. IBM WebSphere MQ ve IBM MessageSightde içinde olmak üzere, tüm MQTT sunucuları IBM ' dan bunu yapar. Bkz. “MQTT Server sunucuları ile çalışmaya başlama” sayfa 134.
2. Mobil İleti Sistemi ve M2M Client Pack dosyasını yükleyin ve MQTT SDK ' yı kurun.

   Kuruluş programı yok, yalnızca karşıdan yüklenen dosyayı genişletiyorsunuz.

   a. Mobil İleti Sistemi ve M2M Client Pack' ı karşıdan yükleyin.
   b. SDK ' yı kurabildiğiniz bir klasör oluşturun.

      `MQTT`klasörünün adını vermek isteyebilirsiniz. Bu klasöre giden yol, burada *`sdkroot`*olarak adlandırılır.
   c. Sıkıştırılmış Mobil İleti Sistemi ve M2M Client Pack dosyasının içeriğini *`sdkroot`*' e genişletin. Genişletme, *`sdkroot`*\SDK' ta başlayan bir dizin ağacı oluşturur.
3. İsteğe bağlı: Familiarize yourself with the MQTT API by studying the MQTT istemcisi örnek C uygulaması.
   a) Build the synchronous MQTT client sample C app `MQTTV3sample.c` for Windows or Linux. Bkz. “C için MQTT istemcisiyle çalışmaya başlama” sayfa 25.
   b) Bir MQTT version 3 sunucusuna bağlanın ve sunucudaki konuları yayınlayın ve yayınlayın.
   c) Kaynak kodu ve MQTT API belgelerini araştır. MQTT istemci kitaplıklarına ilişkin istemci API belgelerine ilişkin bağlantılar için bkz. MQTT istemci programlama başvurusu.

      Learn how to create and resume MQTT clients, and publish and subscribe to MQTT topics by studying the synchronous sample. Zamanuyumlu örnek, zamanuyumsuz örnekten daha basittir. MQTT 'u daha önce programlamadıysanız, MQTT programlama modelini ve API' yi bilmeniz için zamanuyumlu bir MQTT programı yazınız.
   d) asynchronousüzerinde C için zamanuyumsuz MQTT istemci kitaplığını oluşturun; Windows ya da Linux. Bkz. “C kitaplıkları için MQTT istemcisi oluşturulması” sayfa 29.
   e) Zamanuyumsuz MQTT istemcisi örnek yayınlama ve abone olma C uygulamasını oluşturun ve çalıştırın.
   f) MQTT istemcisi örnek zamanuyumsuz C uygulama kaynak kodunu ve MQTT başvuru belgelerini araştır.

Mobil aygıt için bir MQTT uygulaması yazmak için zamanuyumsuz arabirimi kullanmanız gerekir. Zamanuyumsuz arabirime hitap eden yazılı uygulamalar, zamanuyumlu arabirime yazılan uygulamalardan daha hızlı yanıt verebilir ve pil ömrünü uzatabilr.

Zamanuyumsuz arabirimin zamanuyumsuz olarak iki derece zamanuyumsuz olması gerekir:

i) Birinci derece, MQTT istemci kitaplığı sunucudan yayın almayı beklerken uygulamanın engelini kaldırmak olur.

ii) İkinci derece, istemci kitaplığı sunucuya bağlıyken, abonelikler yaratarak ve yayınlar gönderirken uygulamanın engelini kaldırmak gerekir.

4. iOS geliştirme araçlarını karşıdan yükleyin ve kurun.
   a. Yönetici ayrıcalıklarına sahip bir kullanıcı kimliğiyle oturum açın.
   b. Apple Mac ürününüzün 10.8.2 sürümünde ya da sonraki sürümlerinden olup olmadığını denetleyin.
   c. Go to the website Xcode to download Xcode from the Mac app store.
   d. Xcodekomutunu, komut satırı ortamını ve benzetimcisini kurun.

   Mac app store, benzetimcisinin birden çok sürümünü sunarsa, uygulamanız için hedeflediğiniz iOS düzeyiyle uyumlu sürümü seçin.
5. iOSüzerinde C için MQTT istemci kitaplıklarını oluşturun. Bkz. “C kitaplıkları için MQTT istemcisi oluşturulması” sayfa 29.

Sonraki adım

1. Geliştirdiğiniz C için MQTT istemci kitaplıklarını doğrulayın:
   a. Zamanuyumsuz MQTT istemcisi örnek C uygulaması' i derlemek ve C için güvenli olmayan zamanuyumsuz MQTT istemci kitaplığına bağlantı oluşturmak için Xcode geliştirme ortamını kullanın.
   b. Bir iOS aygıtında zamanuyumsuz MQTT istemci örnek C uygulamasını çalıştırmak için Xcode geliştirme ortamını kullanın. Örneği yapılandırdığınız MQTT version 3 Server sunucusuna bağlayın; bkz. “Configuring the MQTT service from the command line” sayfa 138.
2. iOSiçin bir MQTT istemcisi C uygulaması oluşturun. Zamanuyumsuz C uygulamasına ilişkin kodlama örnekleri yararlı olabilir. Örnekler, *sdkroot*`\SDK\clients\c\samples`içinde `MQTTV3ASample.c` ve `MQTTV3ASSample.c` ' dir. Bir alıştırma olarak, MQTT publish/subscreen örneğini uygulayarak başlatın.

   İpucu: To get an idea of what the app might look like and do, look at screen captures of the MQTT istemcisi örnek C uygulaması. Bkz. “Android üzerinde Javaiçin MQTT istemcisi ile çalışmaya başlama” sayfa 17.

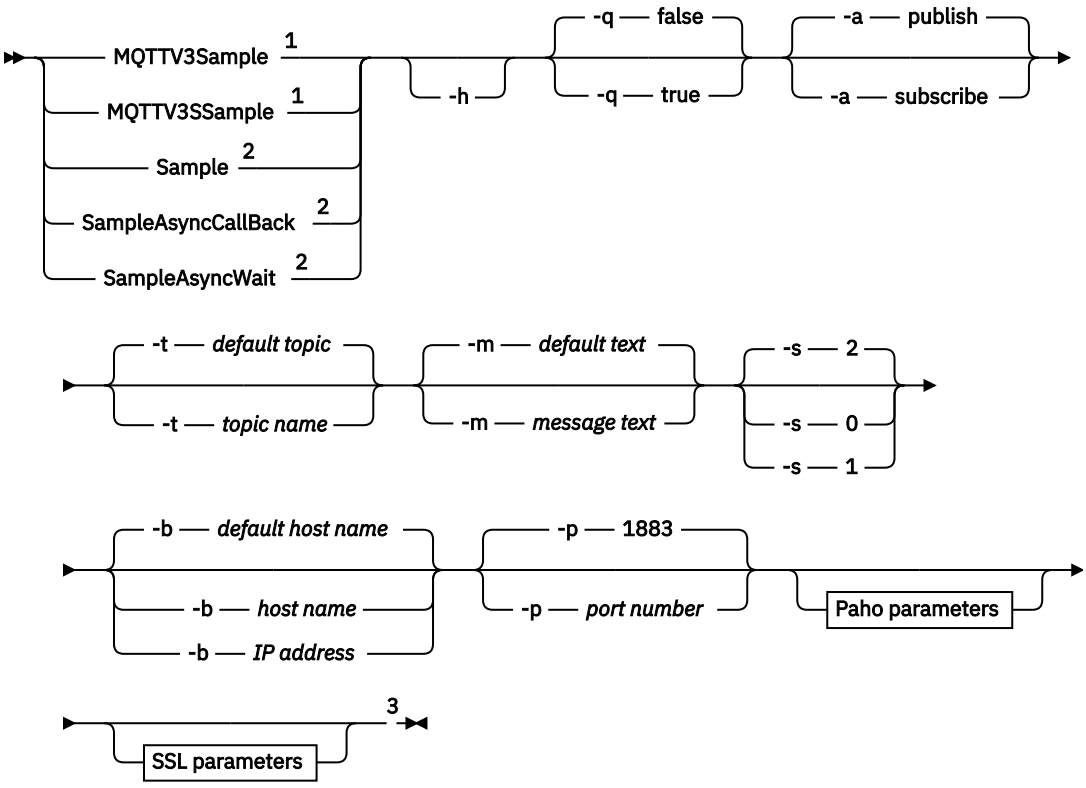
MQTT komut satırı örnek programları

MQTT komut satırı örnek programlarının sözdizimi ve değiştirgeleriyle ilgili bilgiler.

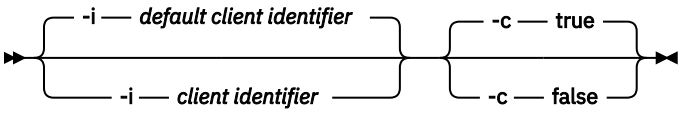
Amaç

Bir konuyu yayınlayın ve bir konuya abone olun.

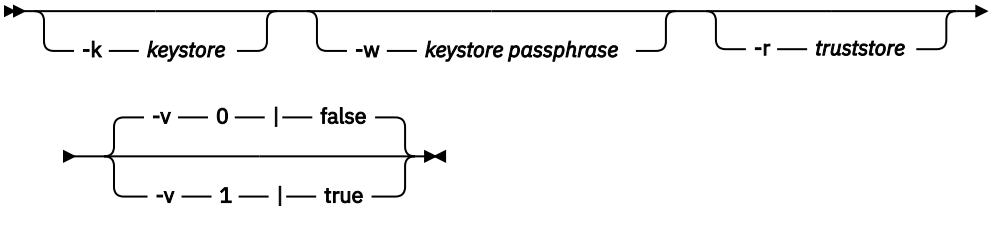
Syntax



Paho parameters



SSL parameters



Notlar:

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

Parametreler

`-h`
Bu yardım metnini yazdır ve çık

`-q`
False (yanlış) kipinin varsayılan kipini kullanmak yerine sessiz kipi ayarlayın.

`-a publish|subscribe`
Varsayılan yayınlama işleminin varsayılması yerine, işlemi `publish` ya da `subscribe`olarak ayarlayın.

-t ***konu adı***
Varsayılan konuya abone olmak ya da abone olmak yerine *topic name*' e abone olun ya da bu konuya abone olun. Varsayılan konular şunlardır:

Paho örnekleri

Yayınla
Sample/Java/v3

Abone Ol
Sample/#

IBM WebSphere MQ örnekleri

Yayınla
MQTTV3Sample/Java/v3 ya da MQTTV3Sample/C/v3

Abone Ol
MQTTV3Sample/#

-m ***ileti metni***
Varsayılan metni göndermek yerine *message text* yayınını yayınlayın. Varsayılan metin, "Message from MQTTv3 C client"ya da "Message from MQTTv3 Java client"' dir.

-s 0|1|2
Varsayılan QoS, 2kullanmak yerine hizmet kalitesini (QoS) ayarlayın.

-b ***anasistem adı***
Varsayılan ana makine adına bağlanmak yerine *host name* ya da IP adresine bağlanın. Paho örnekleri için varsayılan anasistem adı m2m.eclipse.org' dir. IBM WebSphere MQ örnekleri için bu localhost' dir.

-p ***kapı numarası***
Varsayılan kapıyı (1883) kullanmak yerine *port number* kapısını kullanın.

Paho parametreleri

-i ***istemci tanıtıcısı***
İstemci tanıtıcısını *client identifier.* olarak ayarlayın. Varsayılan istemci tanıtıcısı: SampleJavaV3_"+action; burada action , publish ya da subscribe' dir.

-c true|false
Temizleme oturumu işaretini ayarlayın. Varsayılan değer şudur: true: abonelikler dayanıklı değildir.

SSL parametreleri

-k ***anahtar deposu***
Yolu, istemciyi *keystore*' e tanıtan özel anahtarı içeren anahtar deposunun yolunu ayarlayın. C örnekleri için mağaza Privacy-Enhanced Mail (PEM) dosyasıdır. Java örnekleri için bu bir Java anahtar deposıdır (JKS).

-w ***anahtar deposu geçiş tümcesi***
Parolayı, istemciyi anahtar deposuna *keystore passphrase*' e erişecek şekilde yetkilendirecek şekilde ayarlayın.

-r ***güvenilirlik deposu***
İstemcinin *truststore*' a güvendiği MQTT sunucularının ortak anahtarlarını içeren anahtar deposunun yolunu ayarlayın. Anahtar deposu Privacy-Enhanced Mail (PEM) dosyasıdır. C örnekleri için mağaza Privacy-Enhanced Mail (PEM) dosyasıdır. Java örnekleri için bu bir Java anahtar deposıdır (JKS).

-v 0|false|1|true
Doğrulama seçeneğini 1|true olarak ayarlamak için sunucu sertifikası gerekir. Varsayılan değer 0| false' dir: Sunucu sertifikası denetlenmez. SSL kanalı her zaman şifrelenir.

C programları için 0|1 , Java programları için true|false seçeneğini ayarlayın.

İlgili görevler

“Javaiçin MQTT istemcisiyle çalışmaya başlama” sayfa 11
' den bir istemci kitaplığı kullanır ve either IBM MessageSight or IBM WebSphere MQ as the MQTT serveristemcisini kullanarak Java örnek uygulamaları için MQTT Client istemcisiyle çalışır. Örnek uygulamalar, MQTT yazılım geliştirme araç takımından (SDK) IBM' den istemci kitaplığı kullanır. `SampleAsyncCallBack` örnek uygulaması, Android uygulamaları ve olay odaklı diğer işletim sistemleri için MQTT uygulamaları yazmak için kullanılan bir modeldir.

“C için MQTT istemcisiyle çalışmaya başlama” sayfa 25
C kaynağını derleyebileceğiniz herhangi bir platform üzerinde C için örnek MQTT istemcisiyle çalışır ve çalışır. Örnek MQTT Client for C ' yi either IBM MessageSight or IBM WebSphere MQ as the MQTT serverile çalıştırabildiğinizi doğrulayın.

“C kitaplıkları için MQTT istemcisi oluşturulması” sayfa 29
C kitaplıkları için MQTT istemcisini oluşturmak için bu adımları izleyin. Bu konuda, bir dizi platformun derleme ve bağlantı anahtarları ve iOS ve Windowsüzerindeki kitaplıkların oluşturulmasına ilişkin örnekler yer alır.

MQTT güvenliği

Üç kavram, MQTT güvenliği için temel olur: kimlik, kimlik doğrulama ve yetkilendirme. Kimlik, yetkilendirilmekte olan ve yetki verilen istemciyi adlandırma üzeredir. Kimlik doğrulaması, istemcinin kimliğini kanıtlamayla ilgilidir ve yetkilendirme, istemciye verilen hakların yönetilmesine ilişkin olarak ortaya çıktı.

Güvenlik örneklerini deneyin

- “Güvenli MQTT istemcisi örnek Java uygulaması' in oluşturulması ve çalıştırılması” sayfa 53
- “Connecting the MQTT client sample Java app on Android over SSL” sayfa 62
- “JAASile birlikte bir MQTT istemcisi Java uygulamasının kimlik doğrulaması” sayfa 71
- V 7.5.0.1 “MQTT messaging client for JavaScript ile SSL ve WebSocketsarasındaki bağlantı kurulması” sayfa 76
- “Güvenli MQTT istemcisi örnek C uygulaması' in oluşturulması ve çalıştırılması” sayfa 84

Özdeşlik

Bir MQTT istemcisini istemci tanıtıcısı, kullanıcı kimliği ya da genel sayısal sertifika ile tanıtın. Bu özniteliklerden biri ya da biri istemci kimliğini tanımlar. Bir MQTT sunucusu, istemci tarafından gönderilen sertifikayı SSL protokolüyle ya da istemci tarafından ayarlanmış bir parolayla istemci kimliğine doğrular. İstemci, istemci kimliğine dayalı olarak istemcinin erişebileceği kaynakları denetler.

MQTT sunucusu kendisini, IP adresi ve sayısal sertifikasıyla istemciye tanıtır. MQTT istemcisi, sunucu tarafından gönderilen sertifikenin kimliğini doğrulamak için SSL protokolünü kullanır. Bazı durumlarda, sertifikana gönderen sunucunun sertifika sahibi olarak kaydedildiğini doğrulamak için sunucunun DNS adını kullanır.

İstemcinin kimliğini aşağıdaki yöntemlerden birini kullanarak ayarlayın:

İstemci tanıtıcısı
`MqttClient` sınıfı (C içindeki`MQTTClient_create` ya da `MQTTAsync_create`) istemci tanıtıcısını ayarlar. İstemci tanıtıcısını parametre olarak ayarlamak için sınıf oluşturucusunu çağırın ya da rasgele oluşturulmuş bir istemci tanıtıcısını döndürün. İstemci tanıtıcısı, sunucuya bağlanan tüm istemcilerde benzersiz olmalıdır ve sunucuda kuyruk yöneticisi adıyla aynı olmamalıdır. Kimlik denetimi için kullanılmamış olsa da, tüm istemcilerin bir istemci tanıtıcısı olmalıdır. Bkz. “İstemci tanıtıcısı” sayfa 126.

Kullanıcı kimliği
`MqttClient` sınıfı (C içindeki`MQTTClient_create` ya da `MQTTAsync_create`) istemci kullanıcı kimliğini `MqttConnectOptions` özniteliği (C ' de`MqttClient_ConnectOptions`) olarak ayarlar. Kullanıcı kimliğinin bir istemciye benzersiz olması gerekmez.

Müşteri dijital sertifikası
İstemci sayısal sertifikası istemci anahtar deposunda depolanır. Anahtar deposu yeri istemciye bağlıdır:

- **Java**

  Set the location and properties of the client keystore by calling the `setSSLProperties` method of `MqttConnectOptions` and passing the keystore properties. Bkz. SSL Nitelikleri Example.java. **`keytool`** aracı, Java anahtarlarını ve anahtar depolarını yönetir.

- **C**

  `MQTTClient_create` ya da `MQTTAsync_create` , anahtar deposu özelliklerini `MQTTClient_SSLOptions ssl_opts`özniteliği olarak ayarlar. **openSSL** aracı, C için MQTT istemcisinin eriştiği anahtarları ve anahtar depolarını yaratır ve yönetir.

- **Android**

  Ayarlar > **Güvenlik** menüsünden bir Android aygıt anahtar deposunu yönetin. SD kartından yeni sertifikaları yükleyin.

Sunucu anahtar deposunda özel anahtarını depolayarak sunucunun kimliğini ayarlayın:

IBM WebSphere MQ
MQTT sunucusu anahtar deposu, istemcinin bağlandığı telemetri kanalının bir öznitesidir.

Anahtar deposu konumunu ve özniteliklerini IBM WebSphere MQ Explorerile ya da **`DEFINE CHANNEL`** komutuyla ayarlayın; bkz. DEFINE CHANNEL (MQTT). Bir anahtar deposunu birden çok kanal paylaşabilir.

Kimlik denetimi

Bir MQTT istemcisi, bağlandığı MQTT sunucusunun kimliğini doğrulayabilir ve sunucu, bağlantı kuran istemcinin kimliğini doğrulayabilir.

İstemci, SSL protokolüyle bir sunucuyu doğrular. MQTT sunucusu, bir istemciyi SSL protokolüyle ya da parolayla ya da her ikisiyle doğrulayan bir istemci oluşturur.

İstemci sunucuyu doğruluyorsa, ancak sunucu istemcinin kimliğini doğrulamazsa, istemci genellikle anonim bir istemci olarak bilinir. SSL üzerinden anonim bir istemci bağlantısı oluşturmak ve sonra SSL oturumu tarafından şifrelenen bir parolayla istemcinin kimliğini doğrulamak yaygın bir durum. Sertifika dağıtım ve yönetim sorunu nedeniyle, istemci sertifikasına sahip bir istemcinin kimliğini doğrulamak için çok daha yaygın bir durum söz edilir. Büyük olasılıkla ATM ' ler ve çip-pim makineleri gibi yüksek değerli aygıtlarda kullanılan ve akıllı elektrik sayaçları gibi özel aygıtlarda kullanılan istemci sertifikalarını bulabilirsiniz.

İstemci tarafından sunucu kimlik doğrulaması
Bir MQTT istemcisi, SSL protokolüyle sunucu sertifikasının doğrulanarak doğru sunucuya bağlı olduğunu doğrular. HTTPS iletişim kuralı üzerinden bir web sitesine göz attığınızda, bu doğrulama formu size tanıdık geliyor.

Sunucu, sertifika yetkilisi tarafından imzalanmış olan genel sertifikasını istemciye gönderir. İstemci, sunucu sertifikasında sertifika yetkilisinin imzasını doğrulamak için sertifika yetkilisinin genel anahtarını kullanır. Ayrıca, sertifikanda geçerli olup olmadığını da denetler. Bu denetimler, sertifikenin geçerli olduğunu belirtir.

Sertifika yetkilisi sertifikaları, genellikle kök sertifikaları termed kök sertifikaları istemci güvenilirlik deposunda saklanır:

- **Java**

Call the `setSSLProperties` method of `MqttConnectOptions` and pass the truststore properties to set the location and properties of the client truststore. Bkz. SSL Nitelikleri Example.java. Manage certificates and truststores with the **keytool** tool.

- **C**

  `MQTTClient_create` ya da `MQTTAsync_create` güvenli depo özelliklerini `MQTTClient_SSLOptions ssl_opts`özniteliği olarak ayarlar. Manage certificates and truststores with the **openSSL** tool.

- **Android**

  Ayarlar > **Güvenlik** menüsünden bir Android aygıt güvenli deposunu yönetin. SD kartından yeni kök sertifikalarını yükleyin.

Sunucu tarafından istemci kimlik doğrulaması

Bir MQTT sunucusu, SSL protokolüyle istemci sertifikasını doğrulayarak ya da bir parolayla istemci kimliği doğrulanarak doğru istemciye bağlı olduğunu doğrular.

İstemcinin kimliğini, istemci tarafından bir MQTT protocol üstbilgisinde sunucuya gönderilen parolayla doğrular. Sunucu, parola ile istemci tanıtıcısının, kullanıcı kimliğinin ya da sertifikının kimliğini doğrulamayı tercih edebilir. Bu, sunucuya bağlıdır. Genellikle sunucu, kullanıcı kimliğinin kimliğini doğrular. Parolaları açık bir şekilde göndermek için sunucu doğrulanarak güvenli kılınan bir SSL bağlantısı üzerinden parolaları doğrulayın.

- **IBM WebSphere MQ**

  IBM WebSphere MQ , SSL protokolüyle istemci sertifikasının kimliğini doğrular. Kök sertifikalarını IBM WebSphere MQ Telemetry anahtar depounda saklayın. Bir istemci sertifikasının kimliğini yalnızca karşılıklı SSL kimlik doğrulamasının bir parçası olarak doğrulayabilirsiniz. Yani, istemciyi sunucu genel sertifikasıyla birlikte sağlamanın yanı sıra, istemci genel sertifikasına sahip bir istemci de sağlamalısınız.

  IBM WebSphere MQ Telemetry , hem kendi özel hem de genel sertifikası için aynı mağazeyi ve sertifika yetkililerinin sağladığı kök sertifikalar gibi diğer genel sertifikalar için de kullanılır.

  Anahtar deposu konumunu ve özniteliklerini IBM WebSphere MQ Explorerile ya da **DEFINE CHANNEL** komutuyla ayarlayın; bkz. DEFINE CHANNEL (MQTT). Bir anahtar deposunu birden çok kanal paylaşabilir.

  IBM WebSphere MQ authenticates the client user ID, or the client identifier, by calling the Java authentication and authorization service (JAAS).

  JAAS dosyasını, `jaas.config` dosyasında saklanan bir `MQXRConfig` yapılandırma stanza içinde yapılandırın. Dosya, IBM WebSphere MQ veri yolundaki `qmgrs\`*`QmgrName`*`\mqxr` dizininde depolanır.

  `JAASLoginModule`için bir `login` yöntemi yazarak istemcinin güvenilirliğini denetleyin. Bkz. “Telemetri kanalı JAAS yapılandırması” sayfa 113.

  IBM WebSphere MQ Telemetry , `JAASLoginModule.login` yöntemini aşağıdaki değiştirgelerden geçirir:

  – Kullanıcı kimliği
  – Parola
  – İstemci tanıtıcısı
  – Ağ tanıtıcısı
  – Kanal adı
  – ValidPrompts

Yetkilendirme

Yetkilendirme, MQTT protocol' nin bir parçası değil. Bu, MQTT sunucuları tarafından sağlanır. Yetkili olan sunucunun ne yaptığıya bağlıdır. MQTT sunucuları, yayınlama/abone olma aracılarıdır ve yararlı MQTT yetkilendirme kuralları, hangi istemcilerin sunucuya bağlanabileceğini ve hangi konuların bir istemcinin yayınlayabileceğini ya da abone olabileceğini denetler. Bir MQTT istemcisi sunucuyu yönetirse, hangi istemcilerin sunucunun farklı yönlerini denetleyebileceğini daha fazla yetki kuralları denetimi sağlar.

Olası istemcilerin sayısı çok büyük olduğundan, her istemciye ayrı ayrı yetki vermek mümkün değildir. Bir MQTT sunucusunun, istemcilerin profillere ya da gruplara göre gruplanacak bir aracı olması gerekir.

Bir istemcinin kimliği, erişim ve yetkilendirme açısından, bir MQTT istemcisi için benzersiz bir şey değildir. İstemci tanıtıcısı ile, bir istemcinin kimliğini eşitlemeyin. Onlar da aynı olabilir, ama genellikle farklı. Örneğin, bir dizi hizmette ortak olan bir kullanıcı adınız ve bu hizmetlerden bazıları "tekli oturum açma"alanında işbirliği içinde olabilir. Kurumsal ölçekte MQTT sunucusu, farklı uygulamalar için ortak kimlikler ve yetkiler sunan bir yetkilendirme hizmetini çağıracaktır.

IBM WebSphere MQ

IBM WebSphere MQ , takılabilir bir yetki hizmetine sahiptir. Windows ve Linux üzerinde sağlanan varsayılan yetkilendirme hizmeti, nesne yetkilisi yöneticidir (OAM). Bkz. UNIX, Linux ve Windows sistemlerinde OAM kullanarak nesnelere erişim denetleme. İşletim sistemi kullanıcı kimliklerini ve gruplarını, konular ve kuyruklar gibi IBM WebSphere MQ nesneleriyle ilgili işlemler ile ilişkilendirir.

Bir telemetri kanalını, sabit bir kullanıcı kimliğiyle IBM WebSphere MQ ' e erişecek şekilde yapılandırabilirsiniz. Bu, örnek kanalının nasıl ayarlanandır. Ya da IBM WebSphere MQ olanağına MQTT istemcisi tarafından ayarlanan kullanıcı kimliği ile erişebilirsiniz. WebSphere MQ nesnelerine erişmek için MQTT istemcilerinin yetkilendirmesi , kaba, orta ve para cezasına çarptırılan istemci erişim denetime ulaşmak için IBM WebSphere MQ Telemetry ' un ayarlanmasını açıklar.

İlgili görevler

“Güvenli MQTT istemcisi örnek C uygulaması' in oluşturulması ve çalıştırılması” sayfa 84
Bir Windows örneğine dayalı olarak, C kaynağını derleyebileceğiniz herhangi bir işletim sisteminde güvenli örnek C uygulamasıyla çalışmaya devam edebilir ve bu uygulamayı çalıştırabilirsiniz. Örnek C uygulamasını either IBM MessageSight or IBM WebSphere MQ as the MQTT server' ta çalıştırabildiğinizi doğrulayın.

“Güvenli MQTT istemcisi örnek Java uygulaması' in oluşturulması ve çalıştırılması” sayfa 53
Bir Windows örneğine dayalı olarak, either IBM MessageSight or IBM WebSphere MQ as the MQTT serverüzerindeki güvenli örnek Java uygulamasıyla çalışmaya devam edebilir ve bu uygulamayı çalıştırabilirsiniz. You can run an Javaiçin MQTT istemcisi app on any platform with JSE 1.5 or above that is "Java Uyumlu"

“MQTT messaging client for JavaScript ile SSL ve WebSocketsarasındaki bağlantı kurulması” sayfa 76
Connect your web app securely to IBM WebSphere MQ by using the MQTT messaging client for JavaScript sample HTML pages with SSL and the WebSocket protocol.

“Connecting the MQTT client sample Java app on Android over SSL” sayfa 62
Get up and running with the sample Android MQTT client connected to IBM WebSphere MQ over SSL.

“JAASile birlikte bir MQTT istemcisi Java uygulamasının kimlik doğrulaması” sayfa 71
Learn how to authenticate a client with 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.

Güvenli MQTT istemcisi örnek Java uygulaması' in oluşturulması ve çalıştırılması

Bir Windows örneğine dayalı olarak, either IBM MessageSight or IBM WebSphere MQ as the MQTT serverüzerindeki güvenli örnek Java uygulamasıyla çalışmaya devam edebilir ve bu uygulamayı çalıştırabilirsiniz. You can run an Javaiçin MQTT istemcisi app on any platform with JSE 1.5 or above that is "Java Uyumlu"

Başlamadan önce

1. SSL üzerinden MQTT protocol ' yi destekleyen bir MQTT version 3.1 sunucusuna erişiminiz olmalıdır.
2. İstemciniz ve sunucu arasında bir güvenlik duvarı varsa, bunun MQTT trafiğini engellediğinden emin olun.
3. You can run an Javaiçin MQTT istemcisi app on any platform with JSE 1.5 or above that is "Java Uyumlu". Bkz. IBM Mobil İleti Sistemi ve M2M Client Packiçin sistem gereksinimleri.
4. SSL kanalları başlatılmalıdır.

Bu görev hakkında

As an illustration, this article shows you how to compile and run the secure MQTT istemcisi örnek Java uygulaması on Windows from the command line.

SSL kanalını, sertifika yetkilisi imzalı anahtarlarla ya da kendinden onaylı anahtarlarla sabitleyin.

Yordam

1. İstemci uygulamasını bağlayabileceğiniz bir MQTT sunucusu seçin.

   Sunucu, SSL üzerinden MQTT version 3.1 iletişim kuralını desteklemelidir. IBM WebSphere MQ ve IBM MessageSightde içinde olmak üzere, tüm MQTT sunucuları IBM ' dan bunu yapar. Bkz. “MQTT Server sunucuları ile çalışmaya başlama” sayfa 134.
2. İsteğe bağlı: Sürüm 7 ya da sonraki bir yayın düzeyinde bir Java geliştirme seti (JDK) kurun.

   Sertifikaları onaylamak için **keytool** komutunu çalıştırmak için sürüm 7 gereklidir. Sertifikaları onaylayamayacaksa, Sürüm 7 JDK ' yi zorunlu kılmayın.
3. Mobil İleti Sistemi ve M2M Client Pack dosyasını yükleyin ve MQTT SDK ' yı kurun.

   Kuruluş programı yok, yalnızca karşıdan yüklenen dosyayı genişletiyorsunuz.

   a. Mobil İleti Sistemi ve M2M Client Pack' ı karşıdan yükleyin.

   b. SDK ' yı kurabildiğiniz bir klasör oluşturun.

      MQTTklasörünün adını vermek isteyebilirsiniz. Bu klasöre giden yol, burada *sdkroot*olarak adlandırılır.

   c. Sıkıştırılmış Mobil İleti Sistemi ve M2M Client Pack dosyasının içeriğini *sdkroot*' e genişletin. Genişletme, *sdkroot*\SDK' ta başlayan bir dizin ağacı oluşturur.
4. Anahtar çiftleri ve sertifikalar oluşturmak için komut dosyalarını oluşturun ve çalıştırın ve IBM WebSphere MQ ' ı MQTT sunucusu olarak yapılandırın.

   Komut dosyalarını oluşturmak ve çalıştırmak için “Anahtarlar ve sertifikaların oluşturulması” sayfa 94 içindeki adımları izleyin. Komut dosyaları da “Windowsiçin SSL sertifikalarını yapılandırmak için komut dosyaları örneği” sayfa 56içinde listelenir.
5. SSL kanallarının çalıştığını ve beklediğiniz şekilde ayarlandığınızı denetleyin.

   IBM WebSphere MQ' ta, komut penceresine aşağıdaki komutu yazın:

   - 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. Güvenli MQTT istemcisi örnek Java uygulaması' i oluşturmak ve çalıştırmak için komut dosyalarını oluşturun.

a) Kendinden onaylı sertifikalarla güvenli bir SSL kanalını test etmek için `ssjavaclient.bat` oluşturun ve çalıştırın.

b) Sertifika yetkilisi imzalı sertifikalarla güvenli bir SSL kanalını test etmek için `cajavaclient.bat` oluşturun ve çalıştırın.

MQTT güvenli Java istemcisini çalıştırmak için komut dosyaları

Run the scripts in "Windowsiçin SSL sertifikalarını yapılandırmak için komut dosyaları örneği" sayfa 56 before you run these scripts.

MQTT güvenli Java istemcisi, kendinden onaylı sertifikalar sağlar.

Bu komut dosyasını, `sscerts.bat` komut dosyasını çalıştırarak oluşturduğunuz kendinden onaylı sertifikalar ile çalıştırın.

```
@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
```

Şekil 14. `ssjavaclient.bat`

Run the MQTT secure Java client with certificate authority signed certificates.

Bu komut dosyasını, `cacerts.bat` komut dosyasını çalıştırarak oluşturduğunuz sertifika yetkilisi tarafından imzalanmış sertifikalarla çalıştırın.

```
@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
```

Şekil 15. `cajavaclient.bat`

İlgili kavramlar

“MQTT güvenliği” sayfa 50
Üç kavram, MQTT güvenliği için temel olur: kimlik, kimlik doğrulama ve yetkilendirme. Kimlik, yetkilendirilmekte olan ve yetki verilen istemciyi adlandırma üzeredir. Kimlik doğrulaması, istemcinin kimliğini kanıtlamayla ilgilidir ve yetkilendirme, istemciye verilen hakların yönetilmesine ilişkin olarak ortaya çıktı.

İlgili görevler

“Anahtarlar ve sertifikaların oluşturulması” sayfa 94
Follow this procedure to generate keys and certificates for Java and C clients, including Android and iOS apps, and the IBM WebSphere MQ and IBM MessageSight servers.

“Connecting the MQTT client sample Java app on Android over SSL” sayfa 62
Get up and running with the sample Android MQTT client connected to IBM WebSphere MQ over SSL.

“JAASile birlikte bir MQTT istemcisi Java uygulamasının kimlik doğrulaması” sayfa 71
Learn how to authenticate a client with 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.

Windowsiçin SSL sertifikalarını yapılandırmak için komut dosyaları örneği

** **

Örnek komut dosyaları, sertifikalar ve sertifika depolarını, görevdeki adımlarda açıklandığı gibi oluşturur. Buna ek olarak, örnek, sunucu sertifika deposunu kullanmak için MQTT istemci kuyruk yöneticisini ayarlar. Bu örnek, IBM WebSphere MQile birlikte verilen `SampleMQM.bat` komut dosyasını çağırarak kuyruk yöneticisini siler ve yeniden yaratır.

`initcert.bat`

`initcert.bat` , sertifikaların adlarını ve yollarını ve **keytool** ve **openSSL** komutları için gerekli olan diğer parametreleri ayarlar. Ayarlar, komut dosyasındaki açıklamalarda açıklanır.

```
@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

`cleancert.bat` komut kütüğündeki komutlar, sunucu sertifika deposunun kilitlenmediğinden emin olmak için MQTT istemci kuyruk yöneticisini siler ve örnek güvenlik komut dosyaları tarafından yaratılan tüm anahtar depolarını ve sertifikaları siler.

```
@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

genkeys.bat komut kütüğündeki komutlar, özel sertifika yetkiliniz, sunucu ve bir istemci için anahtar çiftleri yaratır.

```
@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

sscerts.bat komut dosyasındaki komutlar istemci ve sunucu kendinden onaylı sertifikalarını anahtar depolarından dışa aktarır ve sunucu sertifikasını istemci güvenilirlik deposuna ve istemci sertifikasını sunucu anahtar deposuna (içe) aktarır. Sunucunun güvenilir deposu yok. Komutlar, istemci JKS güvenilirlik deposundan PEM biçiminde bir istemci güvenilirlik deposu yaratır.

```
@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

Komut dosyası, sertifika yetkilisi kök sertifikasını özel anahtar depolarına aktarır. Kök sertifika ile imzalanmış sertifika arasında anahtar zinciri oluşturmak için CA kök sertifikasına gerek vardır. `cacerts.bat` komut dosyası, istemciyi ve sunucu sertifikası isteklerini anahtar depolarından dışa aktarır. Komut dosyası, `cajkskeystore.jks` anahtar deposunda özel sertifika yetkilisinin anahtarı ile sertifika isteklerini imzalar ve imzalı sertifikalar, istek geldiği anahtar depolarına geri aktarılır. İçe aktarma, sertifika zincirini CA kök sertifikasıyla yaratır. Komut dosyası, istemci JKS güvenli deposundan PEM biçiminde bir istemci güvenilirlik deposu yaratır.

```
@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

Komut dosyası, sertifika dizinindeki anahtar depolarını ve sertifikalarını listeler. Daha sonra MQTT örnek kuyruk yöneticisini yaratır ve güvenli telemetri kanallarını yapılandırır.

```
@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
```

Connecting the MQTT client sample Java app on Android over SSL

Get up and running with the sample Android MQTT client connected to IBM WebSphere MQ over SSL.

Başlamadan önce

Bu makalede, en az Android API düzeyi 14 (ICS 4.0) çalıştırılıyorsanız varsayılır. Önceki düzeylerde bir anahtar deposu vardı, ancak yalnızca sistem uygulamaları buna erişebilirdi.

1. SSL üzerinden MQTT protocol ' yi destekleyen bir MQTT version 3.1 sunucusuna erişiminiz olmalıdır.
2. İstemciniz ve sunucu arasında bir güvenlik duvarı varsa, bunun MQTT trafiğini engellediğinden emin olun.
3. Bir Android aygıtındaki bağlantıyı sınaysanız, sertifikayı aygıta aktarmak için bir SD kartına gerek duyabilirsiniz.
4. Sanal bir Android aygıtında bağlantıyı sınaysanız, sanal aygıt için bir SD kartı yapılandırın.
5. SSL kanalları başlatılmalıdır.

Bu görev hakkında

Androidiçin MQTT istemcisi örnek Java uygulaması ' ı SSL üzerinden çalıştırmak için bu görevi tamamlayın. Başarılı bir SSL bağlantısı, Android aygıtınız ile MQTT sunucusu arasında güvenli bir şifreli kanal kurar. Sunucunun kimliği doğrulanır.

Androidile, sunucuyu SSL ile doğrulayabilirsiniz. Ayrıca, örnek uygulama bunu desteklemese de, aygıtın kimliğini doğrulayabilirsiniz. Aygıtı doğrulamak için KeyChain API 'sini kullanınya da istemci tanıtıcısını, istemci IP adresini ya da MQTT Android uygulaması tarafından sağlanan kullanıcı adını ve parolayı doğrulamak için JAAS kullanın.

Android güvenilirlik deposuna kurduğunuz tüm X.509 sertifikalarının bir sertifika yetkilisi tarafından imzalanmış olması gerekir. Örnekte, Android aygıtınıza yüklediğiniz sertifikayı imzalayan bir sertifika yetkilisi oluşturursiniz. Çok sayıda kök sertifika, Android aygıtlarına önceden kurulur.

Güvenilir bir sertifikayı kurmadan önce Android aygıtınızda bir kilit oluşturmalısınız. Kilit, bilginiz olmadan birinin aygıtı aygıta kurmasını önler.

Yordam

1. İstemci uygulamasını bağlayabileceğiniz bir MQTT sunucusu seçin.

   Sunucu, SSL üzerinden MQTT version 3.1 iletişim kuralını desteklemelidir. IBM WebSphere MQ ve IBM MessageSightde içinde olmak üzere, tüm MQTT sunucuları IBM ' dan bunu yapar. Bkz. “MQTT Server sunucuları ile çalışmaya başlama” sayfa 134.
2. Run the MQTT client sample app for Android "`MQTTExerciser`" on an unsecured MQTT channel. Bkz. “Android üzerinde Javaiçin MQTT istemcisi ile çalışmaya başlama” sayfa 17.

   Güvenli kanalı test etmek için uygulamayı yeniden kullanıyorsunuz.

   Bir Android sanal aygıtı başlattıysa, çalıştırmaya devam edin.
3. İsteğe bağlı: Sürüm 7 ya da sonraki bir yayın düzeyinde bir Java geliştirme seti (JDK) kurun.

   Sertifikaları onaylamak için **`keytool`** komutunu çalıştırmak için sürüm 7 gereklidir. Sertifikaları onaylayamayacaksa, Sürüm 7 JDK ' yi zorunlu kılmayın.

4. Anahtar çiftleri ve sertifikalar oluşturmak için komut dosyalarını oluşturun ve çalıştırın ve IBM WebSphere MQ ' ı MQTT sunucusu olarak yapılandırın.

   Komut dosyalarını oluşturmak ve çalıştırmak için “Anahtarlar ve sertifikaların oluşturulması” sayfa 94 içindeki adımları izleyin. Komut dosyaları da “Windowsiçin SSL sertifikalarını yapılandırmak için komut dosyaları örneği” sayfa 66içinde listelenir.

   Sertifika yetkilisi genel sertifikasına ve sunucu anahtar deposuna gereksinim duyarsınız. İstemci sertifikalarına ya da `.pem` ya da `.p12` biçimindeki sertifikalara gereksiniminiz yoktur.
5. SSL kanallarının çalıştığını ve beklediğiniz şekilde ayarlandığını denetleyin.

   IBM WebSphere MQ' ta, komut penceresine aşağıdaki komutu yazın:

   - 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. Sertifika yetkilisi sertifikasını Android güvenilirlik deposuna kurun.

   Örnekteki sertifika yetkilisi dosyası `cacert.cer`' dir.

   a) Sertifikayı `cacert.crt`olarak yeniden adlandırın

   b) Sertifikayı kök iç depolamaya ya da SD kartına kopyalayın.

      For a running virtual device, open Eclipse or run the Android Debug Bridge (ADB) to copy the certificate to the virtual device:

      Eclipse

      i) Eclipsekomutunu çalıştırın ve DDMS perspektifini açın.

      ii) Ana görünümde, **Dosya Gezgini** penceresini açın.

      iii) `mnt/sdcard` dizinini açın.

      iv) `cacert.crt` dosyasını `mnt/sdcard` dizinine sürükleyin.

      ADB

      i) Bir komut penceresi açın ve geçerli dizinini, android kuruluş dizininde `android-sdk\platform-tools` olarak ayarlayın; örneğin, `C:\Program Files\Android\android-sdk\platform-tools`.

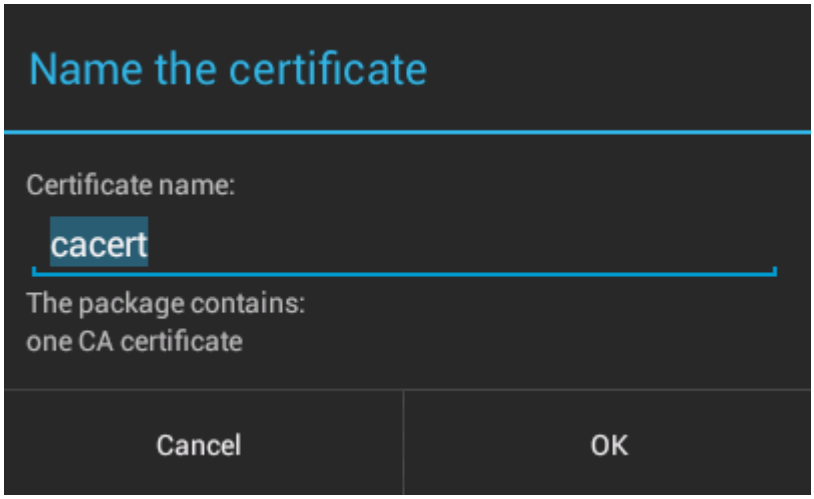
      ii) Sertifikayı `mnt/sdcard` dizinine kopyalayın:

      ```
      adb push %cacert% /mnt/sdcard/cacert.crt
      ```
7. SertifikayıAndroid aygıtındaki sertifika güvenilirlik deposuna kurun.

   Sertifikanda, `Subject Type=CA`değerine sahip bir Temel Kısıtlamalar yantümcesi olmalıdır.

   a) Aygıtın kilidini açın ve **gereçler** düğmesini tıklatın.

   b) **Ayarlar** > **Güvenlik** > **Kimlik Bilgileri Deposu** > **SD karttan kur**seçeneklerini tıklatın.

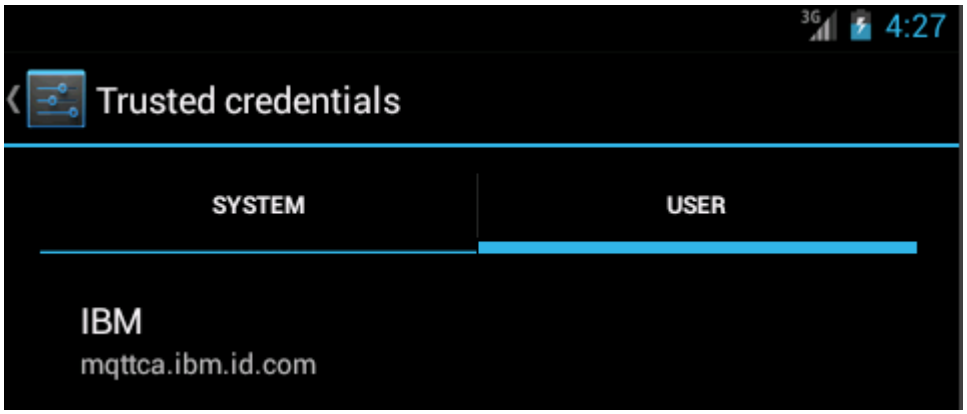


c) Sertifika dosyası adının doğru olduğunu onaylayın ve **Tamam**düğmesini tıklatın.

Not: Aygıt için bir kilit tanımlamadıysanız, şimdi bir kilit ayarlamak için Android sizden bilgi isteminde bulunmanız istenir.

8. Sertifikenin aygıtta kurulu olduğunu doğrulayın.

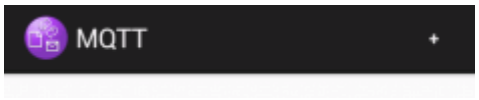
a) **Güvenilen Kimlik Bilgileri** > **Kullanıcı** seçeneklerini tıklatın ve sertifikanız kullanıcı sertifikaları listesinde görüntülenebilmesi için birkaç dakika bekleyin.



9. `MQTTExerciser` uygulamasını yeniden çalıştırın ve anonim SSL istemcileri için yapılandırdığınız bir MQTT kanalına bağlanın.

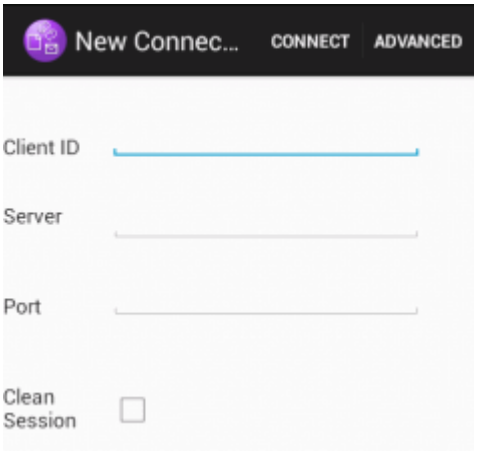
a) Androidiçin MQTT istemcisi örnek Java uygulaması' ı açın.

Bu pencere Android aygıtınızda açık:



b) Bir MQTT sunucusuna bağlanın.

i) Yeni bir MQTT bağlantısı açmak için **+** işaretini tıklatın.



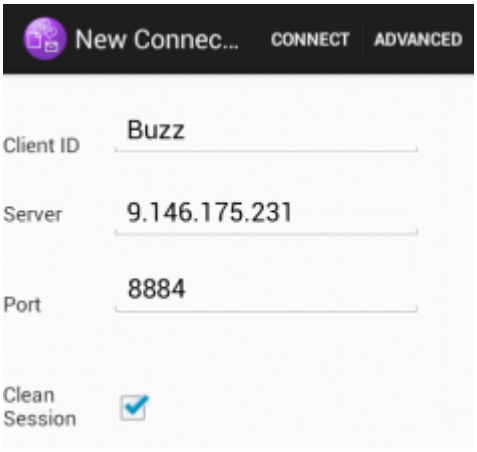
ii) **İstemci tanıtıcısı** alanına benzersiz bir tanıtıcı girin. Sabırlı olun, tuş vuruşu yavaş olabilir.

iii) **Sunucu** alanına, MQTT sunucunuzun IP adresini girin.

Bu, ilk ana adımda seçtiğiniz sunucudur. IP adresi `127.0.0.1`olmamalıdır

iv) MQTT bağlantısına ilişkin kapı numarasını girin.

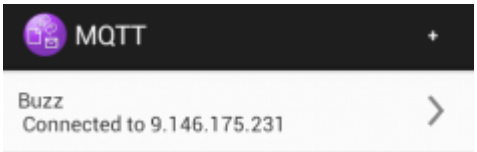
Set the port number to 8884, which is set by the variable `%sslportopt%` in the example scripts. This port number is the port number of the MQTT channel that you have configured for anonymous SSL clients by running the sample scripts in step “4” sayfa 63.



v) **Gelişmiş** sekmesini tıklatın ve **SSL** seçeneğini belirleyin. **Kaydet**'i tıklatın.

vi) **Bağlan**'ı tıklayın.

Bağlantı başarılı olursa, bu pencerenin ardından bir "Bağlanıyor" iletisi görürsünüz:



Sonuçlar

`MQTTExerciser` uygulaması bağlanma ve ileti alışverişi için biraz daha uzun sürer, ancak güvenli olmayan bir bağlantıyla bağlantıda bağlanmaktan başka bir şekilde hareket etmimez.

İlgili kavramlar

“MQTT güvenliği” sayfa 50
Üç kavram, MQTT güvenliği için temel olur: kimlik, kimlik doğrulama ve yetkilendirme. Kimlik, yetkilendirilmekte olan ve yetki verilen istemciyi adlandırma üzeredir. Kimlik doğrulaması, istemcinin

kimliğini kanıtlamayla ilgilidir ve yetkilendirme, istemciye verilen hakların yönetilmesine ilişkin olarak ortaya çıktı.

İlgili görevler

“Anahtarlar ve sertifikaların oluşturulması” sayfa 94
Follow this procedure to generate keys and certificates for Java and C clients, including Android and iOS apps, and the IBM WebSphere MQ and IBM MessageSight servers.

“Güvenli MQTT istemcisi örnek Java uygulaması' in oluşturulması ve çalıştırılması” sayfa 53
Bir Windows örneğine dayalı olarak, either IBM MessageSight or IBM WebSphere MQ as the MQTT serverüzerindeki güvenli örnek Java uygulamasıyla çalışmaya devam edebilir ve bu uygulamayı çalıştırabilirsiniz. You can run an Javaiçin MQTT istemcisi app on any platform with JSE 1.5 or above that is "Java Uyumlu"

“JAASile birlikte bir MQTT istemcisi Java uygulamasının kimlik doğrulaması” sayfa 71
Learn how to authenticate a client with 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.

Windowsiçin SSL sertifikalarını yapılandırmak için komut dosyaları örneği

** **

Örnek komut dosyaları, sertifikalar ve sertifika depolarını, görevdeki adımlarda açıklandığı gibi oluşturur. Buna ek olarak, örnek, sunucu sertifika deposunu kullanmak için MQTT istemci kuyruk yöneticisini ayarlar. Bu örnek, IBM WebSphere MQile birlikte verilen `SampleMQM.bat` komut dosyasını çağırarak kuyruk yöneticisini siler ve yeniden yaratır.

`initcert.bat`

`initcert.bat` , sertifikaların adlarını ve yollarını ve **`keytool`** ve **openSSL** komutları için gerekli olan diğer parametreleri ayarlar. Ayarlar, komut dosyasındaki açıklamalarda açıklanır.

```
@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

cleancert.bat komut kütüğündeki komutlar, sunucu sertifika deposunun kilitlenmediğinden emin olmak için MQTT istemci kuyruk yöneticisini siler ve örnek güvenlik komut dosyaları tarafından yaratılan tüm anahtar depolarını ve sertifikaları siler.

```
@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

genkeys.bat komut kütüğündeki komutlar, özel sertifika yetkiliniz, sunucu ve bir istemci için anahtar çiftleri yaratır.

```
@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

`sscerts.bat` komut dosyasındaki komutlar istemci ve sunucu kendinden onaylı sertifikalarını anahtar depolarından dışa aktarır ve sunucu sertifikasını istemci güvenilirlik deposuna ve istemci sertifikasını sunucu anahtar deposuna (içe) aktarır. Sunucunun güvenilir deposu yok. Komutlar, istemci JKS güvenilirlik deposundan PEM biçiminde bir istemci güvenilirlik deposu yaratır.

```
@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

Komut dosyası, sertifika yetkilisi kök sertifikasını özel anahtar depolarına aktarır. Kök sertifika ile imzalanmış sertifika arasında anahtar zinciri oluşturmak için CA kök sertifikasına gerek vardır. `cacerts.bat` komut dosyası, istemciyi ve sunucu sertifikası isteklerini anahtar depolarından dışa aktarır. Komut dosyası, `cajkskeystore.jks` anahtar deposunda özel sertifika yetkilisinin anahtarı ile sertifika isteklerini imzalar ve imzalı sertifikalar, istek geldiği anahtar depolarına geri aktarılır.

İçe aktarma, sertifika zincirini CA kök sertifikasıyla yaratır. Komut dosyası, istemci JKS güvenli deposundan PEM biçiminde bir istemci güvenilirlik deposu yaratır.

```
@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

Komut dosyası, sertifika dizinindeki anahtar depolarını ve sertifikalarını listeler. Daha sonra MQTT örnek kuyruk yöneticisini yaratır ve güvenli telemetri kanallarını yapılandırır.

```
@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
```

JAASile birlikte bir MQTT istemcisi Java uygulamasının kimlik doğrulaması

Learn how to authenticate a client with 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.

Başlamadan önce

1. You can run an Javaiçin MQTT istemcisi app on any platform with JSE 1.5 or above that is "Java Uyumlu". Bkz. IBM Mobil İleti Sistemi ve M2M Client Packiçin sistem gereksinimleri.
2. İstemciniz ve sunucu arasında bir güvenlik duvarı varsa, bunun MQTT trafiğini engellediğinden emin olun.
3. IBM WebSphere MQ kurulumunda MQXR `JAASLoginModule` ve `JAASPrincipal` Java örneklerine erişiminiz olmalıdır. Örnekler `%MQ_FILE_PATH%\mqxr\samples.`yolunda.
4. Complete the steps on Windows or Linux; the examples are taken from Windows.

5. “1” sayfa 72adımını tamamlamak için, IBM WebSphere MQüzerinde MQXR_SAMPLE_QM kuyruk yöneticisini yaratma yetkisine sahip olmanız gerekir.

Bu görev hakkında

In the task, you output the MQTT Sample client identification parameters from your version of JAASLoginModule. Writing the client parameters out entails modifying the sample JAASLoginModule program and configuring IBM WebSphere MQ to load your version of JAASLoginModule.

Yordam

1. Paho MQTT Sample istemcisini çalıştırmak için “Compile and run all the MQTT client sample Java apps from Eclipse” sayfa 14 içindeki adımları tamamlayın.

   Amacınız, JAAS kimlik doğrulamasını geliştirmek ve test etmek üzere bir geliştirme ortamı hazırlamak. JAAS kimlik doğrulama modülünü uyarlamak için bir Java geliştirme ortamı gereklidir. In the example, you run the sample Paho client for Java to test your JAAS configuration. Basitlik için, hem örnek istemciyi hem de örnek JAAS oturum açma modülünü değiştirmek için aynı geliştirme ortamını kullanın. Alternatively you test your JAAS login module with the MQTT client for C, or any other MQTT client.

2. İsteğe bağlı: MQTT Paho örneğine bir kullanıcı adı ve parola parametresi ekleyin.

   Not: Java için Paho istemcinizde kullanıcı adı ve parola parametreleri varsa, bu adım gereksiz olur. Karşıdan yükleme için karşıdan yükleme sitesini kontrol edin. Bkz. IBM Messaging Community downloads, tersi durumda Sample.javakopyalarınızı değiştirin.

   a) Open the package explorer in the org.eclipse.paho.sample.mqttv3app package in the Paho samples project.

   b) Sample.java **Kopyala** > **Yapıştır**' ı sağ tıklatın. **Name Conflict** (Ad Çakışması) penceresinde, SampleForJAASadını yazın.

   c) main yöntemine aşağıdaki kod satırlarını ekleyin.

      i) After the line "boolean ssl = false;", declare the userName and password variables:

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

      Daha eski MQTT sunucularıyla uyumluluk için, varsayılan olarak parola ve kullanıcı adı parametrelerini ayarlamayın.

      ii) Satır sonra "case 'v': ssl = Boolean.valueOf(args[++i]).booleanValue(); break;", iki yeni giriş parametresini ayrıştırabiliyor:

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

      iii) Before the line, "if (action.equals("publish")) {", add userName and password to the Sample constructor arguments:

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

   d) Add userName and password to the constructor of Sample.

      Değişiklik:

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

      Hedef:

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

e) Aşağıdaki kod satırlarını Sample oluşturucusuna ekleyin.

After the line "conOpt.setCleanSession(clean);", set the userName and password variables in the conOpt object in the Sample method:

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

f) publish ve subscribe yöntemlerinde, aşağıdaki kod satırlarını değiştirin:

Satırı "client.connect();" olarak değiştirin

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

3. JAAS örneğiniz için bir Java projesi (JAASSample) oluşturun.
   a) Eclipse çalışma alanınızda, **Yeni Java Projesi** sihirbazını açın: **Dosya** > **Yeni** > **Java projesi**öğelerini tıklatın.
   b) **Proje adı** alanına JAASSampleyazın.
   c) JRE seçeneklerinde, yürütme ortamı JRE 'si olarak J2SE-1.5 öğesini seçin ve **İleri**düğmesini tıklatın.

      JRE, IBM WebSphere MQ sunucunuzun çalıştırdığı JRE ile eşleşmelidir. IBM WebSphere MQ Version 7.5 , J2SE-1.5çalıştırır.
   d) **Java Ayarları** penceresinde **Kitaplıklar** sekmesini tıklatın ve **Dış JARS Ekle ...**seçeneğini tıklatın. Şu sisteme göz atın:%MQ_FILE_PATH%\mqxr\lib dizini ve **MQXR.jar**seçeneğini belirleyin; **Son**düğmesini tıklatın.
4. JAAS örneklerini JAASLoginModule ve JAASPrincipal sınıflarını içe aktarın.
   a) Paket Gezgini 'nde JAASSample projesini farenin sağ düğmesiyle tıklatın. **İçe Aktar ...** > **Genel** > **Dosya Sistemi** ve **İleri**' yi tıklatın.
   b) %MQ_FILE_PATH%\mqxr\samples 'a göz atın ve JAASLoginModule.java ve JAASPrincipal.java' a bakın; **Son**' u tıklatın.
   c) Paket Gezgini 'nde (Paket Gezgini) Java dosyalarını seçip farenin sağ düğmesiyle tıklatın. **Yeniden Düzenle ...** > **Taşı**.
   d) In the **Taşı** window, verify JAASSample is selected as the destination for both elements, and click **Paket Oluştur ...**
   e) Type samples into the **Ad** field in the **Yeni Java Paketi** wizard; click **Son** > **Tamam**

   Eclipse , içe aktarılan Java sınıflarını kullanılmayan değerlerle ilgili uyarılarla oluşturur.
5. JAASLoginModule sınıfını yeniden adlandırın

   Rename the class so it is easier to distinguish it from the sample JAASLoginModule class that is shipped with IBM WebSphere MQ.
   a) Paket gezgininde JAASloginModule.java öğesini farenin sağ düğmesiyle tıklatın. **Yeniden Düzenle ...** > **Yeniden adlandır**.
   b) **Derleme Birimini Yeniden Adlandır** penceresinde, **Yeni ad** alanını JAASloginModule 'den MyJAASloginModule' e değiştirin; **Son**' u tıklatın.
6. MyJAASloginModule sınıfını, geri bildirme alanlarının içeriğini çıkışa doğru olarak değiştirin.
   a) Aşağıdaki kod satırını MyJAASloginModule.java' e ekleyin.

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

İfadeden hemen önce satırları yerleştirin:"if (true) loggedIn = true;"

b) İçe aktarma işlemleri yeniden düzenlemek için CTRL+Shift+O tuşuna basın ve dosyayı kaydedin.

7. JAASPrincipal dosyasını MyJAASPrincipal olarak yeniden adlandırın.

   Örnek JAASPrincipal sınıfından karışıklığı önlemek için sınıfı yeniden adlandırın. Örnekte, MyJAASPrincipal sınıfının içeriğini değiştirilmemiş olarak bırakın.

8. Give the user ID that is running the queue manager processes read and execute permissions to your JAAS classes.
   a) Windows Explorer 'da, Eclipse çalışma alanı dizininizi açın. Örnekte, Eclipse çalışma alanı konumu Eclipse değişkeniyle gösterilir, *workspace_loc*.
   b) MyJAASLoginModule ve MyJAASPrincipal sınıflarınızı içeren dizine göz atın.

      Dizin yolu şudur: *workspace_loc*\JAASSample\bin\samples
   c) Her iki sınıfı seçip farenin sağ düğmesiyle tıklatın ve **Özellikler**' i tıklatın; **Özellikler** penceresinde **Güvenlik** sekmesini tıklatın.
   d) **Ekle ...**düğmesini tıklatın. mqmnesne adını yazın ve doğrulamak için **Adları denetle** simgesini tıklatın; **Tamam**simgesini tıklatın.
   e) **Grup ya da kullanıcı adları** listesinde **mqm** öğesini seçin ve mqmile ilgili izinler listesinde **read and execute** ve **read** seçeneğini işaretleyin; Tamam düğmesini tıklatın.

9. Configure IBM WebSphere MQ to run your MyJAASLoginModule class.
   a) MyJAASLoginModule sınıfınızı yüklemek üzere sınıf yollarını tanımlamak için IBM WebSphere MQ yapılandırmasına bir service.env dosyası ekleyin.

      WMQ_DATA_PATH dizininizde aşağıdaki sınıf yolu deyimiyle bir service.env dosyası oluşturun:

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

      Burada *user.dir* ,Eclipse çalışma alanınızda derlenen sınıf dosyalarının dizin köküdür. *WMQ_DATA_PATH* dizini, qmgrs dizinini içerir. Bkz. Ek ortam değişkenleri.

      İpucu: CLASSPATH=*user.dir*\JAASSample\bin might be the only statement in the service.env file
   b) MyJAASLoginModule dosyanızı service.env dosyasındaki sınıf yollarıyla göreli olarak tanımlamak için jaas.config dosyasına bir stanza (MyJAASStanza) ekleyin.

      jaas.config , kuyruk yöneticisi mqxr dizinidir; *WMQ_DATA_PATH*\Qmgrs\MQXR_SAMPLE_QM\mqxr.

      Stanza:

      ```
      MyJAASStanza {
        samples.MyJAASLoginModule required debug=true;
      };
      ```
   c) Configure a IBM WebSphere MQ Telemetry channel with the name of your JAAS configuration stanza.

      Bir komut penceresinde aşağıdaki komutu çalıştırın:

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

      Komut, MyJAAS kanalını jaas.config dosyasındaki MyJAASStanza ' ye bağlar. By not specifying an MCAUSER option, or specifying the USECLTID option on the channel definition, the channel authorizes access to queue manager resources with the user name supplied by the MQTT client program. Örnekte, istemci tarafından sağlanan kullanıcı adı "Guest"olarak ayarlıdır. The existing authorizations for Guest set by the SampleMQM command file are used in this example too.

10. Yeni yapılandırma verilerini okumak için IBM WebSphere MQ Telemetry hizmetini yeniden başlatın.

To restart the IBM WebSphere MQ Telemetry service, start the queue manager, or the service from IBM WebSphere MQ Explorer, or run the following commands for the sample configuration:

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

11. Sample programını çalıştırın.

    SampleForJAAS için bir çalıştırma yapılandırması yapılandırmak üzere, “1” sayfa 72adımında olduğu gibi, aşağıdaki değişikliklerle aynı yordamı izleyin:

    a) Set the port number to 1890 to match the MQTT channel configuration.

    b) Sample programı için oluşturduğunuz Abone ve Yayınlayıcı yapılanışları için, `-u Guest -z password` parametrelerini **(x) = Bağımsız Değişkenler** sekmesindeki parolalara ekleyin.

    Örnek programlar, çıkış içinde herhangi bir değişiklik yapılmadan çalıştırılır; ancak, kapı numarası 1883yerine artık 1890 olur.

    WMQ_DATA_PATH\Qmgrs\MQXR_SAMPLE_QM dizininde mqxr.stdout dosyasını açın. The output from MyJAASLoginModule is written to mqxr.stdout:

```
Username=Guest
Password=passord
ClientId=SampleJavaV3_subscribe
Network address=/127.0.0.1
```

Sonraki adım

Örneğiniz çalışmazsa, JAASiçin sorun giderme konusunu okuyun; “Sorun çözülüyor: JAAS oturum açma modülü telemetri hizmeti tarafından çağrılmadı” sayfa 182ve bu hata ayıklama ipuçlarını deneyin.

1. Add `-ayrıntılı` to the parameters in *WMQ_DATA_PATH*\Qmgrs\MQXR_SAMPLE_QM\mqxr\java.properties. Bu günlüğünüzde, sınıfınızın başarıyla yüklenip yüklenmediğini görebilirsiniz.

   Çıkış, *WMQ_DATA_PATH*\Qmgrs\MQXR_SAMPLE_QM\mqxr.stderr' ye yazılır.

2. MyJAASLoginModule' ta yayınlanan kural dışı durumlar için *WMQ_DATA_PATH*\Qmgrs\MQXR_SAMPLE_QM\errors\mqxr.log içine bakın. Örneğin, bir dizgi değil, karakter dizisi olan bir boş passwordçıkışını çıkış girişiminde bulunursanız, kural dışı durum yayınlanır.
3. *WMQ_DATA_PATH*\Qmgrs\MQXR_SAMPLE_QM\errors\AMQERR01.logiçine bakın. userName içindeki kullanıcı adının kuyruk yöneticisi kaynaklarına erişim yetkisi yoksa ve kanal, MCAUSER ya da USECLTID seçeneğiyle yapılandırıldıysa, burada herhangi bir hata raporlanır.
4. Check the name of the stanza in *WMQ_DATA_PATH*\Qmgrs\MQXR_SAMPLE_QM\mqxr\jaas.config is the same as the name in the MQTT channel that is configured for the port the Sample client is trying to connect to.
5. Check the path in the stanza matches the path to the MyJAASLoginModule class in Eclipse; for example:

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

6. To eliminate whether the fault lies in the class path in the service.env file in *WMQ_DATA_PATH* is not being picked up correctly, change the line "set CLASSPATH=%MQXRCLASSPATH%;%CLASSPATH%" in %MQ_FILE_PATH%\mqxr\bin\controlMQXR.BAT to include your class path. Ayrıca sınıf yolunu da yankıtabilirsiniz. Ancak sınıf yolu, service.envdosyasında ayarlanan sınıf yolunu içermiyor; bu nedenle, bu yalnızca controlMQXR.BAT dosyasını değiştirdiğinizde işlev görmektedir.

İlgili kavramlar

“MQTT güvenliği” sayfa 50

Üç kavram, MQTT güvenliği için temel olur: kimlik, kimlik doğrulama ve yetkilendirme. Kimlik, yetkilendirilmekte olan ve yetki verilen istemciyi adlandırma üzeredir. Kimlik doğrulaması, istemcinin kimliğini kanıtlamayla ilgilidir ve yetkilendirme, istemciye verilen hakların yönetilmesine ilişkin olarak ortaya çıktı.

“Telemetri kanalı JAAS yapılandırması” sayfa 113
İstemci tarafından gönderilen `Kullanıcı adı` 'nın kimliğini doğrulamak için JAAS ' ı yapılandırın.

İlgili görevler

“Sorun çözülüyor: JAAS oturum açma modülü telemetri hizmeti tarafından çağrılmadı” sayfa 182
Find out if your JAAS login module is not being called by the telemetry (MQXR) service, and configure JAAS to correct the problem.

“Güvenli MQTT istemcisi örnek Java uygulaması' in oluşturulması ve çalıştırılması” sayfa 53
Bir Windows örneğine dayalı olarak, either IBM MessageSight or IBM WebSphere MQ as the MQTT serverüzerindeki güvenli örnek Java uygulamasıyla çalışmaya devam edebilir ve bu uygulamayı çalıştırabilirsiniz. You can run an Javaiçin MQTT istemcisi app on any platform with JSE 1.5 or above that is "Java Uyumlu"

“Connecting the MQTT client sample Java app on Android over SSL” sayfa 62
Get up and running with the sample Android MQTT client connected to IBM WebSphere MQ over SSL.

İlgili bilgiler

Ek ortam değişkenleri

MQTT messaging client for JavaScript ile SSL ve WebSocketsarasındaki bağlantı kurulması

Connect your web app securely to IBM WebSphere MQ by using the MQTT messaging client for JavaScript sample HTML pages with SSL and the WebSocket protocol.

Başlamadan önce

1. You must have access to an MQTT version 3 server that supports the MQTT protocol over WebSockets.
2. Tarayıcı SSL 'yi ve WebSocket protocol' yi desteklemelidir. Bkz. “SSL üzerinden mobil ileti sistemi web uygulamaları için tarayıcı desteğindeki kısıtlamalar” sayfa 171.
3. SSL kanalları başlatılmalıdır.

Bu görev hakkında

SSL üzerinden JavaScript örnek sayfaları için MQTT ileti sistemi istemcisini çalıştırmak üzere bu görevi tamamlayın. Bu görev, sertifikaları oluşturmak ve IBM WebSphere MQ'yi yapılandırmak için sizi “Anahtarlar ve sertifikaların oluşturulması” sayfa 94 ' a yönlendirir.

SSL kanalını, sertifika yetkilisi imzalı anahtarlarla ya da kendinden onaylı anahtarlarla sabitleyin.

Yordam

1. İstemci uygulamasını bağlayabileceğiniz bir MQTT sunucusu seçin.

   The server must support the MQTT protocol over secure WebSockets.

   - IBM MessageSight, IBM WebSphere MQ Sürüm 7.5.0.1 ve sonraki yayın düzeyleri, bunu gerçekleştirin.
2. İsteğe bağlı: Sürüm 7 ya da sonraki bir yayın düzeyinde bir Java geliştirme seti (JDK) kurun.

   Bir sınama sistemi ayarlıyorsanız ve kendinden onaylı sertifikalar kullanmak istiyorsanız, sertifikalarınızı onaylamak için JDK Sürüm 7 **keytool** komutunu kullanmanız gerekir. Bir üretim sistemi ayarlıyorsanız ve bir dış sertifika yetkilisine sertifika imzalama istekleri (CSR) gönderiyorsanız, Sürüm 7 JDK ' ye gerek yoktur.

3. Anahtar çiftleri ve sertifikalar oluşturmak için komut dosyalarını oluşturun ve çalıştırın ve IBM WebSphere MQ ' ı MQTT sunucusu olarak yapılandırın.

   Komut dosyalarını oluşturmak ve çalıştırmak için “Anahtarlar ve sertifikaların oluşturulması” sayfa 94 içindeki adımları izleyin. Komut dosyaları da “Windowsiçin SSL sertifikalarını yapılandırmak için komut dosyaları örneği” sayfa 79içinde listelenir.

   Sunucu sertifikasının ortak adı, sunucu kanalının DNS adıyla eşleşmelidir. Bazı tarayıcılar, ortak adların listesini içeren sertifikaları kabul eder; örneğin:

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

   Diğer tarayıcılar yalnızca bir ortak ad kabul eder. Örneğin, Firefox, sürüm 18 'e kadar yalnızca bir ortak ad kabul eder. Sonraki sürümler farklı olabilir.
4. SSL kanallarının çalıştığını ve beklediğiniz şekilde ayarlandığını denetleyin.

   IBM WebSphere MQ' ta, komut penceresine aşağıdaki komutu yazın:

   - 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. Sertifikaları tarayıcı sertifika deposuna kurun.

   Örnek için aşağıdaki sunucu sertifikalarından birini seçin:

   a. Kendinden onaylı bir sunucu sertifikası için, sertifika `srvcertselfsigned.cer` ' dir.

   b. Özel sertifika yetkiliniz tarafından imzalanan bir sunucu sertifikası için sertifika `cacert.cer` ' dir.

   c. Bir dış sertifika yetkilisi tarafından imzalanmış bir sunucu sertifikası için, sertifika kuruluşunun kök sertifikasını sertifika deposunda önceden kurulu olarak denetleyin.

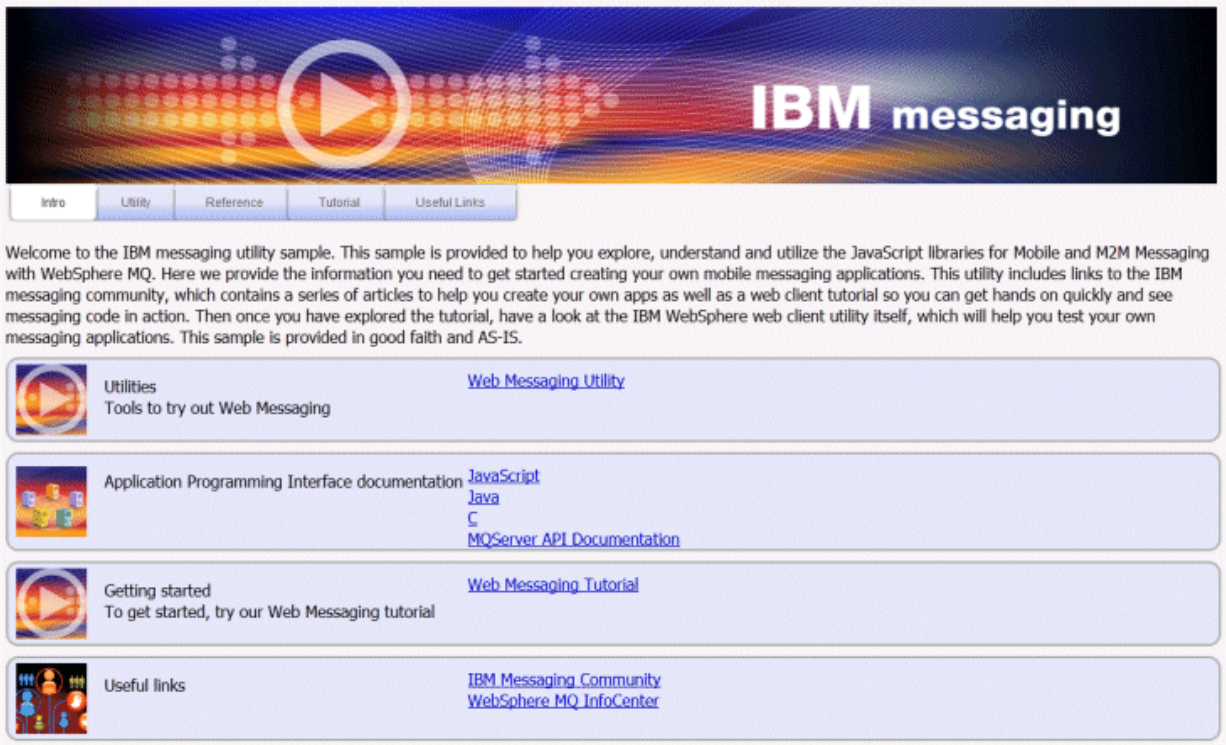
   Depending on the support available in your browser, install `cacert.cer` into the list of Trusted Root Certification Authorities. Bkz. “SSL üzerinden mobil ileti sistemi web uygulamaları için tarayıcı desteğindeki kısıtlamalar” sayfa 171.
6. İsteğe bağlı: İstemcinin kimliğini doğrulayın.

   Örneğin, sunucuyu doğrulamak ve kanalı şifrelemek için `cacert.cer` ' u kursanız, ancak istemcinin kimliğini doğrulamamanız için bu kuruluş işlemi gerçekleştirmeyin. İstemcinin kimliğini doğrulamak için, istemci anahtar deposunu (`cltkeystore.p12`) tarayıcınızı kullanarak tarayıcınızı kurmalısınız. Tüm tarayıcılar, istemcileri doğrularken destek desteklemez. Bkz. “SSL üzerinden mobil ileti sistemi web uygulamaları için tarayıcı desteğindeki kısıtlamalar” sayfa 171.
7. Güvenli WebSockets kanalına bağlanın.

   Tarayıcınızı açın ve adres çubuğuna WebSockets kanalının URL 'sini yazın; örnek:

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

   IBM WebSphere MQ , MQTT ileti sistemi istemcisi örnek JavaScript sayfaları' un ilk sayfasıyla yanıt verir.



Bağlantı başarısız olursa ve örnek komut dosyalarını örnek MQTT kuyruk yöneticisini ayarlamak için çalıştırdıysanız, 1886 numaralı kapıdaki normal WebSockets kanalına bağlanmayı deneyin. 1886 numaralı kapıdaki başarı, SSL bağlantısında hatayı yalıtmayı sağlar.

```
https://localhost:1886
```

İlgili kavramlar

“MQTT messaging client for JavaScript ve web uygulamaları” sayfa 115
“How to program messaging apps in JavaScript” sayfa 119

İlgili görevler

“Anahtarlar ve sertifikaların oluşturulması” sayfa 94
Follow this procedure to generate keys and certificates for Java and C clients, including Android and iOS apps, and the IBM WebSphere MQ and IBM MessageSight servers.

“MQTT messaging client for JavaScriptile çalışmaya başlama” sayfa 23
İleti alışverişi istemcisi örnek giriş sayfasını görüntüleyerek ve bağlarının bulunduğu kaynaklara göz atarak MQTT messaging client for JavaScript ile çalışmaya başlayabilirsiniz. To display this home page, you configure an MQTT server to accept connections from the MQTT ileti sistemi istemcisi örnek JavaScript sayfaları, then you type the URL that you have configured on the server into a web browser. MQTT messaging client for JavaScript otomatik olarak aygıtınızda başlar ve ileti alışverişi istemcisi örnek ana sayfası görüntülenir. Bu sayfa, yardımcı programlar, programlama arabirimi belgeleri, eğitmen ve diğer yararlı bilgiler için bağlantılar içerir.

İlgili başvurular

“SSL üzerinden mobil ileti sistemi web uygulamaları için tarayıcı desteğindeki kısıtlamalar” sayfa 171
Farklı tarayıcılar üzerinde farklı tarayıcılar arasında yetenek farklılıkları vardır. Understanding these differences helps you configure your apps, certificate authorities (CAs) and client certificates to connect using the MQTT messaging client for JavaScript over SSL and WebSockets.

Windowsiçin SSL sertifikalarını yapılandırmak için komut dosyaları örneği

** **

Örnek komut dosyaları, sertifikalar ve sertifika depolarını, görevdeki adımlarda açıklandığı gibi oluşturur. Buna ek olarak, örnek, sunucu sertifika deposunu kullanmak için MQTT istemci kuyruk yöneticisini ayarlar. Bu örnek, IBM WebSphere MQile birlikte verilen `SampleMQM.bat` komut dosyasını çağırarak kuyruk yöneticisini siler ve yeniden yaratır.

`initcert.bat`

`initcert.bat` , sertifikaların adlarını ve yollarını ve **`keytool`** ve **openSSL** komutları için gerekli olan diğer parametreleri ayarlar. Ayarlar, komut dosyasındaki açıklamalarda açıklanır.

```
@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

`cleancert.bat` komut kütüğündeki komutlar, sunucu sertifika deposunun kilitlenmediğinden emin olmak için MQTT istemci kuyruk yöneticisini siler ve örnek güvenlik komut dosyaları tarafından yaratılan tüm anahtar depolarını ve sertifikaları siler.

```
@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

`genkeys.bat` komut kütüğündeki komutlar, özel sertifika yetkiliniz, sunucu ve bir istemci için anahtar çiftleri yaratır.

```
@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

`sscerts.bat` komut dosyasındaki komutlar istemci ve sunucu kendinden onaylı sertifikalarını anahtar depolarından dışa aktarır ve sunucu sertifikasını istemci güvenilirlik deposuna ve istemci sertifikasını sunucu anahtar deposuna (içe) aktarır. Sunucunun güvenilir deposu yok. Komutlar, istemci JKS güvenilirlik deposundan PEM biçiminde bir istemci güvenilirlik deposu yaratır.

```
@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

Komut dosyası, sertifika yetkilisi kök sertifikasını özel anahtar depolarına aktarır. Kök sertifika ile imzalanmış sertifika arasında anahtar zinciri oluşturmak için CA kök sertifikasına gerek vardır. `cacerts.bat` komut dosyası, istemciyi ve sunucu sertifikası isteklerini anahtar depolarından dışa aktarır. Komut dosyası, `cajkskeystore.jks` anahtar deposunda özel sertifika yetkilisinin anahtarı ile sertifika isteklerini imzalar ve imzalı sertifikalar, istek geldiği anahtar depolarına geri aktarılır. İçe aktarma, sertifika zincirini CA kök sertifikasıyla yaratır. Komut dosyası, istemci JKS güvenli deposundan PEM biçiminde bir istemci güvenilirlik deposu yaratır.

```
@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
Komut dosyası, sertifika dizinindeki anahtar depolarını ve sertifikalarını listeler. Daha sonra MQTT örnek kuyruk yöneticisini yaratır ve güvenli telemetri kanallarını yapılandırır.

```
@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
```

Güvenli MQTT istemcisi örnek C uygulaması' in oluşturulması ve çalıştırılması

Bir Windows örneğine dayalı olarak, C kaynağını derleyebileceğiniz herhangi bir işletim sisteminde güvenli örnek C uygulamasıyla çalışmaya devam edebilir ve bu uygulamayı çalıştırabilirsiniz. Örnek C uygulamasını either IBM MessageSight or IBM WebSphere MQ as the MQTT server' ta çalıştırabildiğinizi doğrulayın.

Başlamadan önce

1. SSL üzerinden MQTT protocol ' yi destekleyen bir MQTT version 3.1 sunucusuna erişiminiz olmalıdır.
2. İstemciniz ve sunucu arasında bir güvenlik duvarı varsa, bunun MQTT trafiğini engellediğinden emin olun.
3. İstemcinin C kitaplıklarına ilişkin ikili sürümleri bir dizi işletim sistemi için sağlanır. Bu işletim sistemlerinin bazıları için, istemcinin güvenli sürümü ikili dosya olarak sağlanmaz. For those operating systems, you must follow the instructions in “C kitaplıkları için MQTT istemcisi oluşturulması” sayfa 29.
4. For problem resolution, IBM support might require you to run the MQTT client for C on a reference platform.
5. SSL kanalları başlatılmalıdır.

Desteklenen ve başvuru platformlarına genel bakış için bkz. IBM Mobil İleti Sistemi ve M2M Client Packiçin sistem gereksinimleri. C istemcisi için nelerin desteklenmesine ilişkin ayrıntılar için WebSphere MQ V7.5 Telemetry İçin Sistem Gereksinimleri' in ilgili bölümlerine bakın.

Bu görev hakkında

As an illustration, this article shows you how to compile and run the secure MQTT istemcisi örnek C uygulaması on Windows from the command line. Resimde, istemciyi derlemek için Microsoft Visual Studio 2010 kullanılır. You can modify the command-line scripts to compile and run the sample app on other operating systems, such as Linux and iOS.

Not:

Bu makalede sağlanan Windows komut dosyaları, tüm OpenSSL paketini kaynaktan oluşturmanızı varsayar. If you choose to use the precompiled libraries that IBM provides, you might also prefer to get a precompiled binary release of OpenSSL. Önceden derlenmiş kitaplıklar, iOSile kullanılamaz.

SSL kanalını, sertifika yetkilisi imzalı anahtarlarla ya da kendinden onaylı anahtarlarla sabitleyin.

Yordam

1. İstemci uygulamasını bağlayabileceğiniz bir MQTT sunucusu seçin.

   Sunucu, SSL üzerinden MQTT version 3.1 iletişim kuralını desteklemelidir. IBM WebSphere MQ ve IBM MessageSightde içinde olmak üzere, tüm MQTT sunucuları IBM ' dan bunu yapar. Bkz. “MQTT Server sunucuları ile çalışmaya başlama” sayfa 134.
2. İsteğe bağlı: Sürüm 7 ya da sonraki bir yayın düzeyinde bir Java geliştirme seti (JDK) kurun.

   Sertifikaları onaylamak için **keytool** komutunu çalıştırmak için sürüm 7 gereklidir. Sertifikaları onaylayamayacaksa, Sürüm 7 JDK ' yi zorunlu kılmayın.
3. Oluşturmadığınız platforma bir C geliştirme ortamı kurun.

   Bu konudaki örneklerdeki makefiles, aşağıdaki araçları içerir:

   - iOS For iOS, on Apple Mac with OS X 10.8.2 with the iOS development tools from Xcode.
   - Linux Linuxiçin, gcc sürüm 4.4.6 , Red Hat Enterprise Linux sürümünden 6.2.

     The minimum supported level of the `glibc` C library is 2.12, and of the Linux kernel is 2.6.32.
   - Windows Microsoft Windows, Visual Studio sürüm 10.0için.
4. Mobil İleti Sistemi ve M2M Client Pack dosyasını yükleyin ve MQTT SDK ' yı kurun.

   Kuruluş programı yok, yalnızca karşıdan yüklenen dosyayı genişletiyorsunuz.

   a. Mobil İleti Sistemi ve M2M Client Pack' ı karşıdan yükleyin.
   b. SDK ' yı kurabildiğiniz bir klasör oluşturun.

      `MQTT`klasörünün adını vermek isteyebilirsiniz. Bu klasöre giden yol, burada *`sdkroot`*olarak adlandırılır.
   c. Sıkıştırılmış Mobil İleti Sistemi ve M2M Client Pack dosyasının içeriğini *`sdkroot`*' e genişletin. Genişletme, *`sdkroot`*\SDK' ta başlayan bir dizin ağacı oluşturur.
5. İsteğe bağlı: Follow the steps in “C kitaplıkları için MQTT istemcisi oluşturulması” sayfa 29.

   Bu adımı yalnızca MQTT SDK, hedef işletim sisteminiz için güvenli C istemci kitaplığını içermiyorsa kullanın.

   - Windows Kitaplıklar derlemek için `mqttv3cs.lib` , koşmak için `mqttv3cs.dll` ' dir.
   - Linux Kitaplık `libmqttv3cs.so`
   - iOS Kitaplık `libmqttv3cs.a`
6. Anahtar çiftleri ve sertifikalar oluşturmak için komut dosyalarını oluşturun ve çalıştırın ve IBM WebSphere MQ ' ı MQTT sunucusu olarak yapılandırın.

   Komut dosyalarını oluşturmak ve çalıştırmak için “Anahtarlar ve sertifikaların oluşturulması” sayfa 94 içindeki adımları izleyin. Komut dosyaları da “Windowsiçin SSL sertifikalarını yapılandırmak için komut dosyaları örneği” sayfa 88içinde listelenir.
7. SSL kanallarının çalıştığını ve beklediğiniz şekilde ayarlandığınızı denetleyin.

   IBM WebSphere MQ' ta, komut penceresine aşağıdaki komutu yazın:

- 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. Güvenli MQTT istemcisi örnek C uygulaması' i oluşturmak ve çalıştırmak için komut dosyalarını oluşturun.
   a) Kendinden onaylı sertifikalarla güvenli bir SSL kanalını test etmek için `sscclient.bat` oluşturun ve çalıştırın.
   b) Sertifika yetkilisi imzalı sertifikalarla güvenli bir SSL kanalını test etmek için `cacclient.bat` oluşturun ve çalıştırın.

Sonuçlar

Sonuçlar, güvenli olmayan istemciyi çalıştırmanın benzeridir.

```
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
```

Şekil 16. Güvenli abone

```
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 . . .
```

Şekil 17. Güvenli yayınlayıcı

Güvenli MQTT istemcisi örnek C uygulaması' i çalıştırmak için komut dosyaları

Run the scripts in “Windowsiçin SSL sertifikalarını yapılandırmak için komut dosyaları örneği” sayfa 88 before you run these scripts.

Secure MQTT istemcisi örnek C uygulaması with self-signed certificates.
Bu komut dosyasını, `sscerts.bat` komut dosyasını çalıştırarak oluşturduğunuz kendinden onaylı sertifikalar ile çalıştırın.

```
@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
```

Şekil 18. `sscclient.bat`

MQTT güvenli istemci örnek C uygulamasını, sertifika yetkilisi imzalı sertifikalarla çalıştırın.

Bu komut dosyasını, `cacerts.bat` komut dosyasını çalıştırarak oluşturduğunuz sertifika yetkilisi tarafından imzalanmış sertifikalarla çalıştırın.

```
@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
```

Şekil 19. `cacclient.bat`

İlgili kavramlar

“MQTT güvenliği” sayfa 50

Üç kavram, MQTT güvenliği için temel olur: kimlik, kimlik doğrulama ve yetkilendirme. Kimlik, yetkilendirilmekte olan ve yetki verilen istemciyi adlandırma üzeredir. Kimlik doğrulaması, istemcinin kimliğini kanıtlamayla ilgilidir ve yetkilendirme, istemciye verilen hakların yönetilmesine ilişkin olarak ortaya çıktı.

İlgili görevler

“Anahtarlar ve sertifikaların oluşturulması” sayfa 94

Follow this procedure to generate keys and certificates for Java and C clients, including Android and iOS apps, and the IBM WebSphere MQ and IBM MessageSight servers.

Windowsiçin SSL sertifikalarını yapılandırmak için komut dosyaları örneği

Örnek

Örnek komut dosyaları, sertifikalar ve sertifika depolarını, görevdeki adımlarda açıklandığı gibi oluşturur. Buna ek olarak, örnek, sunucu sertifika deposunu kullanmak için MQTT istemci kuyruk yöneticisini ayarlar. Bu örnek, IBM WebSphere MQile birlikte verilen `SampleMQM.bat` komut dosyasını çağırarak kuyruk yöneticisini siler ve yeniden yaratır.

`initcert.bat`

`initcert.bat` , sertifikaların adlarını ve yollarını ve **`keytool`** ve **openSSL** komutları için gerekli olan diğer parametreleri ayarlar. Ayarlar, komut dosyasındaki açıklamalarda açıklanır.

```
@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

`cleancert.bat` komut kütüğündeki komutlar, sunucu sertifika deposunun kilitlenmediğinden emin olmak için MQTT istemci kuyruk yöneticisini siler ve örnek güvenlik komut dosyaları tarafından yaratılan tüm anahtar depolarını ve sertifikaları siler.

```
@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

genkeys.bat komut kütüğündeki komutlar, özel sertifika yetkiliniz, sunucu ve bir istemci için anahtar çiftleri yaratır.

```
@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

sscerts.bat komut dosyasındaki komutlar istemci ve sunucu kendinden onaylı sertifikalarını anahtar depolarından dışa aktarır ve sunucu sertifikasını istemci güvenilirlik deposuna ve istemci sertifikasını sunucu anahtar deposuna (içe) aktarır. Sunucunun güvenilir deposu yok. Komutlar, istemci JKS güvenilirlik deposundan PEM biçiminde bir istemci güvenilirlik deposu yaratır.

```
@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

Komut dosyası, sertifika yetkilisi kök sertifikasını özel anahtar depolarına aktarır. Kök sertifika ile imzalanmış sertifika arasında anahtar zinciri oluşturmak için CA kök sertifikasına gerek vardır. `cacerts.bat` komut dosyası, istemciyi ve sunucu sertifikası isteklerini anahtar depolarından dışa aktarır. Komut dosyası, `cajkskeystore.jks` anahtar deposunda özel sertifika yetkilisinin anahtarı ile sertifika isteklerini imzalar ve imzalı sertifikalar, istek geldiği anahtar depolarına geri aktarılır. İçe aktarma, sertifika zincirini CA kök sertifikasıyla yaratır. Komut dosyası, istemci JKS güvenli deposundan PEM biçiminde bir istemci güvenilirlik deposu yaratır.

```
@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

Komut dosyası, sertifika dizinindeki anahtar depolarını ve sertifikalarını listeler. Daha sonra MQTT örnek kuyruk yöneticisini yaratır ve güvenli telemetri kanallarını yapılandırır.

```
@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
```

Anahtarlar ve sertifikaların oluşturulması

Follow this procedure to generate keys and certificates for Java and C clients, including Android and iOS apps, and the IBM WebSphere MQ and IBM MessageSight servers.

Başlamadan önce

1. **keytool** komutunun bir kopyasına sahip olmanız gerekir. Tüm **keytool** sürümleri, anahtar depolarının Java anahtar depolarından (JKS) Genel Anahtar Şifreleme Sistemi 'ne (PKCS) dönüştürülmesine ya da sertifika isteklerinin imzalanmasına dönüştürülmez. Bu örnek, bu yeteneklerin her ikisini de destekleyen JDK Sürüm 7.0sürümündeki **keytool** komutunu kullanır.
2. Privacy-Enhanced Mail (PEM) biçiminde olan istemci için istemci için anahtar ve sertifika oluşturmayı düşünüyorsanız, **openSSL** komutunun bir kopyasına sahip olmanız gerekir. openSSL paketini oluşturmak için “C kitaplıkları için MQTT istemcisi oluşturulması” sayfa 29 içindeki adımları izleyin.
3. Change the parameter values in the `initcert.bat` script to meet your needs. Parolaların yazılmasını önlemek için özellikle, parola parametrelerini atlamayı seçebilirsiniz. **keytool** komutu, eksik parolalar için sizden bilgi isteminde bulunur.

Bu görev hakkında

You require keys and certificates to create secure SSL connections between MQTT clients and servers. Bu görev, gereksinim duyduğunuz anahtarları ve sertifikaları oluşturmanın iki farklı yolunu gösterir: kendi kendine imzalanmış ve kendi sertifika yetkiliniz tarafından imzalanmış olarak imzalanmış. İzlediğiniz yöntem, anahtar depolarını ve sertifikaları yönetmeyi planlamanıza bağlıdır.

Bir dış sertifika yetkilisi tarafından imzalanmış sertifikaları kullanmak için, `cacerts.bat`' daki imza adımını, sertifika isteklerini bir dış sertifika yetkilisine göndermeyle değiştirin. Sertifika yetkilisi, imzalanmış sertifikana ek olarak bir ara düzey ve bir kök sertifika döndürebilir. Döndürülen sertifikaların kurulacağı dış sertifika kuruluşu tarafından sağlanan kılavuzları izleyin.

IBM WebSphere MQ sunucusu sertifikaları yalnızca, telemetri kanalı yapılandırma parametrelerinde belirttiğiniz sertifika depounda arar. Bu, JSE `cacerts` mağazasında ek olarak arama yapmaz. Bir Java istemcisi, belirttiğiniz güvenli depoda sertifikaları arar. Bir güvenilirlik deposu belirtmezseniz, bu, JSE `jre\lib\security` dizininde `cacerts` anahtar deposunda arar. Android istemcileri, Android aygıtında önceden tanımlı sertifika depolarında sertifikaları arar. C istemcisi uygulamaları ve iOS uygulamaları yalnızca, uygulamanın belirttiği sertifika depolarında arama yapabilir.

Android ve Java istemcileri güvenilen sertifikalar için önceden yapılandırılmış bir güvenli depo aramanızı sağlar. CA kök sertifikaları Android güvenilir sertifika deposunda ve JSE `jre\lib\security\cacerts` mağazasında depolanır. Sunucu sertifikasını sertifika alan CA ' nın kök sertifikası önceden yapılandırılmış güvenilirlik deposunda kuruluysa, bir istemci güvenilirlik deposu tanımlamayın. Gerekli olan tek yapılandırma, güvenli MQTT sunucu kanalı için TCP/IP kapısını ayarlardır.

Anahtarlar ve sertifikalar yaratmak ve tüm farklı biçimleri yönetmek için kullanılan araçlar basit değildir. Yönetilecek çok sayıda parametre vardır ve **openSSL** için bir yapılandırma dosyası, `openssl.cnf`ve komut satırı parametreleri gerekir. Hiçbir araç, hem C hem de Javaüzerinde çalışan uygulamalara ilişkin anahtarları ve sertifikaları yönetmek için gerekli olan tüm işlevleri sağlamıyor. IBM WebSphere MQ içindeki telemetri kanalları bir JKS anahtar deposu gerektirir; bu nedenle, örnekler ağırlıklı olarak Java sertifika araçlarını, **ikeyman** ve **keytool**' i kullanır. Ancak, Java araçları, C istemcisi uygulamaları için gerekli olan PEM biçimini desteklemez. Anahtar depolarını PEM biçiminde oluşturmak için **openSSL** aracını çalıştırın. **openSSL** aracı, anahtar depolarını PKCS12 biçiminden PEM biçimine dönüştürür ve **keytool** , JKS biçimi ile PKCS12 biçimi arasındaki anahtar depolarını dönüştürür. Anahtar deposu dönüştürmesi için `openssl.cnf` dosyası gerekli değil. Yalnızca C istemcisi uygulamaları ya da iOS

uygulamaları oluşturmayı planlıyorsanız, **openSSL** gereklidir. **openSSL**ile çalışmayı tercih ederseniz, sertifikaları **keytool**ile imzalamak yerine sertifika imzalamak için kullanabilirsiniz.

Yordam

1. Aşağıdaki komut dosyalarını çalıştırmak için bir komut penceresi açın.
2. MQTT güvenli örnek istemcilerini çalıştırmak için gereken parametreleri ayarlamak için `initcert.bat` komut dosyasını oluşturun ve çalıştırın.
3. Yeni anahtar depoları ve sertifikalar yaratmak üzere hazır ortamı temizlemek için `cleancert.bat` komut dosyasını oluşturun ve çalıştırın.
4. Gereksinim duyduğunuz anahtar çiftlerini oluşturmak için `genkeys.bat` komut dosyasını oluşturun ve çalıştırın.
5. Aşağıdaki seçeneklerden birini yapın:
   - Kendinden onaylı sertifikalar oluşturmak için `sscerts.bat` komut dosyasını oluşturun ve çalıştırın.
   - Sertifika yetkilisi imzalı sertifika oluşturma zincirleri oluşturmak için `cacerts.bat` komut dosyasını oluşturun ve çalıştırın.
6. `MQXR_SAMPLE_QM` kuyruk yöneticisini yaratmak ve telemetri kanallarını yapılandırmak için `mqcerts.bat` komut dosyasını oluşturun ve çalıştırın.

İlgili görevler

“Güvenli MQTT istemcisi örnek C uygulaması' in oluşturulması ve çalıştırılması” sayfa 84
Bir Windows örneğine dayalı olarak, C kaynağını derleyebileceğiniz herhangi bir işletim sisteminde güvenli örnek C uygulamasıyla çalışmaya devam edebilir ve bu uygulamayı çalıştırabilirsiniz. Örnek C uygulamasını either IBM MessageSight or IBM WebSphere MQ as the MQTT server' ta çalıştırabildiğinizi doğrulayın.

“Güvenli MQTT istemcisi örnek Java uygulaması' in oluşturulması ve çalıştırılması” sayfa 53
Bir Windows örneğine dayalı olarak, either IBM MessageSight or IBM WebSphere MQ as the MQTT serverüzerindeki güvenli örnek Java uygulamasıyla çalışmaya devam edebilir ve bu uygulamayı çalıştırabilirsiniz. You can run an Javaiçin MQTT istemcisi app on any platform with JSE 1.5 or above that is "Java Uyumlu"

Windowsiçin SSL sertifikalarını yapılandırmak için komut dosyaları örneği

Örnek

Örnek komut dosyaları, sertifikalar ve sertifika depolarını, görevdeki adımlarda açıklandığı gibi oluşturur. Buna ek olarak, örnek, sunucu sertifika deposunu kullanmak için MQTT istemci kuyruk yöneticisini ayarlar. Bu örnek, IBM WebSphere MQile birlikte verilen `SampleMQM.bat` komut dosyasını çağırarak kuyruk yöneticisini siler ve yeniden yaratır.

`initcert.bat`

`initcert.bat` , sertifikaların adlarını ve yollarını ve **keytool** ve **openSSL** komutları için gerekli olan diğer parametreleri ayarlar. Ayarlar, komut dosyasındaki açıklamalarda açıklanır.

```
@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

`cleancert.bat` komut kütüğündeki komutlar, sunucu sertifika deposunun kilitlenmediğinden emin olmak için MQTT istemci kuyruk yöneticisini siler ve örnek güvenlik komut dosyaları tarafından yaratılan tüm anahtar depolarını ve sertifikaları siler.

```
@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

genkeys.bat komut kütüğündeki komutlar, özel sertifika yetkiliniz, sunucu ve bir istemci için anahtar çiftleri yaratır.

```
@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

sscerts.bat komut dosyasındaki komutlar istemci ve sunucu kendinden onaylı sertifikalarını anahtar depolarından dışa aktarır ve sunucu sertifikasını istemci güvenilirlik deposuna ve istemci sertifikasını sunucu anahtar deposuna (içe) aktarır. Sunucunun güvenilir deposu yok. Komutlar, istemci JKS güvenilirlik deposundan PEM biçiminde bir istemci güvenilirlik deposu yaratır.

```
@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

Komut dosyası, sertifika yetkilisi kök sertifikasını özel anahtar depolarına aktarır. Kök sertifika ile imzalanmış sertifika arasında anahtar zinciri oluşturmak için CA kök sertifikasına gerek vardır. `cacerts.bat` komut dosyası, istemciyi ve sunucu sertifikası isteklerini anahtar depolarından dışa aktarır. Komut dosyası, `cajkskeystore.jks` anahtar deposunda özel sertifika yetkilisinin anahtarı ile sertifika isteklerini imzalar ve imzalı sertifikalar, istek geldiği anahtar depolarına geri aktarılır. İçe aktarma, sertifika zincirini CA kök sertifikasıyla yaratır. Komut dosyası, istemci JKS güvenli deposundan PEM biçiminde bir istemci güvenilirlik deposu yaratır.

```
@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

Komut dosyası, sertifika dizinindeki anahtar depolarını ve sertifikalarını listeler. Daha sonra MQTT örnek kuyruk yöneticisini yaratır ve güvenli telemetri kanallarını yapılandırır.

```
@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
```

MQTT istemcisi tanıtıcısı, yetki kimliği ve kimlik doğrulaması

Telemetri (MQXR) hizmeti, MQTT istemcileri adına, MQTT kanallarını kullanarak WebSphere MQ konularına abone olur ya da abone olur. WebSphere MQ yöneticisi, WebSphere MQ yetkilendirmesi için kullanılan MQTT kanal kimliğini yapılandırır. Yönetici, kanal için ortak bir kimlik tanımlayabilir ya da kanala bağlı bir istemcinin `Kullanıcı Adı` ya da `ClientIdentifier` ' ini kullanabilir.

Telemetri (MQXR) hizmeti, istemci tarafından sağlanan `Kullanıcı adı` ' yı kullanarak ya da bir istemci sertifikası kullanarak istemcinin kimliğini doğrulayabilir. `Kullanıcı Adı` ' in kimliği, istemci tarafından sağlanan bir parola kullanılarak doğrulanır.

Özetlemek için: Müşteri kimliği, istemci kimliğinin seçimidir. Bağlama bağlı olarak, istemci `ClientIdentifier`, `Username`(Kullanıcı Adı), yönetici tarafından oluşturulan ortak bir istemci kimliği ya da istemci sertifikası ile tanımlanır. Özgünlük denetimi için kullanılan istemci tanıtıcısının, yetkilendirme için kullanılan tanıtıcı olması gerekmez.

MQTT istemci programları, bir MQTT kanalını kullanarak sunucuya gönderilen `Kullanıcı adı` ve `Parola` adını ayarlar. Ayrıca, bağlantıyı şifrelemek ve bağlantıyı doğrulamak için gereken SSL özelliklerini de ayarlayabilirler. Yönetici, MQTT kanalının doğrulanıp doğrulanmayacağını ve kanalın nasıl doğrulanıp doğrulanmayacağını belirler.

MQTT istemcisine IBM WebSphere MQ nesnelerine erişim yetkisi vermek için, istemcinin `ClientIdentifier`ya da `Kullanıcı Adı` ' ini yetkilendirin ya da ortak bir istemci kimliğini yetkilendirin. Bir istemcinin IBM WebSphere MQ'a bağlanmasına izin vermek için, `Kullanıcı adı`' nın kimliğini doğrulayın ya da bir istemci sertifikası kullanın. `Kullanıcı Adı`kimliğini doğrulamak için JAAS 'u yapılandırın ve bir istemci sertifikasının kimliğini doğrulamak için SSL' yi yapılandırın.

İstemcide bir `Parola` ayarlıysa, VPN ' yi kullanarak bağlantıyı şifreleyin ya da parolayı özel tutmak için MQTT kanalını SSL kullanacak şekilde yapılandırın.

İstemci sertifikalarını yönetmek zordur. Bu nedenle, parola kimlik doğrulamasıyla ilgili riskler kabul edilebilir bir durumsa, parola doğrulaması genellikle istemcilerin kimliğini doğrulamak için kullanılır.

İstemci sertifikasını yönetmenin ve depolamanın güvenli bir yolu varsa, sertifika kimlik doğrulamasına güvenmek mümkün olur. Ancak, sertifikaların telemetrenin kullanıldığı ortam tiplerinde güvenli bir şekilde yönetilebilir olması çok ender bir durum. Bunun yerine, istemci sertifikalarını kullanan aygıtların kimlik doğrulaması, sunucudaki istemci parolalarının doğrulanarak tamamlanır. Ek karmaşıklık nedeniyle, istemci sertifikalarının kullanımı son derece hassas uygulamalarla sınırlandırılmıştır. İki kimlik doğrulama biçiminin kullanılması iki etkenli kimlik doğrulaması olarak adlandırılır. Parola gibi faktörlerden birini bilmeniz gerekir; örneğin, bir sertifika gibi.

chip-ve-pin cihazı gibi son derece hassas bir uygulamada, dahili donanım ve yazılıma müdahale etmek için üretim sırasında cihaz kapalı olarak kilitlenir. Aygıta güvenilen, zaman sınırlanmış bir istemci sertifikası kopyalanmıştır. Aygıt, kullanılabileceği yere konuşlandırılır. Aygıt her kullanışında, parola kullanılarak ya da akıllı karttan başka bir sertifika kullanılarak daha fazla kimlik doğrulaması gerçekleştirilir.

MQTT istemcisi kimliği ve yetkilendirmesi

WebSphere MQ nesnelerine erişmek için yetkilendirme için `ClientIdentifier`, `Kullanıcı adı`ya da ortak bir istemci kimliği kullanın.

IBM WebSphere MQ yöneticisi, MQTT kanalının kimliğini seçmek için üç seçeneğe sahiptir. Yönetici, istemci tarafından kullanılan MQTT kanalını tanımlarken ya da değiştirirken seçimi yapar. Kimlik, IBM WebSphere MQ konularına erişimi yetkilendirmek için kullanılır. Seçenekler şunlardır:

1. İstemci tanıtıcısı.
2. Yöneticinin kanal için sağladığı kimlik.
3. The `Kullanıcı Adı` passed from the MQTT client.

`Kullanıcı adı` , MqttConnectSeçenekleri sınıfının bir öznitesidir. İstemcinin hizmete bağlamadan önce ayarlanması gerekir. Varsayılan değeri boş değerli.

Hangi nesnelerin ve hangi işlemlerin MQTT kanalı ile ilişkili kimlik tarafından kullanılması için yetkilendirileceğini seçmek için IBM WebSphere MQ **setmqaut** komutunu kullanın. For example, to authorize a channel identity, `MQTTClient`, provided by the administrator of queue manager, `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
```

İlgili bilgiler

MQTT istemcilerinin WebSphere MQ nesnelerine erişmesi için yetki verme

Parola kullanarak MQTT istemcisi kimlik doğrulaması

İstemci parolasını kullanarak `Kullanıcı adı` ' nın kimliğini doğrulayın. İstemcinin kimliğini doğrulamak ve konulara abone olmak için istemciyi yetkilendirmek için kullanılan kimliğe farklı bir kimlik kullanarak doğrulayabilirsiniz.

Telemetri (MQXR) hizmeti, istemcinin `Kullanıcı adı`kimliğini doğrulamak için JAAS ' ı kullanır. JAAS , MQTT istemcisi tarafından sağlanan `Parola` ' yı kullanır.

The IBM WebSphere MQ administrator decides whether to authenticate the `Kullanıcı Adı`, or not to authenticate at all, by configuring the MQTT channel a client connects to. İstemciler farklı kanallara atanabilir ve her kanal, istemcilerinin kimliklerini farklı şekillerde doğrulamak üzere yapılandırılabilir. JAAS' ı kullanarak, istemcinin kimliğini doğrulamak zorunda olan ve isteğe bağlı olarak istemciyi doğrulayabilecek yöntemleri yapılandırabilirsiniz.

Kimlik doğrulaması için kimlik seçimi, yetkilendirme için kimlik seçmesini etkilemez. Yönetimle ilgili kolaylıklar için bir yetkilendirme için ortak bir kimlik ayarlamak isteyebilirsiniz, ancak her bir kullanıcının kimliğini kullanması için kimlik doğrulaması yapmak isteyebilirsiniz. Aşağıdaki yordama göre, tek tek kullanıcıların ortak bir kimliğe sahip olması için kimlik doğrulama adımları özetlenmiştir:

1. The IBM WebSphere MQ administrator sets the MQTT channel identity to any name, such as `MQTTClientUser`, using IBM WebSphere MQ Explorer.
2. IBM WebSphere MQ yöneticisi, `MQTTClient` ' e herhangi bir konuyu yayınlayıp abone olmak için yetki verir:

   ```
   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. MQTT istemcisi uygulama geliştiricisi, sunucuya bağlanmadan önce bir `MqttConnectOptions` nesnesi yaratır ve `Kullanıcı adı` ve `Parola` seçeneğini ayarlar.
4. The security developer creates a JAAS `LoginModule` to authenticate the `Kullanıcı Adı` with the `Parola` and includes it in the JAAS configuration file.
5. The IBM WebSphere MQ administrator configures the MQTT channel to authenticate the `UserName` of the client using JAAS.

SSL kullanan MQTT istemcisi kimlik doğrulaması

Connections, her zaman MQTT istemcisi tarafından başlatılır. MQTT istemcisi her zaman SSL istemcisidir. MQTT istemcisine ilişkin sunucu ve sunucu kimlik doğrulamasının istemci kimlik doğrulaması her ikisi de isteğe bağlıdır.

İstemciyi özel imzalı bir dijital sertifikayla sağlayarak, MQTT istemcisinin kimliğini IBM WebSphere MQolarak doğrulayabilirsiniz. IBM WebSphere MQ Administrator, MQTT istemcilerini SSL kullanarak kuyruk yöneticisine kendilerini doğrulamaya zorlayabilir. İstemci kimlik denetimini yalnızca karşılıklı kimlik doğrulamanın bir parçası olarak isteyebilirsiniz.

SSL ' yi kullanmaya alternatif olarak, IPsec gibi bir tür Sanal Özel Ağ (VPN), bir TCP/IP bağlantısının uç noktalarını doğrular. VPN, ağ üzerinden akan her bir IP paketini şifreler. Böyle bir VPN bağlantısı kurulduktan sonra, güvenilir bir ağ kurmuyorsunuz. MQTT istemcilerini, VPN ağı üzerinden TCP/IP kullanarak telemetri kanallarına bağlayabilirsiniz.

Client kimlik doğrulaması, istemciye bir sır vermesinden kaynaklanıyor. Gizli, kendinden onaylı sertifika ya da bir sertifika yetkilisi tarafından sağlanan bir anahtarda istemcinin özel anahtrıdır. Anahtar, istemcinin dijital sertifikasını imzalamak için kullanılır. İlgili genel anahtara sahip olan herkes, sayısal sertifikayı doğrulayabilir. Sertifikalar güvenilir, ya da zincirlenmişse, sertifika zincirinden güvenilir bir kök sertifikadan geriye doğru izlenebilir. İstemci doğrulaması, istemci tarafından sunucuya sağlanan sertifika zincirindeki tüm sertifikaları sunucuya gönderir. Sunucu, sertifika zincirini güvendiği bir sertifika buluncaya kadar denetler. Güvenilir sertifika, kendinden onaylı bir sertifikadan oluşturulan genel sertifikadır ya da genellikle bir sertifika yetkilisi tarafından verilen bir kök sertifikadır. Son olarak, isteğe bağlı olarak, güvenilir sertifikayı "canlı" bir sertifika iptal listesiyle karşılaştırılabilir.

Güvenilen sertifika, bir sertifika yetkilisi tarafından yayınlanabilir ve önceden JRE sertifika deposuna eklenmiş olabilir. Kendinden onaylı bir sertifika ya da telemetri kanalı anahtar deposuna güvenilen bir sertifika olarak eklenmiş olan herhangi bir sertifika olabilir.

Not: Telemetri kanalı, bir ya da daha fazla telemetri kanalına özel anahtarları ve istemcileri doğrulamak için gerekli tüm genel sertifikalara sahip bir birleşik anahtar deposu/güvenilirlik deposu içerir. Bir SSL kanalının bir anahtar deposu olması ve kanal truststore ile aynı dosyaya sahip olması gerektiğinden, JRE sertifika deposuna hiçbir zaman gönderme yapılmamaktadır. Bunun anlamı, bir istemcinin kimlik doğrulaması bir CA kök sertifikası gerektiriyorsa, CA kök sertifikası zaten JRE sertifika deposunda olsa bile, kanal için kök sertifikayı, kanal için anahtar deposuna yerleştirmeniz gerekir. JRE sertifika deposuna hiçbir zaman gönderme yapılmamaktadır.

İstemci kimlik doğrulamasının sayacağı tehditleri ve müşterinin ve sunucunun tehditlere karşı koymak için rol oynadığı tehditleri düşünün. Tek başına istemci sertifikasının doğrulanması, bir sisteme yetkisiz erişimi önlemek için yeterli değildir. İstemci aygıtı başka biri tarafından tutulduysa, istemci aygıtının sertifika sahibinin yetkisiyle hareket etmesi gerekmez. İstenmeyen saldırılara karşı tek bir savunmaya asla güvenmeyin. En azından iki faktörli kimlik doğrulama yaklaşımı ve özel bilgiler bilgisine sahip bir sertifikana ek sahip olma. Örneğin, JAAS' ı kullanın ve sunucu tarafından yayınlanan bir parola kullanarak istemcinin kimliğini doğrulayın.

Müşteri sertifikasına ilişkin birincil tehdit, yanlış ellere geçmektedir. Sertifika, istemcide bir parola korumalı anahtar deposunda tutulur. Anahtar depoya nasıl yerleştirilsin? MQTT istemcisi parolayı nasıl anahtar deposuna alır? Parola korumasının güvenliği ne kadar güvenli? Telemetri cihazları çoğu zaman kolayca çıkarılabilir ve daha sonra özel olarak hacklenebilirler. Aygıt donanımını kurcalaması gerekiyor mu? İstemci tarafı sertifikalarının dağıtılması ve korunması zor olduğu tanınmaktadır; bu, anahtar yönetimi sorunu olarak adlandırılır.

İkincil bir tehdit, aygıtın istenmeyen yollarla sunuculara erişmek için yanlış kullanılsa da olabilir. Örneğin, MQTT uygulaması kurcalandıysa, kimliği doğrulanmış istemci kimliğini kullanarak sunucu yapılanışındaki bir zayıflığı kullanmak mümkün olabilir.

SSL kullanan bir MQTT istemcisini doğrulamak için telemetri kanalını ve istemciyi yapılandırın.

-
-

SSL kullanan istemci kimlik doğrulaması için MQTT istemcisi yapılanışı

SSL kullanan MQTT istemcisini doğrulamak için, istemci SSL kullanarak bir telemetri kanalına bağlanır. SSL istemcilerini doğrulamak için yapılandırılmış bir telemetri kanalına karşılık gelen bir TCP kapısı belirtmelidir.

Örneğin, istemcide:

```
MQTTClient mqttClient = new MqttClient( "ssl://www.example.org:8884", "clientId1");
mqttClient.connect();
```

İstemci JVM 'nin JSSE' deki standart yuva üreticisini kullanması gerekir. Java ME kullanıyorsanız, JSSE paketinin yüklendiğinden emin olmanız gerekir. Java SE kullanıyorsanız, Java sürümü 1.4.1olduğundan JSSE JRE ' ye dahil edilmiştir.

SSL bağlantısı, bağlanmadan önce bir dizi SSL özelliği ayarlanmasını gerektirir. You can set the properties either by passing them to the JVM using the `-D` switch, or you can set the properties using the `MqttConnectionOptions.setSSLProperties` method.

Standart olmayan bir yuva üreticisini yüklerseniz, `MqttConnectOptions.setSocketFactory(javax.net.SocketFactory)` yöntemini çağırarak, SSL ayarlarının ağ yuvasına geçirilir olması uygulama tanımlıdır.

İstemcinin özel anahtarını kullanarak ya da istemcideki parola korumalı anahtar deposuna bir CA tarafından imzalanmış olan istemcinin sayısal sertifikasını ekleyin. Sertifikanda bir anahtar zinciri varsa, sertifikaları anahtar zincirinden mağazaya ekleyebilirsiniz. Sunucu istemci sertifikasını doğruladığında, istemci tarafından gönderilen sertifikaların anahtar depolarındaki sertifikalarla eşleşmesini sağlar. Anahtar zincirindeki ilk eşleşmeyi, sahip olduğu bir sertifikayla arıyor. Anahtar zincirinin geri kalan kısmı yok sayılır.

MQTT istemcisi, anahtar depolarındaki tüm sertifikaları sunucuya gönderir. Sunucu, istemcinin gönderdiği anahtar zincirlerinden herhangi birini doğrularsa, istemcinin kimliği doğrulanır.

İstemci kimlik doğrulaması için SSL şifreleme takımlarını da kullanabilirsiniz. Burada, şu anda desteklenmekte olan SSL şifreleme takımlarının alfabetik listesi gösterilmektedir:

- `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`

- 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 SHA-2 şifreleme takımlarını kullanmayı planlıyorsanız, bkz. “System requirements for using SHA-2 cipher suites with MQTT clients” sayfa 170.

İlgili kavramlar

“SSL kullanan kanal kimlik doğrulaması için MQTT istemcisi yapılanışı” sayfa 106
Telemetri kanalını SSL kullanarak doğrulamak için, istemci, telemetri kanalına SSL kullanarak bağlanmalıdır. Bu, SSL için yapılandırılmış bir telemetri kanalına karşılık gelen bir kapı belirtmelidir. Yapılandırma, sunucunun özel olarak imzalanmış dijital sertifikasını içeren bir geçiş tümcecik korumalı anahtar deposu içermelidir.

SSL kullanan telemetri kanalı kimlik doğrulaması

Connections, her zaman MQTT istemcisi tarafından başlatılır. MQTT istemcisi her zaman SSL istemcisidir. MQTT istemcisine ilişkin sunucu ve sunucu kimlik doğrulamasının istemci kimlik doğrulaması her ikisi de isteğe bağlıdır.

İstemci, anonim bağlantıyı destekleyen bir CipherSpec kullanmak üzere yapılandırılmadıkça, istemci her zaman sunucuyu doğrulamayı dener. Kimlik doğrulama başarısız olursa, bağlantı kurulmaz.

SSL ' yi kullanmaya alternatif olarak, IPsec gibi bir tür Sanal Özel Ağ (VPN), bir TCP/IP bağlantısının uç noktalarını doğrular. VPN, ağ üzerinden akan her bir IP paketini şifreler. Böyle bir VPN bağlantısı kurulduktan sonra, güvenilir bir ağ kurmuyorsunuz. MQTT istemcilerini, VPN ağı üzerinden TCP/IP kullanarak telemetri kanallarına bağlayabilirsiniz.

Server kimlik doğrulaması, gizli bilgi göndermek üzere olduğunuz sunucuyu kimlik doğrulamasını gerçekleştirir. İstemci, sunucudan gönderilen sertifikalarla, güvenli depoya yerleştirilen sertifikalara ya da JRE `cacerts` deposunda eşleşen denetimleri gerçekleştirir.

JRE sertifika deposu bir JKS dosyasıdır, `cacerts`. It is located in `JRE InstallPath\lib\security\`. It is installed with the default password `changeit`. JRE sertifika deposunda ya da istemci güvenilirlik deposunda güvendiğiniz sertifikaları saklayabilirsiniz. Her iki mağaza da kullanamazsınız. Genel sertifikaların istemcilerin diğer Java uygulamalarının kullandığı sertifikalardan ayrı olarak güvendiği sertifikalarını alıkoymak istiyorsanız, istemci güvenli deposunu kullanın. İstemcide çalışan tüm Java uygulamaları için ortak bir sertifika deposu kullanmak istiyorsanız, JRE sertifika deposunu kullanın. JRE sertifika deposunu kullanmaya karar verirseniz, içerdiği sertifikalar onlara güvendiğinizden emin olmak için bu sertifikayı gözden geçirin.

Farklı bir güven sağlayıcı belirterek JSSE yapılandırmasını değiştirebilirsiniz. Bir sertifika üzerinde farklı denetimler gerçekleştirmek için bir güven sağlayıcısını özelleştirebilirsiniz. In some OGSi environments that have used the MQTT client, the environment provides a different trust provider.

Telemetri kanalını SSL kullanarak doğrulamak için sunucuyu ve istemciyi yapılandırın.

•

İlgili kavramlar

“SSL kullanan kanal kimlik doğrulaması için MQTT istemcisi yapılanışı” sayfa 106
Telemetri kanalını SSL kullanarak doğrulamak için, istemci, telemetri kanalına SSL kullanarak bağlanmalıdır. Bu, SSL için yapılandırılmış bir telemetri kanalına karşılık gelen bir kapı belirtmelidir. Yapılandırma, sunucunun özel olarak imzalanmış dijital sertifikasını içeren bir geçiş tümcecik korumalı anahtar deposu içermelidir.

SSL kullanan kanal kimlik doğrulaması için MQTT istemcisi yapılanışı

Telemetri kanalını SSL kullanarak doğrulamak için, istemci, telemetri kanalına SSL kullanarak bağlanmalıdır. Bu, SSL için yapılandırılmış bir telemetri kanalına karşılık gelen bir kapı belirtmelidir. Yapılandırma, sunucunun özel olarak imzalanmış dijital sertifikasını içeren bir geçiş tümcecik korumalı anahtar deposu içermelidir.

Örneğin, istemcide:

```
MQTTClient mqttClient = new MqttClient( "ssl://www.example.org:8884", "clientId1");
mqttClient.connect();
```

İstemci JVM 'nin JSSE' deki standart yuva üreticisini kullanması gerekir. Java ME kullanıyorsanız, JSSE paketinin yüklendiğinden emin olmanız gerekir. Java SE kullanıyorsanız, Java sürümü 1.4.1olduğundan JSSE JRE ' ye dahil edilmiştir.

SSL bağlantısı, bağlanmadan önce bir dizi SSL özelliği ayarlanmasını gerektirir. You can set the properties either by passing them to the JVM using the `-D` switch, or you can set the properties using the `MqttConnectionOptions.setSSLProperties` method.

Standart olmayan bir yuva üreticisini yüklerseniz, `MqttConnectOptions.setSocketFactory(javax.net.SocketFactory)` yöntemini çağırarak, SSL ayarlarının ağ yuvasına geçirilir olması uygulama tanımlıdır.

İstemcinin, SSL kullanarak telemetri kanalına bağlanmasını sağlar ve istemciyi, sunucu sertifikasına aşağıdaki üç yoldan biriyle güvenecek şekilde yapılandırın:

`cacerts` mağazasında tanınan sertifika kuruluşu tarafından imzalanan bir sunucu sertifikasının kullanılması.

Sunucu, sertifika zincirindeki tüm ara anahtarları gönderirse, ek yapılandırma yok. İstemcinin JRE ' deki `cacerts` deposındaki sertifikaları gözden geçirmeniz ve parolayı `cacerts` deposuna değiştirmeniz önerilir.

Diğer sertifikalar

Güvendiğiniz sertifikaları, istemcide güvenilir depoya saklayın. You must store at least one of the certificates in the certificate chain in the truststore, Set the truststore parameters in `MqttConnectionOptions.SSLProperty`.

- `com.ibm.ssl.trustStore`
- `com.ibm.ssl.trustStorePassword`

Özel bir güvenilirlik yöneticisinin kullanılması

Bir güven sağlayıcı uygulayın ve bu sağlayıcıyı, kullanılan algoritmanın adını geçirin. Sağlayıcı sınıfının adını ve `MqttConnectionOptions.SSLProperty`içinde kullanılacak algoritmayı ayarlayın.

- `com.ibm.ssl.trustStoreProvider`
- `com.ibm.ssl.trustStoreManager`

Kanal kimlik doğrulaması için SSL şifreleme takımlarını da kullanabilirsiniz. Burada, şu anda desteklenmekte olan SSL şifreleme takımlarının alfabetik listesi gösterilmektedir:

- `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 SHA-2 şifreleme takımlarını kullanmayı planlıyorsanız, bkz. “System requirements for using SHA-2 cipher suites with MQTT clients” sayfa 170.

İlgili kavramlar

“SSL kullanan istemci kimlik doğrulaması için MQTT istemcisi yapılanışı” sayfa 103
SSL kullanan MQTT istemcisini doğrulamak için, istemci SSL kullanarak bir telemetri kanalına bağlanır. SSL istemcilerini doğrulamak için yapılandırılmış bir telemetri kanalına karşılık gelen bir TCP kapısı belirtmelidir.

telemetri kanallarında yayın gizliliği

Telemetri kanallarında her iki yönde gönderilen MQTT yayınlarının gizliliği, bağlantı üzerinden iletimleri şifrelemek için SSL kullanılarak güvenli kılınmaktadır.

Telemetri kanallarına bağlanan MQTT istemcileri, kanalda iletilen yayınların gizliliğini simetrik anahtar şifrelemesi kullanarak güvenli kılmak için SSL ' yi kullanır. Uç noktaların kimliği doğrulanmadığı için, kanal şifrelemesine tek başına güvenemezsiniz. Güvenli gizliliği sunucu ya da karşılıklı kimlik doğrulamasıyla birleştirin.

SSL ' yi kullanmaya alternatif olarak, IPsec gibi bir tür Sanal Özel Ağ (VPN), bir TCP/IP bağlantısının uç noktalarını doğrular. VPN, ağ üzerinden akan her bir IP paketini şifreler. Böyle bir VPN bağlantısı kurulduktan sonra, güvenilir bir ağ kurmuyorsunuz. MQTT istemcilerini, VPN ağı üzerinden TCP/IP kullanarak telemetri kanallarına bağlayabilirsiniz.

Kanalı şifreleyen ve sunucuyu doğrulayan tipik bir yapılandırma için “SSL kullanan telemetri kanalı kimlik doğrulaması” sayfa 105' a danışın.

SSL bağlantılarının, sunucunun kimlik doğrulaması olmadan şifrelenmesi, ortadaki saldırılara karşı olan bağlantıyı sağlar. Değiştirdiğiniz bilgiler kulak misafirine karşı korunsa da, kiminle değiş tokuş edildiğinizi bilmenize gerek yok. Ağı denetlemezseniz, IP iletimlerinizi ya da uç nokta olarak maskeleme işlemi sırasında bir kişiye maruz kalmakta olduğunuz ortaya çıkar.

Anonim SSL ' yi destekleyen bir Diffie-Hellman anahtar değiş tokası CipherSpec kullanarak sunucuyu doğrulamadan şifrelenmiş bir SSL bağlantısı oluşturabilirsiniz. İstemci ile sunucu arasında paylaşılan ve SSL iletimlerini şifrelemek için kullanılan ana güvenlik dizgisi, özel olarak imzalanmış bir sunucu sertifikası değiş tokasız olarak kurulur.

Anonim bağlantılar güvensiz olduğu için, çoğu SSL somutlaması anonim CipherSpecsözelliğini kullanmak için varsayılan değer değildir. Bir telemetri kanalı tarafından SSL bağlantısı için bir istemci isteği kabul edilirse, kanala geçiş tümceciği tarafından korunan bir anahtar deposu bulunmalıdır. Varsayılan olarak, SSL uygulamaları anonim CipherSpecskullanmadığı için, anahtar deposunun istemcinin doğrulayabileceği özel olarak imzalanmış bir sertifika içermesi gerekir.

Anonim CipherSpecsdeğerini kullanırsanız, sunucu anahtar deposu var olmalıdır, ancak özel olarak imzalanmış herhangi bir sertifika içermemelidir.

Şifrelenmiş bir bağlantı kurmanın başka bir yolu da, istemciye güven sağlayıcısının yerine kendi uygulamanızı koymanız. Güvenilir sağlayıcınız sunucu sertifikasının kimliğini doğrulamazdı, ancak bağlantı şifrelenir.

MQTT istemcilerinin ve telemetri kanallarının SSL yapılandırması

MQTT istemcileri ve WebSphere MQ Telemetry (MQXR) hizmeti, SSL kullanarak telemetri kanallarını bağlamak için Java Secure Socket Extension (JSSE) olanağını kullanır. MQTT C istemcileri ve aygıtlar için WebSphere MQ Telemetry cini SSL ' yi desteklemiyor.

Telemetri kanalını ve MQTT istemcisini doğrulamak için SSL ' yi yapılandırın ve müşteriler ile telemetri kanalı arasındaki iletilerin aktarılıp şifrelenmesini şifreleyin.

SSL ' yi kullanmaya alternatif olarak, IPsec gibi bir tür Sanal Özel Ağ (VPN), bir TCP/IP bağlantısının uç noktalarını doğrular. VPN, ağ üzerinden akan her bir IP paketini şifreler. Böyle bir VPN bağlantısı kurulduktan sonra, güvenilir bir ağ kurmuyorsunuz. MQTT istemcilerini, VPN ağı üzerinden TCP/IP kullanarak telemetri kanallarına bağlayabilirsiniz.

TCP/IP üzerinden SSL protokolünü kullanmak için bir Java MQTT istemcisi ile telemetri kanalı arasında bağlantı yapılandırabilirsiniz. Güvenli olan, SSL 'yi JSSE' yi kullanmak için nasıl yapılandırmanıza bağlıdır. En güvenli yapılandırmayla başlayarak, üç farklı güvenlik düzeyi yapılandırabilirsiniz:

1. Yalnızca güvenilen MQTT istemcilerinin bağlanmasına izin verir. Bir MQTT istemcisini yalnızca güvenilir bir telemetri kanalına bağlayın. İstemciyle kuyruk yöneticisi arasındaki iletileri şifrele; bkz. “SSL kullanan MQTT istemcisi kimlik doğrulaması” sayfa 102.
2. Bir MQTT istemcisini yalnızca güvenilir bir telemetri kanalına bağlayın. İstemciyle kuyruk yöneticisi arasındaki iletileri şifrele; bkz. “SSL kullanan telemetri kanalı kimlik doğrulaması” sayfa 105.
3. İstemciyle kuyruk yöneticisi arasındaki iletileri şifrele; bkz. “telemetri kanallarında yayın gizliliği” sayfa 108.

JSSE yapılandırma parametreleri

Bir SSL bağlantısının konfigürasyonunun tanımlanını değiştirmek için JSSE parametrelerini değiştirin. JSSE yapılandırma parametreleri üç küme halinde düzenlenmiştir:

1. IBM WebSphere MQ Telemetri kanalı
2. MQTT Java istemcisi
3. JRE

IBM WebSphere MQ Explorer 'ı kullanarak telemetri kanalı parametrelerini yapılandırın. `MqttConnectionOptions.SSLProperties` öznitelide MQTT Java istemcisi değiştirgelerini ayarlayın. JRE güvenlik değiştirgelerini istemci ve sunucu üzerindeki JRE güvenlik dizinindeki dosyaları düzenleyerek değiştirin.

IBM WebSphere MQ Telemetry kanalı

WebSphere MQ Explorer 'ı kullanarak tüm telemetri kanalı SSL parametrelerini ayarlayın.

ChannelName

`ChannelName` , tüm kanallarda zorunlu bir parametredir.

Kanal adı, belirli bir kapı numarasıyla ilişkilendirilen kanalı tanımlar. MQTT istemcilerinin gruplarını yönetmenize yardımcı olacak ad kanalları.

PortNumber

`PortNumber` , tüm kanallarda isteğe bağlı bir parametredir. Varsayılan değer olarak TCP kanalları için `1883` , SSL kanalları için de `8883` değeri kullanılır.

Bu kanalla ilişkilendirilen TCP/IP kapı numarası. MQTT istemcileri, kanal için tanımlanan kapıyı belirterek bir kanala bağlanır. Kanalda SSL özellikleri varsa, istemci SSL protokolünü kullanarak bağlanmalıdır; örneğin:

```
MQTTClient mqttClient = new MqttClient( "ssl://www.example.org:8884", "clientId1");
mqttClient.connect();
```

KeyFileAd

`KeyFileAd` , SSL kanalları için gerekli bir değiştirgedir. TCP kanalları için atlanmalıdır.

`KeyFileAdı` , sağladığınız sayısal sertifikalar içeren Java anahtar deposunun yoludur. Sunucuda anahtar deposu tipi olarak JKS, JCEKS ya da PKCS12 ' yi kullanın.

Anahtar deposu tipini, aşağıdaki dosya uzantılarından birini kullanarak tanımlayın:

```
.jks
.jceks
.p12
.pkcs12
```

Diğer dosya uzantılarına sahip bir anahtar deposunun JKS anahtar deposu olduğu varsayılır.

Sunucudaki bir anahtar deposu tipini, istemcideki diğer anahtar deposu tipleriyle birleştirebilirsiniz.

Sunucunun özel sertifikasını anahtar depoya yerleştirin. Sertifika, sunucu sertifikası olarak bilinir. Sertifika, bir imzalama yetkilisi tarafından imzalanmış bir sertifika zincirinin ya da sertifika zincirinin bir parçası olabilir.

Bir sertifika zinciri kullanıyorsanız, ilişkili sertifikaları sunucu anahtar deposuna yerleştirin.

Sunucu sertifikası ve sertifika zincirindeki tüm sertifikalar, sunucunun kimliğini doğrulamak üzere istemcilere gönderilir.

`ClientAuth` öğesini `Required`olarak ayarladıysanız, anahtar deposunun istemcinin kimliğini doğrulamak için gereken sertifikaları içermesi gerekir. İstemci kendinden onaylı bir sertifika ya da bir sertifika zinciri gönderir ve istemciyi, bu malzemenin anahtar depoundaki bir sertifiya ilişkin ilk doğrulamaları ile doğrulanır. Bir sertifika zinciri kullanarak, bir sertifika, farklı istemci sertifikalarıyla yayınlansa bile, birçok istemciyi doğrulayabilir.

PassPhrase

`PassPhrase` , SSL kanalları için gerekli bir parametredir. TCP kanalları için atlanmalıdır.

Anahtar deposunu korumak için geçiş tümcecisi kullanılır.

ClientAuth

`ClientAuth` , isteğe bağlı bir SSL parametresidir. İstemci kimlik doğrulaması için varsayılan değer olarak kullanılır. TCP kanalları için atlanmalıdır.

İstemcinin telemetri kanalına bağlanmasını izin vermeden önce, telemetri (MQXR) hizmetinin istemcinin kimliğini doğrulamasına izin vermek için `ClientAuth` seçeneğini belirleyin.

`ClientAuth`seçeneğini ayarladıysanız, istemci sunucuya SSL kullanarak bağlanmalıdır ve sunucunun kimliğini doğrulamalıdır. `ClientAuth`hayarına yanıt olarak istemci, dijital sertifikasını sunucuya gönderir ve diğer sertifikalar anahtar deposuyla birlikte gönderilir. Dijital sertifikası, istemci sertifikası olarak bilinir. Bu sertifikaların kimliği, kanal anahtar deposunda ve JRE `cacerts` mağazasında tutulan sertifikalara göre doğrulanır.

CipherSuite

`CipherSuite` , isteğe bağlı bir SSL parametresidir. Varsayılan olarak, tüm etkin CipherSpecs' ı deneyecek şekilde ayarlanır. TCP kanalları için atlanmalıdır.

If you want to use a particular CipherSpec, set `CipherSuite` to the name of the CipherSpec that must be used to establish the SSL connection.

Telemetri hizmeti ve MQTT istemcisi, her bir uçta etkinleştirilen tüm CipherSpecs 'nden ortak bir CipherSpec ' i kararlaştırabiliyor. Bağlantının her iki ucunda ya da her iki ucunda belirli bir CipherSpec belirtilirse, bu, diğer uçta CipherSpec ile eşleşmelidir.

JSSE ' ye ek sağlayıcılar ekleyerek ek şifreleri kurun.

Federal Bilgi İşleme Standartları (FIPS)

FIPS, isteğe bağlı bir ayardır. Varsayılan olarak ayarlanmaz.

Kuyruk yöneticisinin özellikler panosunda ya da **runmqsc**komutunu kullanarak SSLFIPSayarlayın. `SSLFIPS` , yalnızca FIPS onaylı algoritmaların kullanılıp kullanılmayacağını belirtir.

İptal adılistesi

İptal ad listesi isteğe bağlı bir ayardır. Varsayılan olarak ayarlanmaz.

Kuyruk yöneticisinin özellikler panosunda ya da **runmqsc**komutunu kullanarak SSLCRLNL' yi ayarlayın. `SSLCRLNL` , sertifika iptal konumlarını sağlamak için kullanılan kimlik doğrulama bilgileri nesnelerinin ad listesini belirtir.

SSL özelliklerini ayarlayan başka bir kuyruk yöneticisi değiştirgesi kullanılmadı.

MQTT Java istemcisi

`MqttConnectionOptions.SSLPropertiesiçinde` Java istemcisi için SSL özellikleri ayarlayın; örneğin:

```
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);
```

Belirli özelliklere ilişkin adlar ve değerler, `MqttConnectOptions`için API belgelerinde açıklanmıştır. MQTT istemci kitaplıklarına ilişkin istemci API belgelerine ilişkin bağlantılar için bkz. MQTT istemci programlama başvurusu.

`Protokol`

`Protokol` isteğe bağlıdır.

Protokol, telemetri sunucusu ile müzakere halinde seçilir. Belirli bir protokole gereksinim duyarsanız, bir protokol seçebilirsiniz. Telemetri sunucusu, iletişim kuralını desteklemiyorsa bağlantı başarısız olur.

`ContextProvider`

`ContextProvider` isteğe bağlıdır.

`KeyStore`

`KeyStore` isteğe bağlıdır. İstemcinin kimlik doğrulamasını zorunlu kılacak sunucuda `ClientAuth` ayarlandıysa, bu değeri yapılandırın.

İstemci sayısal sertifikasını, özel anahtarını kullanarak, anahtar deposuna yerleştirin. Anahtar deposu yolunu ve parolasını belirtin. Tip ve sağlayıcı isteğe bağlıdır. JKS, varsayılan tiptir ve IBMJCE varsayılan sağlayıcısıdır.

Yeni bir anahtar deposu sağlayıcısı ekleyen bir sınıfa gönderme yapmak için farklı bir anahtar deposu sağlayıcısı belirtin. Pass the name of the algorithm used by the keystore provider to instantiate the `KeyManagerÜreticisi` by setting the key manager name.

`TrustStore`

`TrustStore` isteğe bağlıdır. JRE `cacerts` mağazasına güvendiğiniz tüm sertifikaları yerleştirebilirsiniz.

İstemci için farklı bir güvenilir deponun olmasını istiyorsanız, güvenli deponun konfigürasyonunu tanımlayın. Sunucu, önceden `cacerts`içinde saklanan kök sertifikasına sahip, iyi bilinen bir sertifika kuruluşu tarafından verilen bir sertifikayı kullanıyorsa, güvenilir deponun konfigürasyonunu tanımlayamayabilirsiniz.

Sunucunun genel olarak imzalanmış sertifikasını ya da güvenli depona kök sertifikayı ekleyin ve güvenilirlik deposu yolunu ve parolasını belirtin. JKS, varsayılan tiptir ve IBMJCE varsayılan sağlayıcısıdır.

Yeni bir güvenilirlik deposu sağlayıcısı ekleyen bir sınıfa gönderme yapmak için farklı bir güvenilirlik deposu sağlayıcısı belirtin. Güvenilirlik deposu sağlayıcısı tarafından kullanılan algoritmanın adını, güvenilirlik yöneticisi adını ayarlayarak `TrustManagerÜreticisi` ' yi somutlaştırmak için kullanın.

JRE

Java güvenliğinin hem istemci, hem de sunucu üzerindeki SSL davranışını etkileyen diğer yönleri JRE ' de yapılandırılır. Windows üzerindeki yapılandırma dosyaları *`Java Installation Directory`*`\jre\lib\security`içinde yer alıyor. IBM WebSphere MQ ile birlikte gönderilen JRE ' yi kullanıyorsanız, yol aşağıdaki çizelgede gösterildiği gibidir:

Çizelge 3. JRE SSL yapılandırma dosyaları için altyapıya göre dosya yolları	
Altyapı	**filePath**
Windows	*WMQ Installation Directory*\java\jre\lib\security
System x 32 bit içinLinux	*WMQ Installation Directory*/java/jre/lib/security
Diğer UNIX and Linux platformları	*WMQ Installation Directory*/java/jre64/jre/lib/security

Tanınmış sertifika yetkilileri

`cacerts` dosyası, tanınmış sertifika yetkililerinin kök sertifikalarını içerir. Bir güvenilirlik deposu belirtmediğiniz sürece, `cacerts` varsayılan olarak kullanılır. `cacerts` deposunu kullanıyorsanız ya da bir güvenilirlik deposu sağlamıyorsa, güvenlik gereksinimlerinizi karşılamak için `cacerts` içindeki imzalayanlar listesini gözden geçirmeniz ve düzenlemeniz gerekir.

`cacerts` programını, IBM Key Management yardımcı programını çalıştıran WebSphere MQ komutunu `strmqikm`komutunu kullanarak açabilirsiniz. Open `cacerts` as a JKS file, using the password `changeit`. Dosyayı güvenli kılmak için parolayı değiştirin.

Güvenlik sınıflarının yapılandırılması

Ek güvenlik sağlayıcıları ve diğer varsayılan güvenlik özelliklerini kaydetmek için `java.security` dosyasını kullanın.

İzinler

Kaynaklara verilen izinleri değiştirmek için `java.policy` dosyasını kullanın. `javaws.policy` , `javaws.jar`için izin verir.

Şifreleme gücü

Bazı JRES ' ler güç şifrelemesi azaltılmış. Anahtarları anahtar depolarına aktaramıyorsanız, azaltılmış güvenlik düzeyi şifreleme nedeni olabilir. Either, try starting **ikeyman** using the **strmqikm** command, or download strong, but limited jurisdiction files from IBM geliştirici setleri, Güvenlik bilgileri.

Önemli: Ülkenizde, şifreleme yazılımların başka bir ülkeye alınması, bulundurmak, kullanılması ya da yeniden dışa aktarılması konusunda sınırlamalar olabilir. Sınırlanmamış ilke dosyalarını karşıdan yüklemeden ya da karşıdan yüklemeden önce, ülkenizin yasalarını denetlemeniz gerekir. İzin verilip verilmediğini belirlemek için şifreleme yazılımını içe aktarma, bulundurma, kullanma ve yeniden ihracata ilişkin ilkeleri ve düzenlemeleri denetleyin.

İstemcinin herhangi bir sunucuya bağlanmasına izin vermek için güven sağlayıcısını değiştirin

Örnekte, bir güven sağlayıcısının nasıl ekleneceği ve bu sağlayıcının MQTT istemci kodundan nasıl başvurulacağı gösterilmektedir. Bu örnek, istemci ya da sunucu için kimlik doğrulaması gerçekleştirmez. Sonuçtaki SSL bağlantısı, kimlik doğrulaması yapılmadan şifrelenir.

Şekil 20 sayfa 112 içindeki kod parçacığı, MQTT istemcisi için `AcceptAllProviders` güven sağlayıcısını ve güvenilirlik yöneticisini ayarlar.

```
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);
```

Şekil 20. MQTT İstemcisi kod parçacığı

```
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());
    }
```

Şekil 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() };
        }
```

Şekil 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);
      }
    }
```

Şekil 23. AcceptAllX509TrustManager.java

Telemetri kanalı JAAS yapılandırması

İstemci tarafından gönderilen `Kullanıcı adı` 'nın kimliğini doğrulamak için JAAS ' ı yapılandırın.

WebSphere MQ yöneticisi, JAASile istemci kimlik doğrulaması gerektiren MQTT kanallarının konfigürasyonunu tanımlar. JAAS kimlik doğrulamasını gerçekleştirecek her kanal için bir JAAS yapılandırmasının adını belirtin. Kanallar, aynı JAAS yapılandırmasını kullanabilir ya da farklı JAAS yapılandırmalarını kullanabilir. Yapılandırmalar *WMQData directory*`\qmgrs\`*qMgrName*`\mqxr\jaas.config`içinde tanımlanır.

`jaas.config` dosyası JAAS yapılandırma adı tarafından düzenlenir. Her bir yapılandırma adı altında Oturum Açma yapılandırmalarının bir listesi yer alıyor; bkz. Şekil 24 sayfa 114.

JAAS , dört standart oturum açma modülü sağlar. Standart NT ve UNIX Oturum Açma modülleri sınırlı değerlerdir.

JndiLoginBirimi
JNDI (Java Naming and Directory Interface; Java Adlandırma ve Dizin Arabirimi) altında yapılandırılan bir dizin hizmetine karşı kimlik doğrulama gerçekleştirir.

Krb5LoginModule
Kerberos protokollerini kullanarak kimlik doğrulama gerçekleştirir.

NTLoginModule
Geçerli kullanıcı için NT güvenlik bilgilerini kullanarak kimlik doğrulama gerçekleştirir.

UnixLoginBirimi
Geçerli kullanıcı için UNIX güvenlik bilgilerini kullanarak kimlik doğrulama gerçekleştirir.

`NTLoginModule` ya da `UnixLoginModule` kullanılmasıyla ilgili sorun, telemetri (MQXR) hizmetinin, MQTT kanalının kimliği değil, `mqm` kimliği ile çalıştırıldığı. `mqm` , kimlik doğrulaması için `NTLoginModule` ya da `UnixLoginModule` ' a aktarılan kimliğdir ve istemcinin kimliği değildir.

Bu sorunu çözmek için, kendi Oturum Açma modülünüzü yazın ya da diğer standart Oturum Açma modüllerini kullanın. Örnek `JAASLoginModule.java` , WebSphere MQ Telemetryile birlikte sağlanır. Bu, `javax.security.auth.spi.LoginModule` arabiriminin bir uygulamasıdır. Kendi kimlik doğrulama yönteminizi geliştirmek için bunu kullanın.

Sağladığınız yeni LoginModule sınıflarının, telemetri (MQXR) hizmetinin sınıf yolunda olması gerekir. Sınıflarınızı, sınıf yolunda bulunan WebSphere MQ dizinlerine yerleştirmeyin. Kendi dizinlerinizi oluşturun ve telemetri (MQXR) hizmeti için tüm sınıf yolunu tanımlayın.

`service.env` dosyasında sınıf yolunu ayarlayarak telemetri (MQXR) hizmeti tarafından kullanılan sınıf yolunu genişletebilirsiniz. `CLASSPATH` büyük harfle yazılmalı ve sınıf yolu deyimi yalnızca hazır bilgiler içerebilir. CLASSPATH değişkeninde değişkenleri kullanamazsınız; örneğin, `CLASSPATH=%CLASSPATH%` yanlış. Telemetri (MQXR) hizmeti kendi sınıf yolunu (classpath) ayarlar. `service.env` içinde tanımlanan CLASSPATH bu dosyaya eklenir.

Telemetri (MQXR) hizmeti, MQTT kanalına bağlı bir istemci için `Kullanıcı adı` ve `Parola` değerini döndüren iki geri arama olanağı sağlar. The `Kullanıcı Adı` and `Parola` are set in the `MqttConnectOptions` object. `Kullanıcı Adı` ve `Parola`erişimine nasıl erişileceğini gösteren bir örnek için bkz. Şekil 25 sayfa 115 .

Örnekler

Adıconfigurationolan bir JAAS yapılandırma dosyası örneği: `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";
};
```

Şekil 24. Örnek `jaas.config` dosyası

Bir MQTT istemcisi tarafından sağlanan `Kullanıcı adı` ve `Parola` ' yı almak için kodlanmış bir JAAS oturum açma modülü örneği.

```
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;
}
```

Şekil 25. Örnek `JAASLoginModule.Login()` yöntemi

İstemci programlama kavramları

Bu bölümde açıklanan kavramlar, MQTT protocolürününün 3.1 sürümü için Java istemcisini anlamanıza yardımcı olur. The concepts complement the API documentation accompanying the package `com.ibm.micro.client.mqttv3`.

`com.ibm.micro.client.mqttv3` , MQTT sürüm 3.1 protokolünün Java uygulamaları için genel yöntemleri sağlayan sınıfları içerir. The `com.ibm.micro.client.mqttv3` package, and the accompanying packages that implement the protocol for Java SE and ME, are provided with the installation of IBM WebSphere MQ Telemetry.

Bir MQTT istemcisini geliştirmek ve çalıştırmak için, bu paketleri istemci aygıtında kopyaya ya da kurmanıza gerek vardır. Ayrı bir istemci yürütme ortamı kurmanıza gerek yoktur.

İstemciler için lisans koşulları, istemcilere bağladığınız sunucuyla ilişkilendirilir.

Java istemcisi, MQTT protocol' in 3.1 sürümünün bir başvuru uygulamasıdır. Farklı aygıt platformlarına uygun farklı dillerde kendi müşterilerinizi uygulayabilirsiniz. Ayrıntılı bilgi için MQ Telemetry Transport format and protocol belgesine bakın.

`com.ibm.micro.client.mqttv3` paketine ilişkin istemci API belgeleri, istemcinin bağlandığı sunucu hakkında herhangi bir varsayım vermez. MQTT istemci kitaplıklarına ilişkin istemci API belgelerine ilişkin bağlantılar için bkz. MQTT istemci programlama başvurusu. İstemcinin işleyişi, farklı sunuculara bağlanıldığında biraz farklı olabilir. Aşağıdaki açıklamalar, IBM WebSphere MQ teletext (MQXR) hizmetine bağlanıldığında istemcinin davranışını tanımlar.

MQTT messaging client for JavaScript ve web uygulamaları

Yakın zamana kadar, web uygulamaları programlama ve ileti sistemi uygulamaları oluşturma ayrı disiplinler olmuştur. Önceki deneyiminiz nerede yalanlar olursa olsun, JavaScript ' un ve ileti sisteminin birlikte kullanılmasında önemli avantajlar vardır. İleti sistemi uygulamanızı bir web uygulaması olarak kodladığınızda, bu uygulama herhangi bir güncel tarayıcıda çekilebilir ve çalıştırılabilir. Uygulamayı değiştirirseniz, tarayıcı her yenilendiğinde en son sürüm çekilir. Tarayıcı ayrıca güvenlik sonrasına ve iletilerin güvenilir iletimine de bakar.

Bir web uygulaması uygulama konuşlandırmasını nasıl kolaylaştırıyor?

(örneğin,) IBM WebSphere MQüzerinde geleneksel ileti sistemi uygulamalarını geliştirme ve devreye alma konusunda deneyiminiz varsa, aşağıdaki devreye alma sürecini yakından tanıyabilirsiniz:

1. Sistem denetimcisi istemci kitaplığını kurar ya da yerleştirir.
2. Sistem yöneticisi, ileti alışverişi uygulamasının son kullanıcılara dağıtılması ve yerel sistemlerde kurulu olması için ayarlar.
3. Kod değiştiğinde, sistem yöneticisi önceki adımları yineler (bu nedenle, değişiklik yönetimi karmaşıktır).

İleti sistemi uygulamanızı bir web uygulaması olarak kodladığınızda, bu konuşlandırma işlemi olur:

1. Sistem yöneticisi, web uygulamasını ve istemci kitaplığını URL ' de sunar.
2. Son kullanıcının tarayıcısı, web uygulamasını ve istemci kitaplığını birlikte çeker.
3. Kod değiştiğinde, tarayıcı yenilendiğinde güncellenen sürüm görüntülenir (bu nedenle değişiklik yönetimi basittir).

Web uygulamalarınızdaki tarayıcıdan doğrudan ileti sistemini kullanmak isteyebileceğiniz

If you have experience of programming apps in JavaScript, you might be interested to know the benefits provided by messaging systems such as IBM WebSphere MQ:

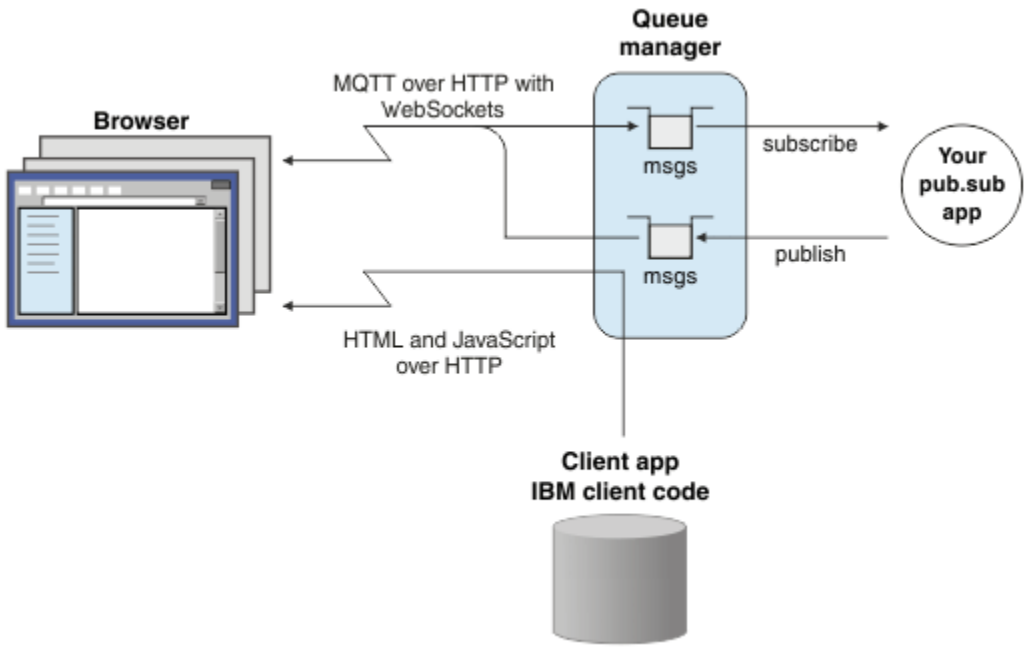
- İleti sistemi aracılığıyla ileti gönderip aldıysanız, iletilerin teslim edildiğinden emin olmak için bu sistem sorumludur.
- Mesajlaşma sistemi teslimattan sonra bakıldığı için, web uygulamanızın "ateş edip unutması" olabilir. Bu, programlama mantığını büyük ölçüde basitleştirir. İletiler sizin için teslim edildiyse, kodunuzun oraya getirilip getirilmediğini kontrol etmemeniz gerekir. Uygulamanızın gider belgesi onayını işlemesine gerek yoktur ya da teslim edilmemiş iletileri saklayıp daha sonra yeniden denemeniz gerekir.
- İleti sistemi sistemleri, olaylarla yönlendirilen ileti alışverişi sağlar. İstemci uygulamanızın artık bir istek göndermek zorunda kalmaması ve yanıt için sürekli olarak yoklama yapılması gerekmez. Bunun yerine, ileti alışverişi sunucusu, ilginç bir olay ortaya çıktığında istemci uygulamanızı bir ileti gönderir. Bunun yanı sıra, istemci uygulamanızın, uygulamanın bir sonraki anketine kadar beklemek yerine olay gerçekleşeceği anda uyarı aldığından emin olun.
- Olay odaklı ileti sistemi, istemci uygulamanızı barındıran aygıttaki yükü, tarayıcı ile ileti sistemi sunucusu arasındaki ağ trafiğini ve ileti alışverişi sunucusundaki yükü büyük ölçüde azaltır. Bu, mobil aygıtlar üzerinde daha fazla sistem çalıştırırken ve kablosuz ağlar arasında bağlantı kurmadıkça, bu durum giderek daha fazla önem çekmektedir.

Parçalar birbirine nasıl sığabiliyor?

MQTT messaging client for JavaScript , bir istemci kitaplığı ve kitaplığı kullanan bir örnek web uygulaması içerir. Kitaplığınızı kullanan kendi web uygulamanızı kodlayın. Web uygulaması ve istemci kitaplığı, seçtiğiniz URL adresinizde (örneğin, aşağıdaki şemada olduğu gibi) ya da bir uygulama sunucusu tarafından kullanılabilir kılınarak, seçtiğiniz URL ' de kullanılabilir. The browser pulls in the web app and client library, and the web app then uses the browser to connect to, and exchange messages with, an MQTT server such as IBM WebSphere MQ Telemetry or IBM MessageSight.

Bunlar akışlar:

1. Tarayıcının her bir eşgörünümü, web uygulamasının kullanılabilir olduğu URL ile bağlantısını yeniler ve web uygulamasının ve istemci kitaplığının güncel bir sürümü tarayıcıya yüklenir.
2. Web uygulaması, WebSocket protocolüzerinden MQTT kullanarak bir kuyruk yöneticisine bağlanır ve bir konuya abone olarak bir ilgi konusuna abone olur.
3. Kuyruk yöneticisi, web uygulaması aboneliğiyle eşleşen iletileri göndermek için aynı bağlantıyı kullanır.



Şekil 26. Yayınlama ve abone olma ileti alışverişi ile MQTT messaging client for JavaScript ' in kullanılması

Web uygulaması uygulama mantığı ve MQTT sunucusunun URL adresini içerir. Bir tarayıcıda açıldığında, uygulama MQTT sunucusuna bağlanır, gereken abonelikleri yaratır, ardından olay odaklı uyarıları almayı bekler ve bu abonelikler üzerinde işlem yapmak için bekler.

Web uygulaması, WebSocketsüzerinde çalışan iletim protokolü olarak MQTT ile bağlantı kurar. Çoğu modern tarayıcı WebSockets bağlantılarını gerçekleştirebilirler. Web uygulaması, WebSocketskomutunu kullanarak, HTTP ve WebSocket protocol'yi kabul eden güvenlik duvarları aracılığıyla iletileri geçirebilir ve IP üzerinden TCP' yi kullanmak gibi veri paketlerini ("çerçeveler"olarak da bilinir) gönderebilir.

Web uygulaması tarafından gönderilen bir ileti MQTT sunucusuna ulaştığında, sunucu tarafı uygulaması bunu bir ileti olarak görür. İletinin bir tarayıcıdan geldiğini bilmiyor.

MQTT sunucusunun denetlenmesi ve denetlenmesi

MQTT sunucusu, ileti sisteminin sunucu tarafı karmaşıklığını işler. Web uygulamasından aldığı iletilerin teslimi sağlar ve web uygulamasına yanıt veren yayınlama ve abone olma uygulamasını barındırır. Herhangi bir MQTT sunucusu için, aşağıdaki adımları tamamlamanız gerekir:

- Bir sunucu oluşturun.
- Bir liman seç.
- Yeni bir MQTT kanalı tanımlayın.
- Yeni MQTT kanalı içinde seçilen kapıya bağlanmak için istemci web uygulamanızı yapılandırın.

browserweb uygulaması yürütülebilir JavaScript dosyasını tarayıcıya da sunmanız gerekir. IBM WebSphere MQ Telemetrykullanıyorsanız, varsayılan olarak MQTT sunucusu, web uygulamasının MQTT sunucusuna bağlanmak için kullandığı aynı MQTT kanalını kullanarak bunu sizin için yapar. MQTT ' yi deniyorsanız, bu, hızlı bir şekilde yukarı çıkıp koşmaya yardımcı olabilir. For production use, particularly in high throughput environments, you might prefer to serve the web app executable JavaScript on a separate channel, using a dedicated application server such as WebSphere Application Server.

Not: Bu, yüksek üretilen iş ortamları için tasarlanmış olduğundan, IBM MessageSight bunu yapmayı bekler.

Örneğin, IBM WebSphere MQ Telemetrykullanıyorsanız, aşağıdaki adımları tamamlamak için IBM WebSphere MQ Explorer **New Telemetry Channel** sihirbazını kullanıyorsunuz:

1. Bir sunucu oluşturun.
2. Bir kapı seçin (varsayılan olarak 1883).
3. Yeni bir MQTT kanalı tanımlayın.
4. Yeni MQTT kanalı içinde seçilen kapıya bağlanmak için istemci web uygulamanızı yapılandırın.

The web app executable JavaScript is (optionally) also served through the queue manager on the same channel. Bunu yapmak için, kuyruk yöneticisinin hem MQTT hem de HTTP ' yi desteklemesi gerekir. MQTTiçin yapılandırılmış bir kuyruk yöneticisiniz varsa, kanal tanımlamasındaki protokolü hem MQTT hem de HTTP ' yi destekleyecek şekilde değiştirmek için MQSC komut satırı aracını kullanabilirsiniz. Bkz. ALTER CHANNEL.

Web uygulaması ve MQTT messaging client for JavaScript istemci kitaplığı, uygulama sunucunuz ya da kuyruk yöneticinizin tanımladığı bir yapıda diskteki depolanır. IBM WebSphere MQ Telemetrykullanıyorsanız, web uygulaması ve istemci kitaplığı aşağıdaki dizin yapısında depolanır:

```
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)
```

Örnek web uygulaması ve istemci kitaplığı, *MQINSTALL*/mqxr/SDK/WebContent dizininde saklanır. Bu dizindeki malzeme tüm kuyruk yöneticileri tarafından hizmet edilir. Kullanıcılarınızın bu malzemeyi görmesini ve kullanmasını istemiyorsanız, uygulamanızın kendi sürümünü oluşturmalısınız. Bu uygulamayı ya da belirli kuyruk yöneticilerindeki kendi değiştirme uygulamanızı yapmak için uygulamayı *MQINSTALL*/qmgrs/*qmgr_name*/mqxr/WebContent dizinine koymanız gerekir. Bir URL ' de hizmet vermek üzere uygulamayı ve ilişkili JavaScript sınıflarını seçmek için, kuyruk yöneticisi önce kendi WebContent dizininde, daha sonra genel WebContent dizininde görünür. Önceki örnek dizin ağacında, kuyruk yöneticisi *sizin_client_app* ve JavaScript sınıflarının genel kopyasını sunar.

Web uygulaması yürütülebilir dosyalarına hizmet veren kuyruk yöneticisini durdurmak ya da kuyruk yöneticisinin yürütülebilir dosyaları nerede aradığını değiştirmek için, **webcontentpath** özelliğini yapılandırıp mqxr.properties dosyasına ekleyin. Bkz. MQXR özellikleri.

İlgili kavramlar

“How to program messaging apps in JavaScript” sayfa 119

İlgili görevler

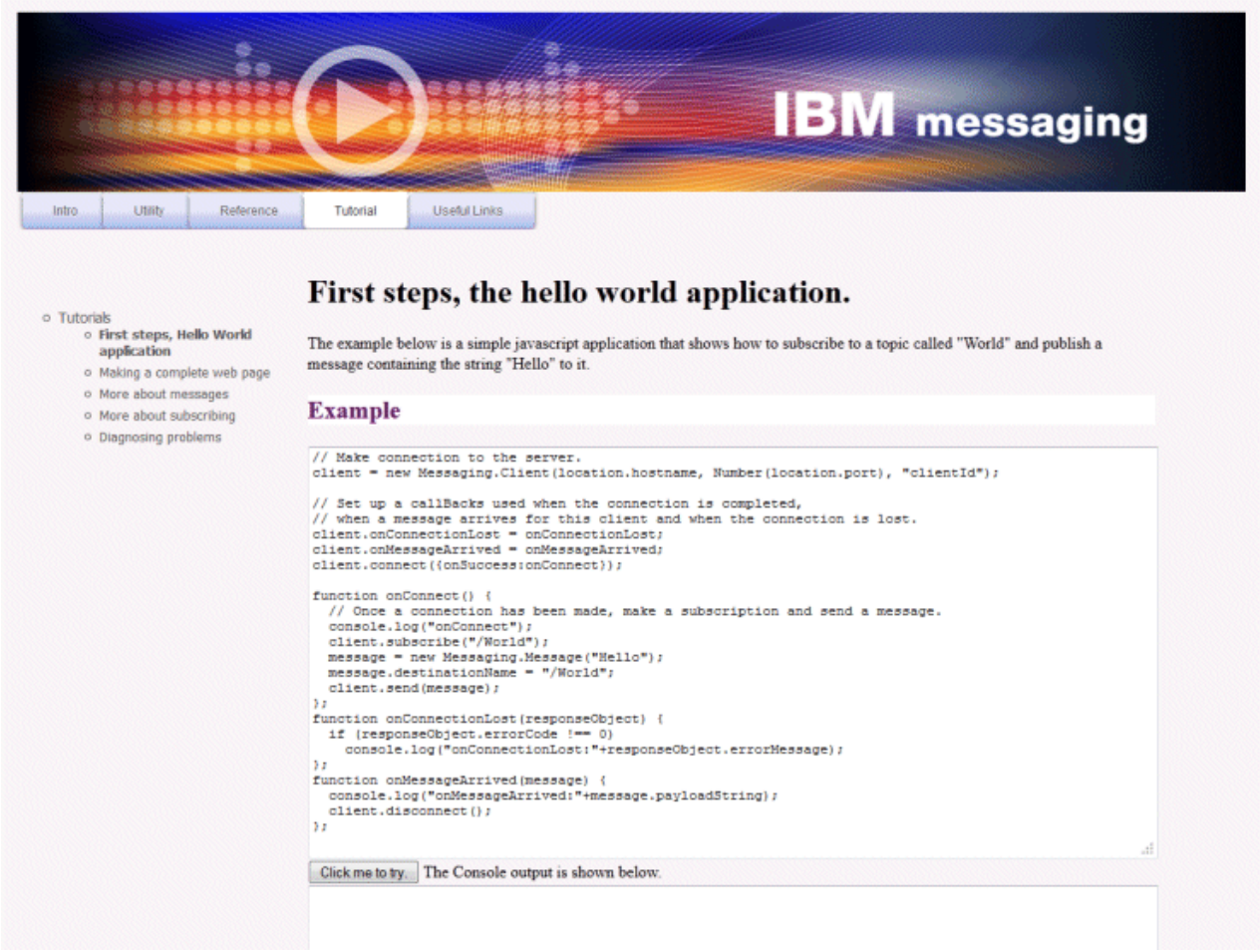
“MQTT messaging client for JavaScript ile SSL ve WebSocketsarasındaki bağlantı kurulması” sayfa 76
Connect your web app securely to IBM WebSphere MQ by using the MQTT messaging client for JavaScript sample HTML pages with SSL and the WebSocket protocol.

“MQTT messaging client for JavaScriptile çalışmaya başlama” sayfa 23
İleti alışverişi istemcisi örnek giriş sayfasını görüntüleyerek ve bağlarının bulunduğu kaynaklara göz atarak MQTT messaging client for JavaScript ile çalışmaya başlayabilirsiniz. To display this home page, you configure an MQTT server to accept connections from the MQTT ileti sistemi istemcisi örnek JavaScript sayfaları, then you type the URL that you have configured on the server into a web browser. MQTT messaging client for JavaScript otomatik olarak aygıtınızda başlar ve ileti alışverişi istemcisi örnek ana sayfası görüntülenir. Bu sayfa, yardımcı programlar, programlama arabirimi belgeleri, eğitmen ve diğer yararlı bilgiler için bağlantılar içerir.

How to program messaging apps in JavaScript

MQTT messaging client for JavaScript , basit bir yayınlama ve abone olma web uygulamasının nasıl oluşturulacağını gösteren bir eğitmeni içerir. "First steps, Hello world" uygulama kodunu keşfederek, mesajlaşma için programlama web uygulamalarının mekaniği hakkında temel bir anlayış elde edebilirsiniz.

Bugüne kadar deneyiminiz daha çok geleneksel ileti sistemi uygulamalarını geliştirmekte ve devreye alıyorsa, “JavaScript kodlama ipuçları” sayfa 119 bölümünü de yararlı bir şekilde bulabilirsiniz. İleti alışverişi için yeni olan deneyimli bir JavaScript geliştiriciyseniz, “İleti Alışverişi” sayfa 121 bölümünde önemli ileti alışverişi kavramlarına ilişkin kısa bir giriş bulabilirsiniz.



JavaScript kodlama ipuçları

İleti alışverişi uygulamalarını geliştirmekte kullanılıyorsanız, ancak web uygulamalarının yeni uygulamaları için aşağıdaki ipuçlarının yararlı olduğunu bulabilirsiniz:

onSuccess geri bildiriminde her olay için kodu sarmak

Bir ileti alışverişi uygulaması kodladığınızda, aşağıdaki olayları aşağıdaki sırayla kodlayın:

1. CONNECT
2. Abone Ol
3. yayınlamak
4. alma iletisi

MQTT messaging client for JavaScript API 'si tamamen zamanuyumsuz, bu da uygulama iş parçacıkınızın, bağlantı kurma ya da abone olma gibi çağrılar beklenirken engellenmediği anlamına gelir. Bu çağrılar yerine, bir `onSuccess` ya da `onFailure` geri çağrısı çağrılarak bu çağrıların tamamlanmasını işaret eder. To be sure that each event has completed before the next event is

triggered, you need to wrap the code for each event in an onSuccess callback. Örneğin, JavaScript uygulaması bağlantı yaratılmadan önce bağlantı çağrısını gerçekleştirmekten geri dönebilir. Abone olmadan önce bağlantının gerçekleştiğinden emin olmak için, bağlantı için bir onSuccess geri bildiriminde abone olma kodunu koymanız gerekir.

"İlk adımlar, merhaba dünya" uygulama kodu bu yaklaşımı kullanır.

Uygulama kodunun HTML marking içinde gömülmesi

Burada bir örnek JavaScript sayfası:

Aşağıda, uygulama kodunun HTML olarak nasıl yerleştirileceğini göstermek için önceki sayfanın kaynağı aşağıda yer alıyor:

```
<!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>
```

İleti Alışverişi

Aşağıda, ileti alışverişi için yeni olan web uygulama geliştiricileri için bazı arka plan ileti sistemi bilgileri yer alıyor:

Zamanuyumsuz ve yangın ve unutulan ileti alışverişi.
MQTT iletişim kuralı, güvenli teslimat ve yangın ve unutulan aktarımları destekler. Protokolde, ileti teslimi zamanuyumsuz olur: uygulama iletiyi istemci API 'sına iletir ve iletinin teslim edilmesini sağlamak için başka bir işlem olmaz. Bu yaklaşım, *ateş-ve-unutun*olarak bilinir. Bir yanıt kullanılabilir olduğunda, otomatik olarak uygulamaya gönderilir.

Zamanuyumsuz teslim, uygulamayı herhangi bir sunucu bağlantısından kurtarır ve ileti beklemekten kurtarır. Etkileşim modeli e-posta gibi, ancak uygulama programlama için optimize edilmiştir.

Ayrıca, “MQTT ' a Giriş” sayfa 5' un "MQTT iletişim kuralı" bölümüne de bakın.

Yayınlama ve abone olma ileti alışverişlerine genel bakış.
Bilgi sağlayıcıya *yayınlayıcı*adı verilir. Bir yayıncı, bu bilgilerle ilgilenen uygulamalar hakkında herhangi bir bilgi sahibi olması gerekmeksizin, bir konu hakkında bilgi sağlar. Yayınlayıcı, belirli bir konuyla ilgili

iletiler için bir kapsayıcı olan bir *konu*seçer. Daha sonra yayınlayıcı, söz konusu konu için her bir bilgi parçasını bir ileti olarak üretir, *yayın*olarak adlandırılır ve ilişkili konuya gönderir.

Bilgi tüketicisine *abone*adı verilir. Abone, ilgilendiği bir konuya ilişkin *abonelik* yaratır. Konuya yeni bir ileti gönderildiğinde, ileti konuya tüm abonelere iletilir. Aboneler birden çok aboneliği gerçekleştirebilirler ve birçok farklı yayıncıdan bilgi alabilirler.

Ayrıca bkz. IBM WebSphere MQ yayınlama/abone olma mesajlarına giriş

Abonelikler ve konular nasıl eşleşir.

If you are using IBM WebSphere MQ as your MQTT server, you need to understand how IBM WebSphere MQ specifies topics. IBM WebSphere MQ' ta bir yayınlayıcı bir ileti oluşturur ve bunu, yayın konusuna en uygun olan bir konu dizesiyle yayınlar. Yayınları almak için, bir abone, yayın konularını seçmek için bir kalıp eşleştirmesi konu dizgisiyle bir abonelik oluşturur. Kuyruk yöneticisi, yayınlara, yayın konularıyla eşleşen abonelikleri olan ve yayınları alma yetkisine sahip abonelere teslim eder.

Genellikle konular, '/' karakteri kullanılarak, konu dizgisinde alt konular yaratmak için' /' karakteri kullanılarak sıradüzenli olarak düzenlenir. Konular, konu ağacındaki düğümlerdir. Konular, başka alt konuları olmayan ya da alt konuları olan ara düğümler olan yaprak düğümler olabilir. Aboneler, bir kerede birden çok konuya abone olmak için joker karakterleri kullanabilir. Örneğin, `/sport/tennis` aboneliği yalnızca tenis alt konusuna gönderilen iletileri alır, oysa `/sport/#` aboneliği, `/sport`' un herhangi bir alt konusuna gönderilen iletileri alır.

Ayrıca bkz. Konular, Konu ağaçlarıve Genel arama karakteri şemaları.

İlgili kavramlar

“MQTT messaging client for JavaScript ve web uygulamaları” sayfa 115

İlgili görevler

“MQTT messaging client for JavaScript ile SSL ve WebSocketsarasındaki bağlantı kurulması” sayfa 76
Connect your web app securely to IBM WebSphere MQ by using the MQTT messaging client for JavaScript sample HTML pages with SSL and the WebSocket protocol.

“MQTT messaging client for JavaScriptile çalışmaya başlama” sayfa 23
İleti alışverişi istemcisi örnek giriş sayfasını görüntüleyerek ve bağlarının bulunduğu kaynaklara göz atarak MQTT messaging client for JavaScript ile çalışmaya başlayabilirsiniz. To display this home page, you configure an MQTT server to accept connections from the MQTT ileti sistemi istemcisi örnek JavaScript sayfaları, then you type the URL that you have configured on the server into a web browser. MQTT messaging client for JavaScript otomatik olarak aygıtınızda başlar ve ileti alışverişi istemcisi örnek ana sayfası görüntülenir. Bu sayfa, yardımcı programlar, programlama arabirimi belgeleri, eğitmen ve diğer yararlı bilgiler için bağlantılar içerir.

MQTT istemci uygulamalarındaki geri bildirme ve eşitleme

MQTT istemci programlama modeli iş parçacıklarını kapsamlı olarak kullanır. İş parçacıkları, bir MQTT istemci uygulamasını, sunucudan ve sunucuya gönderilen iletilerde gecikmelerden, mümkün olduğu kadar, mümkün olduğunca çok azaltan bir uygulamayı çözer. Yayınlar, teslim simgeleri ve bağlantı kaybedilen olaylar, `MqttCallback`' u gerçekleştiren bir geri çağrı sınıfındaki yöntemlere teslim edilir.

Geri Çağırmalar

`MqttCallback` arabiriminde üç geri arama yöntemi vardır; örneğin, Callback.javaiçinde örnek bir somutlama vardır.

`connectionLost(java.lang.Throwable cause)`

`connectionLost` , bir iletişim hatası bağlantının kesimine yol açınca çağrılır. Sunucu bağlantıyı, bağlantı kurulduktan sonra sunucudaki bir hatanın sonucu olarak düşerse de çağrılır. Sunucu hataları kuyruk yöneticisi hata günlüğüne kaydedilir. Sunucu, istemciye bağlantıyı atar ve istemci `MqttCallback.connectionLost`' i çağırır.

İstemci uygulamasıyla aynı iş parçacığıdaki kural dışı durumlar olarak verilen uzak hatalar, `MqttClient.connect`' ten kural dışı durumlardır. Errors detected by the server after the connection is established are reported back to the `MqttCallback.connectionLost` callback method as `throwables`.

`connectionLost` ile sonuçlanan tipik sunucu hataları, yetki hatalarıdır. Örneğin, telemetri sunucusu, konu üzerinde yayınlama yetkisi olmayan bir istemci adına bir konu üzerinde yayınlama girişiminde bulunuyor. Bir `MQCC_FAIL` durum kodunun telemetri sunucusuna döndürülmesi sonucunda ortaya çıkan herhangi bir şey bağlantının atılma durumuyla sonuçlanabilir.

`deliveryComplete(MqttDeliveryToken token)`

`deliveryComplete` , bir teslim simgesini istemci uygulamasına geri geçirmek için MQTT istemcisi tarafından çağrılır; bkz. “Teslim simgeleri” sayfa 126. Geri çağırma, teslim simgesini kullanarak yayınlanan iletiye `token.getMessage`yöntemiyle erişebilir.

Uygulama geri çağrısı, `deliveryComplete` yöntemi tarafından çağrıldıktan sonra MQTT istemcisine denetimi geri döndürdüğünde, teslim işlemi tamamlanır. Until delivery is completed, messages with QoS 1 or 2 are retained by the persistence class.

`deliveryComplete` çağrısı, uygulama ile kalıcılık sınıfı arasındaki eşitleme noktasıdır. `deliveryComplete` yöntemi, aynı ileti için hiçbir zaman iki kez çağrılamamaktadır.

Uygulama geri çağrısı `deliveryComplete` 'dan MQTT istemcisine geri döndüğünde, istemci QoS 1 ya da 2olan iletiler için `MqttClientPersistence.remove` ' i çağırır. `MqttClientPersistence.remove` , yayınlanan iletinin yerel olarak saklanan kopyasını siler.

Bir hareket işleme perspektifinden `deliveryComplete` çağrısı, teslimi kesinleten tek aşamalı bir işlemdir. If processing fails during the callback, on restart of the client `MqttClientPersistence.remove` is called again to delete the local copy of the published message. Geri arama yeniden çağrılmaz. Geri bildirme olanağını, bir teslim edilen ileti günlüğünü saklamak için kullanıyorsanız, günlüğü MQTT istemcisiyle uyumlulaştıramazsınız. If you want to store a log reliably, then update the log in the `MqttClientPersistence` class.

Teslim simgesi ve iletisine, ana uygulama iş parçacığı ve MQTT istemcisi tarafından başvuruluyor. MQTT istemcisi, teslim işlemi tamamlandığında `MqttMessage` nesnesini ve istemci bağlantısını kestiğinde teslim belirteci nesnesini kayıttan kaldırır. The `MqttMessage` object can be garbage collected after delivery is completed if the client app dereferences it. Teslim simgesi, oturumun bağlantısı kesildikten sonra çöp toplanabilir.

Bir ileti yayınlandıktan sonra `MqttDeliveryToken` ve `MqttMessage` özniteliklerini elde edebilirsiniz. İleti yayınlandıktan sonra herhangi bir `MqttMessage` özniteliğini ayarlama girişiminde bulunursanız, sonuç tanımsız olur.

İstemci, aynı `ClientIdentifier`ile önceki oturuma yeniden bağlanıyorsa, MQTT istemcisi teslim onaylarını işlemeye devam eder; bkz. “Temizleme oturumları” sayfa 125. MQTT istemci uygulaması, önceki oturum için `MqttClient.CleanSession` öğesini `false` olarak ayarlamalıdır ve yeni oturumda `false` olarak ayarlanmalıdır. MQTT istemcisi, bekleyen teslimatlar için yeni oturum için yeni teslim simgeleri ve ileti nesneleri yaratır. `MqttClientPersistence` sınıfını kullanan nesneleri kurtarır. Uygulama istemcisinin eski teslim simgeleri ve iletilerine yönelik başvuruları varsa, bu istemcilerin başvuruları kaldırılır. Uygulama geri çağrısı, önceki oturumda başlatılan ve bu oturumda tamamlanan tüm teslimatlar için yeni oturumda çağrılır.

Uygulama geri çağrısı, uygulama istemcisi bağlandıktan sonra, bekleyen bir teslim tamamlandığında çağrılır. Before the application client connects, it can retrieve pending deliveries using the `MqttClient.getPendingDeliveryTokens` method.

İstemci uygulamasının özgün olarak yayınlanan ileti nesnesini ve bilgi yükü bayt dizisini oluşturduğunu fark edin. MQTT istemcisi bu nesnelere gönderme yapar. The message object returned by the delivery token in the method `token.getMessage` is not necessarily the same message object created by the client. Yeni bir MQTT istemcisi eşgörünümü teslim simgesini yeniden yaratacaksa, `MqttClientPersistence` sınıfı `MqttMessage` nesnesini yeniden yaratır. For consistency `token.getMessage` returns `null` if `token.isCompleted` is `true`, regardless of whether the message object was created by the application client or the `MqttClientPersistence` class.

`messageArrived(MqttTopic topic, MqttMessage message)`

`messageArrived` , bir abonelik konuyla eşleşen istemci için bir yayın geldiğinde çağrılır. konu , abonelik süzgecinin değil, yayın konudur. Süzgeç joker karakterler içeriyorsa, bu ikisi farklı olabilir.

Konu, istemci tarafından yaratılan birden çok abonelikle eşleşiyorsa, istemci yayının birden çok kopyasını alır. Bir istemci, aynı zamanda abone olduğu bir konuya yayınlarsa, kendi yayınının bir kopyasını alır.

If a message is sent with a QoS of 1 or 2, the message is stored by the `MqttClientPersistence` class before the MQTT client calls `messageArrived`. `messageArrived` behaves like `deliveryComplete`: it is only called once for a publication, and the local copy of the publication is removed by `MqttClientPersistence.remove` when `messageArrived` returns to the MQTT client. The MQTT client drops its references to the topic and message when `messageArrived` returns to the MQTT client. Uygulama istemcisi nesnelere ilişkin bir başvuruya tutmadıysa, konu ve ileti nesneleri çöp toplanır.

Geri çağrılar, threading ve istemci uygulaması eşitlemesi

MQTT istemcisi, ana uygulama iş parçacığıdaki ayrı bir iş parçacığıda bir geri çağırma yöntemi çağırır. İstemci uygulaması geri çağırma için bir iş parçacığı oluşturmaz; bu, MQTT istemcisi tarafından oluşturulur.

MQTT istemcisi geri çağırma yöntemlerini uyumlulaştırır. Geri bildirme yönteminin yalnızca bir kerede tek bir eşgörünümü çalışır. Eşitleme, yayınların teslim edildiği bir nesnenin güncellenmesini kolaylaştırır. `MqttCallback.deliveryComplete` 'un bir eşgörünümü bir kerede çalışır ve bu nedenle, daha fazla eşitleme olmadan tally' yi güncelleştirmek güvenlidir. Aynı zamanda bir kerede tek bir yayının gelmesi de geçerli olur. `messageArrived` yöntemindeki kodunuz, bir nesneyi uyumlulaştırmadan güncelleyebilir. Tally ya da güncellenmekte olan nesnede başka bir iş parçacığıysa, tally ya da object nesnesini eşitleyin.

Teslim belirteci, ana uygulama iş parçacığı ile bir yayının teslimi arasında bir eşitleme mekanizması sağlar. `token.waitForCompletion` yöntemi, belirli bir yayının teslimi tamamlanıncaya kadar bekler ya da isteğe bağlı bir zamanaşımı süresi doluncaya kadar bekler. Bir yayını aynı anda işlemek için `token.waitForCompletion` ' yi birkaç basit yolla kullanabilirsiniz:

1. Yayını teslim edilinceye kadar uygulama istemcisini duraklatmak için bkz. PubSync.java.
2. `MqttCallback.deliveryComplete` yöntemiyle eşitlemek için. Yalnızca `MqttCallback.deliveryComplete` , MQTT Client 'a geri dönünce `token.waitForCompletion` devam eder. Using this mechanism you can synchronize running code in `MqttCallback.deliveryComplete` before code runs in the main application thread.

Ya her yayının teslim edilmesini beklemeden yayınlamak isteseniz, ancak tüm yayınlar teslim edildiğinde onay almak isteseniz? Tek bir iş parçacığıda yayınlarsanız, gönderilecek son yayın da son teslim edilecek son yayın olur.

Sunucuya gönderilen isteklerin eşitlenmesi

Çizelge 4. Sunucuya gelen isteklerle sonuçlanan yöntemlerin eşzamanlama davranışı.

Bu çizelge, sunucuya bir istek gönderen MQTT Java istemcisine ilişkin yöntemleri listeler. Bu çizelge, her bir yöntem için, yöntemin hangi bekleme ya da döndürdüğü ve yöntemin ne kadar süreyle bekleyeceği ile ilgili koşulları açıklar.

Yöntem	Eşitleme	Zamanaşımı aralığı
`MqttClient.Connect`	Sunucuyla bağlantı kurulması için bekler.	Varsayılan değer olarak 30 saniye ya da bir parametreye göre ayarlandığı gibi.
`MqttClient.Disconnect`	MQTT istemcisinin yapması gereken işi bitirmesini ve TCP/IP oturumunun bağlantısını kesmesini bekler.	Varsayılan değer olarak 30 saniye ya da bir parametreye göre ayarlandığı gibi.

Çizelge 4. Sunucuya gelen isteklerle sonuçlanan yöntemlerin eşzamanlama davranışı.

Bu çizelge, sunucuya bir istek gönderen MQTT Java istemcisine ilişkin yöntemleri listeler. Bu çizelge, her bir yöntem için, yöntemin hangi bekleme ya da döndürdüğü ve yöntemin ne kadar süreyle bekleyeceği ile ilgili koşulları açıklar.

(devamı var)

Yöntem	Eşitleme	Zamanaşımı aralığı
`MqttClient.Subscribe`	Abone olma isteği tamamlanana kadar bekler.	Varsayılan değer olarak 30 saniye ya da bir parametreye göre ayarlandığı gibi.
`MqttClient.UnSubscribe`	Aboneliği kaldırma isteğinin tamamlanması için bekleme işlemleri.	Varsayılan değer olarak 30 saniye ya da bir parametreye göre ayarlandığı gibi.
`MqttClient.Publish`	İsteği MQTT istemcisine ilettikten sonra uygulama iş parçacığa hemen döner.	Yok.
`MqttDeliveryToken.waitForCompletion`	Teslim simgesinin geri döndürülemesini bekler.	Varsayılan olarak ya da bir parametre tarafından ayarlandığı gibi süresiz.

Temizleme oturumları

MQTT istemcisi ve telemetry (MQXR) hizmeti, oturum durumu bilgilerini korur. Devlet bilgileri, "en az bir kez" ve "tam olarak bir kez" teslimi sağlamak için ve "tam olarak bir kez" yayınların alıntısını sağlamak için kullanılır. Oturum durumu, bir MQTT istemcisi tarafından yaratılan abonelikleri de içerir. Oturumlar arasında durum bilgilerini korumadan ya da bakım yapmadan bir MQTT istemcisi çalıştırabilir. Bağlanmadan önce `MqttConnectOptions.cleanSession` ayarını yaparak temizleme oturumu kipini değiştirin.

`MqttClient.connect` yöntemini kullanarak bir MQTT istemcisi uygulaması bağladığınızda, istemci bağlantıyı istemci tanıtıcısını ve sunucunun adresini kullanarak tanımlar. Sunucu, oturum bilgilerinin sunucuya önceki bir bağlantıdan saklanıp saklanmayacağını denetler. Önceki bir oturum hala varsa ve `cleanSession=true`, istemcideki ve sunucudaki önceki oturum bilgileri temizlenir. `cleanSession=false` önceki oturuma devam ederse. Önceki oturum yoksa, yeni bir oturum başlatılır.

Not: WebSphere MQ Yöneticisi açık bir oturumu zorlamalı olarak kapatabilir ve tüm oturum bilgilerini silebilirler. İstemci oturumu `cleanSession=false`ile yeniden açarsa, yeni bir oturum başlatılır.

Yayınlar

Varsayılan `MqttConnectOptions`değerini kullanırsanız ya da istemciyi bağlamadan önce `MqttConnectOptions.cleanSession` değerini `true` olarak ayarlıyorsanız, istemci bağlandığında istemciye ilişkin bekleyen tüm yayın teslimleri kaldırılır.

Temizleme oturumu ayarının `QoS=0`ile gönderilen yayınlarda etkisi yoktur. `QoS=1` ve `QoS=2`için, `cleanSession=true` kullanılması bir yayını kaybetmeye neden olabilir.

Abonelikler

Varsayılan `MqttConnectOptions`değerini kullanıyorsanız ya da istemciyi bağlamadan önce `MqttConnectOptions.cleanSession` değerini `true` olarak ayarlıyorsanız, istemci bağlandığında istemci için eski abonelikler kaldırılır. İstemcinin oturum sırasında yaptığı yeni abonelikler bağlantısı kesildiğinde kaldırılır.

If you set `MqttConnectOptions.cleanSession` to `false` before connecting, any subscriptions the client creates are added to all the subscriptions that existed for the client before it connected. İstemci bağlantısı kesildiğinde, tüm abonelikler etkin kalır.

`cleanSession` özniteliğinin abonelikleri etkilemesinin, bunu bir kalıcı öznitelik olarak algılamanın başka bir yolu da olabilir. In its default mode, `cleanSession=true`, the client creates subscriptions and receives publications only within the scope of the session. Alternatif modda `cleanSession=false`, abonelikler dayanıklıdır. İstemci bağlanabilir ve bağlantı kesilebilir ve abonelikleri etkin kalmaya devam eder. İstemci yeniden bağlandığında, teslim edilmemiş yayınlar alır. Bağlantı bağlıyken, kendi adına etkin olan abonelikler kümesini değiştirebilir.

Bağlanmadan önce `cleanSession` kipini ayarlamalısınız; kip tüm oturum için geçerli olur. Ayarını değiştirmek için bağlantıyı kesmeniz ve istemciyi yeniden bağlamanız gerekir. If you change modes from using `cleanSession=false` to `cleanSession=true`, all previous subscriptions for the client, and any publications that have not been received, are discarded.

İstemci tanıtıcısı

İstemci tanıtıcısı, MQTT istemcisini tanımlayan 23 baytlık bir dizgidir. İstemci tanıtıcısı, sunucuya bağlanan tüm istemcilerde benzersiz olmalıdır ve sunucuda kuyruk yöneticisi adıyla aynı olmamalıdır. Bu kısıtlar içinde herhangi bir tanımlama dizilimini kullanabilirsiniz. İstemci tanıtıcılarını ayırmak için bir yordama ve seçilen tanıtıcıya sahip bir istemcinin yapılandırılmasına ilişkin bir yordama sahip olmak önemlidir.

İstemci tanıtıcısı bir MQTT sisteminin yönetiminde kullanılır. Potansiyel olarak yüz binlerce müşteriyle birlikte, belirli bir müşteriyi hızla tanımlamayı başarmanız gerekir. Örneğin, bir aygıt arızalanır ve size bir yardım masası çalan bir müşteri tarafından bildirilir. Müşteri, aygıtı nasıl tanımlar ve genellikle istemciye bağlı olan sunucu ile bu tanıma nasıl ilişkilendirilir? Her bir aygıtı bir istemci tanıtıcısına ve bir sunucuya eşleyen bir veritabanına danışmak zorunda mısınız? Aygıtın adı, hangi sunucuya bağlı olduğunu tanımlar mı? MQTT istemci bağlantılarına göz attığınızda, her bağlantı istemci tanıtıcısıyla etiketlenir. Bir istemci tanıtıcısını fiziksel bir aygıtla eşlemek için bir çizelgeye bakmanız gerekiyor mu?

İstemci tanıtıcısı, belirli bir aygıtı, kullanıcıyı mı, yoksa istemcide çalışan bir uygulamayı mı tanımlıyor? Bir müşteri arızalı bir aygıtı yeni bir aygıtla değiştirirse, yeni aygıt eski aygıtla aynı tanıtıcıya sahip olur mu? Yeni bir tanıtıcı ayırıyor musunuz? Fiziksel bir aygıtı değiştiriyorsanız, ancak aynı tanıtıcıyı alıyorsanız, olağanüstü yayınlar ve etkin abonelikler otomatik olarak yeni aygıta aktarılır.

İstemci tanıtıcılarının benzersiz olmasını nasıl sağlardınız? Benzersiz tanıtıcılar oluşturmak için bir sistem yanı sıra, istemcide tanımlayıcıyı ayarlamak için güvenilir bir işleminiz olmalıdır. Belki de istemci aygıtı, kullanıcı arabirimi olmayan bir "kara kutu" dır. Aygıtı, MAC adresini kullanmak gibi bir istemci tanıtıcısı ile üretiyor musunuz? Ya da aygıtı etkinleştirmeden önce yapılandıran yazılım kuruluşu ve yapılandırma işleminiz var mı?

Tanıtıcıyı kısa ve benzersiz tutmak için 48 bit aygıtı MAC adresinden bir istemci tanıtıcısı oluşturabilirsiniz. İletim büyüklüğü kritik bir sorun değilse, adresi denetlemek üzere geri kalan 17 baytı kullanabilirsiniz.

Teslim simgeleri

Bir istemci bir konuda yayınlandığında yeni bir teslim belirteci oluşturulur. Bir yayının teslimini izlemek için teslim simgesini ya da teslim edilinceye kadar istemci uygulamasını engellemek için kullanın.

Simge, bir `MqttDeliveryToken` nesnesidir. It is created by calling the MqttTopic.publish() method and is retained by the MQTT client until the client session is disconnected and the delivery is completed.

Simgenin normal kullanımı, teslinin tamamlanıp tamamlanmadığını kontrol etmek için kullanılır. Teslim edilinceye kadar istemci uygulamasını engelle, döndürülen simgenin kullanılması için `token.waitForCompletion`komutunu arayın. Diğer bir seçenek olarak, bir `MqttCallBack` işleyicisi de sağlayın. When the MQTT client has received all the acknowledgments it expects as part of delivering the publication, it calls `MqttCallBack.deliveryComplete` passing the delivery token as a parameter.

Until delivery is complete, you can inspect the publication using the returned delivery token by calling `token.getMessage`.

Tamamlanan teslimatlar

Tesllerin tamamlanması zamanuyumsuz olur ve yayınla ilişkili hizmet kalitesine bağlıdır.

En çok bir kez

QoS=0
Delivery is complete immediately on return from `MqttTopic.publish`. `MqttCallback.deliveryComplete` hemen çağrılır.

En az bir kez

QoS=1
Yayın, kuyruk yöneticisinden yayınla ilgili bir alındı bildirimi alındığında tamamlanır. `MqttCallback.deliveryComplete` is called when the acknowledgment is received. İletişim yavaşsa ya da güvenilmezse, ileti `MqttCallback.deliveryComplete` çağrılmadan önce bir kereden fazla sağlanabilir.

Tam bir kez

QoS=2
Müşteri, yayının abonelere yayınlandığını belirten bir tamamlanma iletisi aldığında teslim edilir. `MqttCallback.deliveryComplete` is called as soon as the publication message is received. Bu, tamamlanma iletisinin beklemesini beklemez.

In rare circumstances, your client app might not return to the MQTT client from `MqttCallback.deliveryComplete` normally. You know that delivery has completed, because the `MqttCallback.deliveryComplete` was called. İstemci aynı oturumu yeniden başlattıysa, `MqttCallback.deliveryComplete` yeniden çağrılmaz.

Eksik teslim sayısı

İstemci oturumunun bağlantısı kesildikten sonra teslim tamamlanmazsa, istemciyi yeniden bağlayabilir ve teslimatı tamamlayabilirsiniz. İleti, `MqttConnectionOptions` özniteliği `false`değerine ayarlanmış bir oturumda yayınlandıysa, iletinin teslim edilmesini yalnızca tamamlayabilirsiniz.

Aynı istemci tanıtıcısını ve sunucu adresini kullanarak istemciyi yaratın ve daha sonra, `cleanSession` `MqttConnectionOptions` özniteliğini `false` ' e yeniden ayarlayarak bağlanın. `cleanSession` seçeneğini `true`olarak ayarlıyorsanız, bekleyen teslim simgeleri atılır.

Bekleyen teslimatın olup olmadığını denetlemek için `MqttClient.getPendingDeliveryTokens`' i arayarak denetleyebilirsiniz. İstemciyi bağlamadan önce `MqttClient.getPendingDeliveryTokens` ' u arayabilirsiniz.

Son irade ve vasiyet yayını

Bir MQTT istemci bağlantısı beklenmeyen bir şekilde sona ererse, WebSphere MQ Telemetry 'i "son bir irade ve ahit" yayını gönderecek şekilde yapılandırabilirsiniz. Yayının içeriğini ve gönderinin gönderileceği konuyu önceden tanımlayın. "Son irade ve vasiyet" bir bağlantı özellidir. İstemciyi bağlamadan önce bunu yaratın.

Son irade ve vasiyet için bir konu oluşturun. `MQTTManagement/Connections/`*`server URI`*`/`*`client identifer`*`/Lost`gibi bir konu oluşturabilirdiniz.

`MqttConnectionOptions.setWill(MqttTopic lastWillTopic, byte [] lastWillPayload, int lastWillQos, boolean lastWillRetained)` yöntemini kullanarak bir "son irade ve ahit" olarak ayarlayın.

`lastWillPayload` iletisinde bir zaman damgası yaratmayı düşünün. İstemcinin belirlenmesine ve bağlantının koşullarına yardımcı olacak diğer istemci bilgilerini de ekleyin. `MqttConnectionOptions` nesnesini `MqttClient` oluşturucusuna geçirin.

Set `lastWillQos` to 1 or 2, to make the message persistent in IBM WebSphere MQ, and to ensure delivery. Son kayıp bağlantı bilgilerini korumak için `lastWillRetained` , `true`olarak ayarlayın.

Bağlantının beklenmedik bir şekilde sona ermesi durumunda abonelere "son irade ve ahit" yayını gönderilir. It is sent if the connection ends without the client calling the `MqttClient.disconnect` method.

Bağlantıları izlemek için, bağlantıları kaydetmek ve programlanmış bağlantıları kaydetmek için diğer yayınlarla "last will and ahit" yayınını tamamlar.

Message persistence in MQTT clients

Publication iletileri, "en az bir kez" ya da "tam olarak bir kez" hizmet kalitesiyle gönderilirse, kalıcı kılınabilirler. İstemcide kendi kalıcılık mekanizmanızı uygulayabilir ya da istemciyle birlikte sağlanan varsayılan kalıcılık mekanizmasını kullanabilirsiniz. Kalıcılık, istemciye gönderilen ya da istemciden gönderilen yayınlar için her iki yönde de işlev görmektedir.

MQTT' ta ileti kalıcılığı iki yönlüdür; iletinin nasıl aktarıldığı ve bu iletinin MQTT sunucusunda kalıcı bir ileti olarak kuyruğa yollanıp kuyruklanmadığı da olabilir.

1. MQTT istemcisi çiftleri, hizmet kalitesiyle ileti kalıcılığı sağlar. İleti için seçtiğiniz hizmet kalitesine bağlı olarak, ileti kalıcı olarak yapılır. İleti sürekliliği, gereken hizmet kalitesini uygulamak için gereklidir.

   "En fazla bir kez" belirtirseniz, `QoS=0`, ileti yayınlanır yayınlanmaz iletiyi atar. İletinin yukarı işlenmesinde herhangi bir hata varsa, ileti yeniden gönderilmez. İstemci etkin olmaya devam ederse bile, ileti yeniden gönderilmez. `QoS=0` iletilerinin davranışı, IBM WebSphere MQ hızlı olmayan kalıcı olmayan iletiler ile aynıdır.

   Bir ileti, 1 ya da 2 değeri `QoS` olan bir istemci tarafından yayınlanırsa, kalıcı olarak yapılır. İleti yerel olarak saklanır ve "en az bir kez", `QoS=1`ya da "tam olarak bir kez", `QoS=2`, teslim edildiğinden emin olmak için artık gerekmediği durumlarda istemciden atılır.
2. Bir ileti `QoS` 1 ya da 2 olarak işaretlenirse, kalıcı bir ileti olarak kuyruğa alınır. `QoS=0`olarak işaretlenmişse, kalıcı olmayan bir ileti olarak kuyruğa alınır. In IBM WebSphere MQ nonpersistent messages are transferred between queue managers "exactly once", unless the message channel has the `NPMSPEED` attribute that is set to `FAST`.

Kalıcı bir yayın, istemci uygulaması tarafından alınıncaya kadar istemcide depolanır. `QoS=2`için, uygulama geri çağrısı denetimi geri döndürdüğünde, yayın istemciden atılır. `QoS=1` için, bir hata oluşursa, uygulama yayını yeniden alabilir. `QoS=0`için geri bildirme, yayını bir kereden fazla alır. Bir hata varsa ya da yayınlama sırasında istemcinin bağlantısı kesildiyse, bu yayın olmayabilir.

Bir konuya abone olduğunuzda, abonenin kalıcılık yetenekleriyle eşleşmesi için iletilerin aldığı `QoS` değerini azaltabilirsiniz. Daha yüksek bir `QoS` ' da oluşturulan yayınlar, istenen abonenin en yüksek `QoS` ile gönderilir.

İletilerin saklanması

küçük cihazlarda veri depolamanın uygulanması büyük bir anlaşmaya göre değişiklik gösteriyor. MQTT istemcisi tarafından yönetilen depolama alanındaki kalıcı iletilerin geçici olarak kaydedilmesi modeli çok yavaş olabilir ya da çok fazla depolama talep edebilir. Mobil aygıtlarda, mobil işletim sistemi, MQTT iletileri için ideal bir depolama hizmeti sağlayabilir.

Küçük aygıtların kısıtlamalarını yerine getirmede esneklik sağlamak için, MQTT istemcisinin iki kalıcılık arabirimi vardır. Arabirimler, kalıcı iletilerin depolanması içinde yer alan işlemleri tanımlar. Arabirimler, Javaiçin MQTT istemcisiiçin API belgelerinde açıklanmıştır. MQTT istemci kitaplıklarına ilişkin istemci API belgelerine ilişkin bağlantılar için bkz. MQTT istemci programlama başvurusu. Bir aygıta uyacak şekilde arabirimleri uygulayabilirsiniz. Java SE ' de çalışan MQTT istemcisi, dosya sisteminde kalıcı iletileri saklayan arabirimlerin varsayılan bir somutlamasını içerir. `java.io` paketini kullanır. The client also has a default implementation for Java ME, `MqttDefaultMIDPPersistence`.

Kalıcılık sınıfları

`MqttClientPersistence`
`MqttClientPersistence` uygulamanızın bir eşgörünümünü, `MqttClient` oluşturucusunun bir parametresi olarak MQTT istemcisine geçirin. `MqttClientPersistence`

parametresini `MqttClient` oluşturucudan çıkarırsanız, MQTT istemcisi kalıcı iletileri `MqttDefaultFilePersistence` ya da `MqttDefaultMIDPPersistence`sınıfını kullanarak saklar.

`MqttPersistable`
`MqttClientPersistence` , `MqttPersistable` nesnelerini bir depolama anahtarı kullanarak alır ve yerleştirir. You must provide an implementation of `MqttPersistable` as well as the implementation of `MqttClientPersistence` if you are not using the `MqttDefaultFilePersistence` or `MqttDefaultMIDPPersistence`.

`MqttDefaultFilePersistence`

MQTT istemcisi `MqttDefaultFilePersistence` sınıfını sağlar. İstemci uygulamanıza `MqttDefaultFilePersistence` örneğini somutlaştırırsanız, kalıcı iletilerin `MqttDefaultFilePersistence` oluşturucusunun bir parametresi olarak saklanabileceği dizini sağlayabilirsiniz.

Diğer bir seçenek olarak, MQTT istemcisi `MqttDefaultFilePersistence` ' yi örnek bir dizinde oluşturabilir ve dosyaları bir varsayılan dizine yerleştirebilir. Dizinin adı *`client identifier`*`-tcp` *`hostname portnumber`*. `"\", "\\", "/", ":"` ve `" "` , dizin adı dizgisinden kaldırılır.

Dizin yolu, `rcp.data`sistem özellkünün değeridir. `rcp.data` ayarlanmadıysa, yol `usr.data`sistem özelliğinin değeridir.

`rcp.data` , bir OSGi ya da Eclipse Rich Client Platform (RCP) kuruluşuyla ilişkilendirilmiş bir özeldir.

`usr.data` , uygulamanın başlatıldığı Java komutunun başlatıldığı dizindir.

`MqttDefaultMIDPPersistence`
`MqttDefaultMIDPPersistence` , varsayılan bir oluşturucuya sahip ve parametre yok. İletileri depolamak için `javax.microedition.rms.RecordStore` paketini kullanır.

Yayınlar

Yayınları, bir konu dizgisiyle ilişkili `MqttMessage` eşgörünümleridir. MQTT client can create publications to send to IBM WebSphere MQ, and subscribe to topics on IBM WebSphere MQ MQ to receive publications.

`MqttMessage` , bilgi yükü olarak bayt dizisine sahiptir. İletilerin mümkün olduğunca küçük olmasını hedefle. MQTT iletişim kuralı tarafından izin verilen ileti uzunluğu üst sınırı 250 MB 'dir.

Tipik olarak, bir MQTT istemci programı ileti içeriğini işlemek için `java.lang.String` ya da `java.lang.StringBuffer` kullanır. Kolaylık sağlamak amacıyla, `MqttMessage` sınıfında bilgi yükünü dizgiye dönüştürecek bir `toString` yöntemi vardır. To create the byte array payload from a `java.lang.String` or `java.lang.StringBuffer`, use the `getBytes` method.

`getBytes` yöntemi, bir dizeyi, platform için varsayılan karakter kümesine dönüştürür. Varsayılan karakter kümesi genellikle UTF-8karakteridir. Yalnızca metin içerenMQTT yayınları genellikle UTF-8ile kodlanır. Varsayılan karakter kümesini geçersiz kılmak için `getBytes("UTF8")` yöntemini kullanın.

IBM WebSphere MQ' ta, bir MQTT yayını `jms-bytes` iletisi olarak alınır. İleti, `<mqtt>`ve bir `<mqps>` klasörünü içeren bir `MQRFH2` klasörü içerir. `<mqtt>` klasörü, `clientId` ve `qos`' u içerir, ancak bu içerik gelecekte değişebilir.

Bir `MqttMessage` ' de üç ek öznitelik vardır: hizmet kalitesi, alıkonulup tutulmadığına ve yinelenip yinelenmeyeceği. Yinelenen işaret, hizmet kalitesi "en az bir kez" ya da "tam olarak bir kez" olduğunda ayarlanır. İleti daha önce gönderildiyse ve MQTT istemcisi tarafından yeterince hızlı bir şekilde onaylanmadıysa, yinelenen öznitelik `true`olarak ayarlanarak ileti yeniden gönderilir.

Yayınlama

Bir MQTT istemcisi uygulamasında bir yayın oluşturmak için bir `MqttMessage`oluşturun. Bilgi yükünü, hizmet kalitesini ve alıkonulup tutulmadığını belirleyin ve `MqttTopic.publish(MqttMessage message)` yöntemini çağırın; `MqttDeliveryToken` iade edilir ve yayının tamamlanması da zamanuyumsuz olur.

Diğer bir seçenek olarak, MQTT istemcisi bir yayın yarattığında `MqttTopic.publish(byte [] payload, int qos, boolean retained)` yöntemindeki değiştirgelerden sizin için geçici bir ileti nesnesi yaratabilir.

Yayının "en az bir kez" ya da "tam olarak bir kez" hizmet kalitesi, `QoS=1` ya da `QoS=2`varsa, MQTT istemcisi `MqttClientPersistence` arabirimini çağırır. Uygulamaya bir teslim simgesi döndürülmeden önce iletiyi depolamak için `MqttClientPersistence` ' i çağırır.

Uygulama, `MqttDeliveryToken.waitForCompletion` yöntemini kullanarak, ileti sunucuya teslim edilinceye kadar bloke etmeyi seçebilir. Diğer bir seçenek olarak, uygulama engellemeden de devam edebilir. Yayınların engellemeden teslim olup olmadığını denetlemek istiyorsanız, `MqttCallback` istemcisini MQTT istemcisiyle gerçekleştiren bir geri çağrı sınıfı örneğini kaydedin. The MQTT client calls the `MqttCallback.deliveryComplete` method as soon as the publication has been delivered. Hizmet kalitesine bağlı olarak, teslim alma işlemi `QoS=0`için hemen hemen hemen hemen hemen hemen hemen olabilir ya da `QoS=2`için biraz zaman alabilir.

Teslimlerin tamamlandıysa yoklama yapmak için `MqttDeliveryToken.isComplete` yöntemini kullanın. `MqttDeliveryToken.isComplete` değeri `false`eise, ileti içeriğini almak için `MqttDeliveryToken.getMessage` adını arayabilirsiniz. `MqttDeliveryToken.isComplete` çağrısının sonucu `true`eise, ileti atılır ve `MqttDeliveryToken.getMessage` çağrıldığında boş değerli gösterge kural dışı durumu yayınlanır. `MqttDeliveryToken.getMessage` ile `MqttDeliveryToken.isComplete`arasında yerleşik eşitleme yok.

İstemci, bekleyen teslim simgeleri almadan önce bağlantı kesilirse, istemcinin yeni bir eşgörünümü bağlanmadan önce teslim alma simgelerini sorgulayabilir. İstemci bağlanıncaya kadar, yeni teslimatlar tamamlanmaz ve `MqttDeliveryToken.getMessage`' i aramanız güvenlidir. Hangi yayınların teslim edilmediğini öğrenmek için `MqttDeliveryToken.getMessage` yöntemini kullanın. Pending delivery tokens are discarded if you connect with `MqttConnectOptions.cleanSession` set to its default value, `true`.

abone olunması

Bir MQTT abonesine göndermek üzere yayın yaratmaktan bir kuyruk yöneticisi ya da IBM MessageSight sorumludur. Kuyruk yöneticisi, bir MQTT istemcisi tarafından yaratılan bir abonelikte konu süzgecinin yayındaki konu dizgisiyle eşleşip eşleşmediğini denetler. Eşleşme tam olarak eşleşebilir ya da eşleşme joker karakterler içerebilir. Yayın, kuyruk yöneticisi tarafından aboneye iletilmeden önce, kuyruk yöneticisi yayınla ilişkili konu özniteliklerini denetler. Bir denetim konusu nesnesinin abone olma yetkisi olup olmadığını saptamak için, Genel arama karakterleri içeren bir konu dizgisini kullanarak abone olunması başlıklı konuda açıklanan arama yordamlarından sonra gelir.

MQTT istemcisi "en az bir kez" hizmet kalitesine sahip bir yayın aldığında, yayını işlemek için `MqttCallback.messageArrived` yöntemini çağırır. Yayının hizmet kalitesi "tam olarak bir kez", `QoS=2`ise, MQTT istemcisi, iletiyi alındığında iletiyi depolamak için `MqttClientPersistence` arabirimini çağırır. Daha sonra `MqttCallback.messageArrived`' u çağırır.

MQTT istemcisi tarafından sağlanan hizmet nitelikleri

Bir MQTT istemcisi, yayınları WebSphere MQ ' ya ve MQTT istemcisine teslim etmek için üç nitelikte hizmet sağlar: "en az bir kez", "en az bir kez" ve "tam olarak bir kez". Bir MQTT istemcisi abonelik oluşturmak için IBM WebSphere MQ ' e bir istek gönderdiğinde, istek, "en az bir kez" hizmet kalitesiyle gönderilir.

Bir yayının hizmet kalitesi, `MqttMessage`' nin bir öznitesidir. Bu, `MqttMessage.setQos`yöntemi tarafından ayarlanır.

`MqttClient.subscribe` yöntemi, bir konuyla ilgili olarak bir istemciye gönderilen yayınlara uygulanan hizmet kalitesini düşürebilir. Bir aboneye iletilen bir yayının hizmet kalitesi, yayının hizmet kalitesinden farklı olabilir. İki değerin alt değeri bir yayını iletmek için kullanılır.

En çok bir kez
`QoS=0`

İleti en çok bir kez teslim edilir ya da hiç teslim edilmez. Ağ üzerindeki teslimi onaylanmadı.

İleti saklanmaz. İstemcinin bağlantısı kesildiyse ya da sunucu başarısız olursa ileti kaybedilebilir.

QoS=0 , aktarma için en hızlı kiptir. bazen "ateş ve unutun" denilir.

The MQTT protocol does not require servers to forward publications at QoS=0 to a client. Sunucunun yayını aldığı süre içinde istemcinin bağlantısı kesilirse, sunucuya bağlı olarak, yayın atılabilir. Telemetri (MQXR) hizmeti, QoS=0ile gönderilen iletileri atmıyor. Bunlar, kalıcı olmayan iletiler olarak depolanır ve kuyruk yöneticisi durursa yalnızca atılır.

En az bir kez

QoS=1

QoS=1 , varsayılan aktarma kipidir.

İleti her zaman en az bir kez teslim edilir. Gönderen bir alındı bildirimi almazsa, bir alındı bildirimi alınıncaya kadar ileti, yeniden DUP işaretiyle gönderilir. Bir sonuç alıcısı olarak aynı iletiyi birkaç kez gönderilebilir ve bunu birden çok kez işleyebilirler.

İleti, işleninceye kadar gönderene ve alıcıya yerel olarak saklanmalıdır.

İleti, iletiyi işledikten sonra alıcıdan silinir. Alıcı bir aracıysa, ileti abonelerine yayınlanır. Alıcı bir istemciyse, ileti abone uygulamasına teslim edilir. İleti silindikten sonra, alıcı gönderene bir alındı bildirimi gönderir.

İleti, alıcıdan bir alındı bildirimi aldıktan sonra göndericiden silinir.

Tam bir kez

QoS=2

İleti her zaman tam olarak bir kez teslim edilir.

İleti, işleninceye kadar gönderene ve alıcıya yerel olarak saklanmalıdır.

QoS=2 , aktarım için en güvenli, ancak en yavaş kiptir. İleti göndericiden silinmeden önce, gönderen ile alıcı arasında en az iki çift iletim alır. İleti ilk iletilmeden sonra alıcıda işlenebilir.

İlk iletimler çiftinde, gönderen iletiyi iletir ve iletiyi sakladığı alıcısından alındı bildirimi alır. Gönderen bir alındı bildirimi almazsa, bir alındı bildirimi alınıncaya kadar ileti, yeniden DUP işaretiyle gönderilir.

İkinci iletimler çiftinde, gönderen, iletiyi işlemeyi tamamlayabileceğini belirtir, "PUBREL". Gönderen, "PUBREL" iletisine ilişkin bir alındı bildirimi almazsa, bir alındı bildirimi alınıncaya kadar "PUBREL" iletisi yeniden gönderilir. The sender deletes the message it saved when it receives the acknowledgment to the "PUBREL" message

İleti, iletiyi yeniden işlememesi koşuluyla, iletiyi birinci ya da ikinci aşamalarda işleyebilir. Alıcı bir aracıysa, iletiyi abonelere yayınlar. Alıcı bir istemciyse, iletiyi abone uygulamasına teslim eder. Alıcı, iletiyi işlemeyi bitirdiği bir tamamlanma iletisini gönderene geri gönderir.

Alıkonan yayınlar ve MQTT istemcileri

Alıkonan bir yayını olan bir konuya abonelik oluşturursanız, konuyla ilgili en son tutulan yayın, hemen size iletilir.

Bir konudaki yayınının alıkonulup tutulmadığını belirtmek için `MqttMessage.setRetained` yöntemini kullanın.

To delete a retained publication in IBM WebSphere MQ, run the CLEAR TOPICSTR**CLEAR TOPICSTR** MQSC command.

Boş değerli bilgi yükü ile bir yayın oluşturursanız, boş yayın abonelere iletilir. Diğer MQTT aracıları abonelere boş bir yayını iletemeyebilir.

Alıkonmayan bir yayını alıkonan bir yayınla ilgili bir konuya yayınlıyorsanız, alıkonan yayın bundan etkilenmez. Yürürlükteki aboneler yeni yayını alır. Yeni aboneler önce alıkonan yayını alır, sonra da yeni yayınlar alır.

Alıkonan bir yayını yarattığınızda ya da güncellediğinizde, yayını bir QoS ya da 1 ya da 2 ile gönderin. Bunu bir QoS /0 ile gönderdiğinizde, IBM WebSphere MQ , kalıcı olmayan bir alıkonacı yayın yaratır. Kuyruk yöneticisi durursa, yayın korunmaz.

Bir ölçümün en son değerini kaydetmek için alıkonan yayınları kullanın. Alıkonan konuya yeni aboneler hemen ölçümün en son değerini alır. Yayın konusuna son abone olan aboneden bu yana yeni ölçümler alınmazsa ve abone yeniden abone olursa, abone yine konuyla ilgili en son tutulan yayınını alır.

Abonelikler

Bir konu süzgecini kullanarak, yayın konularına ilgi kaydetmek için abonelikler oluşturun. Bir istemci birden çok abonelik ya da genel arama karakterleri kullanan bir konu süzgeci içeren bir abonelik yaratabilir ve birden çok konuya ilgi gösterebilirler. Süzgeçlerle eşleşen konularla ilgili yayınlar istemciye gönderilir. Bir istemci bağlantısı kesilirken abonelikler etkin kalabilir. Yayınlar yeniden bağlandığında istemciye gönderilir.

`MqttClient.subscribe` yöntemlerini kullanarak abonelikler oluşturun, bir ya da daha fazla konu süzgeci ve hizmet kalitesi parametreleri iletin. Hizmet kalitesi parametresi, bir iletiyi almak için abonenin kullanmak üzere hazırlamış olduğu maksimum hizmet kalitesini ayarlar. Bu istemciye gönderilen iletiler, daha yüksek hizmet kalitesiyle teslim edilemez. İleti yayınlandığında ve abonelik için belirtilen düzey olduğunda, hizmet kalitesi özgün değerin alt değerine ayarlanır. İleti almak için varsayılan hizmet kalitesi: `QoS=1`, en az bir kez.

Abonelik isteği, `QoS=1`ile birlikte gönderilir.

Publications are received by a subscriber when the MQTT client calls the `MqttCallback.messageArrived` method. `messageArrived` yöntemi, iletinin aboneye yayınlandığı konu dizesini de geçirir.

`MqttClient.unsubscribe` yöntemlerini kullanarak bir aboneliği ya da bir kümeyi ya da abonelikleri kaldırabilirsiniz.

Bir WebSphere MQ komutu, bir aboneliği kaldırabilir. List subscriptions using WebSphere MQ Explorer, or by using **runmqsc** or PCF commands. Tüm MQTT istemci abonelikleri adlandırılır. Bu formlara bir ad verilir: *`ClientIdentifier`*:*`Topic name`*

Varsayılan `MqttConnectOptions`değerini kullanıyorsanız ya da istemciyi bağlamadan önce `MqttConnectOptions.cleanSession` değerini `true` olarak ayarlıyorsanız, istemci bağlandığında istemci için eski abonelikler kaldırılır. İstemcinin oturum sırasında yaptığı yeni abonelikler bağlantısı kesildiğinde kaldırılır.

If you set `MqttConnectOptions.cleanSession` to `false` before connecting, any subscriptions the client creates are added to all the subscriptions that existed for the client before it connected. İstemci bağlantısı kesildiğinde, tüm abonelikler etkin kalır.

`cleanSession` özniteliğinin abonelikleri etkilemesinin, bunu bir kalıcı öznitelik olarak algılamanın başka bir yolu da olabilir. In its default mode, `cleanSession=true`, the client creates subscriptions and receives publications only within the scope of the session. Alternatif modda `cleanSession=false`, abonelikler dayanıklıdır. İstemci bağlanabilir ve bağlantı kesilebilir ve abonelikleri etkin kalmaya devam eder. İstemci yeniden bağlandığında, teslim edilmemiş yayınlar alır. Bağlantı bağlıyken, kendi adına etkin olan abonelikler kümesini değiştirebilir.

Bağlanmadan önce `cleanSession` kipini ayarlamalısınız; kip tüm oturum için geçerli olur. Ayarını değiştirmek için bağlantıyı kesmeniz ve istemciyi yeniden bağlamanız gerekir. If you change modes from using `cleanSession=false` to `cleanSession=true`, all previous subscriptions for the client, and any publications that have not been received, are discarded.

Etkin aboneliklerle eşleşen yayınlar, yayınlandığı anda istemciye gönderilir. İstemcinin bağlantısı kesildiyse, aynı istemci tanıtıcısı ve `MqttConnectOptions.cleanSession` ile `false`değerine ayarlanmış aynı sunucuya yeniden bağlanıyorsa, istemci istemciye gönderilir.

Belirli bir istemciye ilişkin abonelikler istemci tanıtıcısı tarafından tanımlanır. İstemciyi farklı bir istemci aygıtından aynı sunucuya yeniden bağlayabilir ve aynı aboneliklere devam edebilir ve teslim edilmemiş yayınlarını alabilirsiniz.

Topic strings and topic filters in MQTT clients

Konu dizgileri ve konu süzgeçleri yayınlamak ve abone olmak için kullanılır. MQTT istemcilerinde konu dizgilerinin ve süzgeçlerin sözdizimi büyük ölçüde IBM WebSphere MQiçindeki konu dizgileriyle aynıdır.

Konu dizgileri, abonelere yayın göndermek için kullanılır. `MqttClient.getTopic(java.lang.String topicString)` yöntemini kullanarak bir konu dizgisi yaratın.

Konu süzgeçleri konu başlıklarına abone olmak ve yayınları almak için kullanılır. Konu süzgeçleri joker karakterler içerebilir. Genel arama karakterleriyle birden çok konuya abone olabilirsiniz. Bir abonelik yöntemi kullanarak bir konu süzgeci yaratın; örneğin, `MqttClient.subscribe(java.lang.String topicFilter)`.

Konu dizgileri

Bir IBM WebSphere MQ konu dizgisinin sözdizimi, Konu Dizgileri' nde açıklanmaktadır. MQTT konu dizgilerinin sözdizimi, Javaiçin MQTT istemcisi. MQTT istemci kitaplıklarına ilişkin istemci API belgelerine ilişkin bağlantılar için bkz. MQTT istemci programlama başvurusu.için API belgelerindeki `MqttClient` sınıfında açıklanmıştır.

Her konu dizgisinin sözdizimi hemen hemen aynı. Dört küçük fark vardır:

1. Topic strings sent to IBM WebSphere MQ by MQTT clients must follow the convention for queue manager names. Özellikle, konu dizgilerinde kısa çizgi karakterleri bulunamaz.
2. Uzunluk üst sınırı farklıdır. IBM WebSphere MQ konu dizgileri 10.240 karakterle sınırlıdır. Bir MQTT istemcisi, 65535 byte 'a kadar konu dizgileri yaratabilir.
3. MQTT istemcisi tarafından yaratılan bir konu dizgisi boş değerli karakter içeremez.
4. WebSphere Message Broker 'da boş bir konu düzeyi `'...//...'` geçersizdi. Boş konu düzeyleri IBM WebSphere MQtarafından desteklenmektedir.

IBM WebSphere MQ yayınlama/abone olma gibi, `mqttv3` iletişim kuralının bir denetim konusu nesnesi kavramı yoktur. Bir konu nesnesinden ve konu dizgisinden bir konu dizgisi oluşturamazsınız. Ancak, bir konu dizgisi WebSphere MQ' daki bir yönetim konularıyla eşlenir. Yönetimle ilgili konu ile ilişkilendirilen erişim denetimi, bir yayının konuya yayınlanıp yayınlanmadığını ya da atılıp atılmadığını belirler. Abonelere iletildiğinde bir yayına uygulanan öznitelikler, denetim konularının özniteliklerinden etkilenir.

Konu süzgeçleri

Bir IBM WebSphere MQ konu süzgecinin sözdizimi, Konu tabanlı genel arama şeması' nde açıklanmıştır. Bir MQTT istemcisiyle oluşturabileceğiniz konu süzgeçlerinin sözdizimi, Javaiçin MQTT istemcisiiçin API belgelerindeki `MqttClient` sınıfında açıklanmıştır. MQTT istemci kitaplıklarına ilişkin istemci API belgelerine ilişkin bağlantılar için bkz. MQTT istemci programlama başvurusu.

Her konu süzgecinin sözdizimi hemen hemen aynıdır. Tek fark, farklı MQTT aracılarının bir konu süzgecini yorumlamandır. WebSphere Message Broker V6' da, çok düzeyli joker karakter yalnızca bir konu süzgecinin sonunda kullanılabilir. WebSphere MQ' da, çok düzeyli bir genel arama karakteri, konu ağacındaki herhangi bir düzeyde kullanılabilir; örneğin, `USA/#/Dutchess County`.

MQTT istemci programlama başvurusu

Mobil İleti Sistemi ve M2M Client Pack' a ve ilişkili istemci API belgelerine yönelik bağlantılar burada bulunur.

Mobil İleti Sistemi ve M2M Client Pack' de MQTT istemci kitaplıkları, oluşturulan API belgeleriyle birlikte paketlenir. İstemci paketini IBM ileti sistemi topluluğu karşıdan yüklemelerindenyükleyebilirsiniz.

Eclipse Paho projesine aşağıdaki bağlantıları izleyerek en yeni API belgelerinin çevrimiçi kopyalarını görebilirsiniz:

- Java sınıfları için MQTT istemcisi
- C için MQTT istemci kitaplığı

- C için zamanuyumsuz MQTT istemci kitaplığı

Not:

1. MQTT Java uygulamalarını `com.ibm.micro.client.mqttv3`yerine `org.eclipse.paho.client.mqttv3` paketiyle bağlantılayın. gösterir. The `com.ibm.micro.client.mqttv3` package is provided to support existing MQTT Java applications.
2. V 7.5.0.1 C için MQTT istemci uygulamalarını, `MQTTClient` kitaplığı yerine `MQTTAsync` kitaplığına bağlayın. `MQTTClient` , C için var olan MQTT uygulamalarını desteklemek için sağlanır.
3. MQTT messaging client for JavaScript , WebSockets' u destekleyen bir MQTT sunucusu gerektirir. Örneğin, IBM WebSphere MQ Version 7.5 ve sonraki sürümler bunu yapar.

MQTT Server sunucuları ile çalışmaya başlama

MQTT aktarım iletişim kuralını destekleyen ileti alışverişi sunucuları IBM ve diğerleri tarafından kullanılabilir. The most basic MQTT server enables mobile apps and devices, supported by MQTT client libraries, to exchange messages. IBM WebSphere MQ ve IBM MessageSight , IBMsunucularından MQTT sunucularıdır. Temel MQTT sunucuları olarak işlev görmenin yanı sıra, MQTT istemci uygulamaları ile kurumsal uygulamalar arasında da ileti alışverişi vardır. IBM ' daki tüm MQTT sunucuları, MQTT version 3.1 iletişim kuralını destekler ve WebSocket protocolüzerinden MQTT .

Yürürlükteki MQTT Server from IBM

IBM WebSphere MQ

- IBM WebSphere MQ , kurumsal düzeyde ileti alışverişi sağlar. Telemetri bileşeni, IBM WebSphere MQ ' in de bir MQTT sunucusu olarak işlev görmesine olanak sağlar.
- Bu, mobil, makineden makineye (M2M) ve aygıt tabanlı uygulamalarınızı destekler ve ayrıca, IBM WebSphere MQ ve JMS uygulamaları gibi kurumsal ileti sistemi uygulamalarıyla ileti alışverişi yapılmasına da olanak sağlar.
- IBM WebSphere MQ kuruluşu, MQTT SDK ' nin IBMolanağından bir kopyasını içerir. Bu SDK, örnek MQTT istemci uygulamaları ve bu uygulamaları destekleyen MQTT istemci kitaplıkları sağlar.

  Not: Bu SDK ' nın en güncel sürümünü almak için, Mobil İleti Sistemi ve M2M Client Packdosyasını karşıdan yükleyin. Daha fazla bilgi için “MQTT istemcilerini kullanmaya başlama” sayfa 9 başlıklı konuya bakın.
- MQTT desteği ilk olarak IBM WebSphere MQ Version 7.0.1içinde yer aldı. IBM WebSphere MQ' un her bir yayın düzeyine ilişkin tam bilgi edinmek için aşağıdaki ürün belgelerine bakın:
  - WebSphere MQ Telemetry Sürüm 7.5
  - WebSphere MQ Telemetry Sürüm 7.1

  IBM WebSphere MQ' e kısa bir giriş ve IBM WebSphere MQ Telemetry bileşeniyle çalışmaya başlama adımları için bkz. “MQTT sunucusu olarakIBM WebSphere MQ” sayfa 136.

IBM MessageSight

- IBM MessageSight , çok sayıda MQTT istemcisi ile aynı anda bağlantı kurabilen ve aynı zamanda büyüyen çok sayıda mobil aygıt ve algılayıcıların barınması için gerekli performansı ve ölçeklenebilirliği sağlayan, aygıt tabanlı bir MQTT sunucudur. It supports the MQTT version 3.1 protocol, and MQTT over the WebSocket protocol.

- The main features and benefits of IBM MessageSight as an MQTT server are as follows:
  - Yüksek performans, güvenilirlik ve ölçeklenebilir ileti sistemi.
  - Eşzamanlı olarak bağlantılı uç noktaları için büyük toplulukları destekleyerek özellikle makineden makineye (M2M) ve Nesnelerin İnterneti senaryolarına özel olarak tasarlanmıştır.
  - Kurulum ve kullanım kolaylığı. Bu işlem, 30 dakika altında çalışır durumda olabilir.
  - Android ve iOSiçinde yer alan yerel mobil uygulamalar için destek.
  - Yayınlama/abone olma aracısı olarak IBM WebSphere MQ ile bütünleştirme.
- IBM MessageSight' e hızlı bir giriş yapmak için bkz. YouTubeile ilgili MessageSight tanıtımı ve MessageSight duyurusu. Ayrıntılı teknik bilgiler için bkz. MessageSight ürün belgeleri.

IBM WebSphere MQ Telemetry daemon for devices

- Bu, IBM WebSphere MQ Telemetry advanced client for Colarak da bilinir. Bu, genellikle ağ kenarının yakınındaki uydu konumlarında ya da aygıtlarda çalışan, küçük bir alan kaplayan bir MQTT sunucudur; örneğin, en üst kutular, uzaktan telemetri birimleri ya da satış noktası uçbirimleri gibi.
- Bunun için tipik bir kullanım,lots'a tek bir MQTT bağlantısı üzerinden IBM WebSphere MQ ' e bağlı olan birçok MQTT istemci bağlantısını yoğunlaştırabilmekte. Örneğin, bir binaya çok sayıda algılayıcı yerleştirebilir, bunları IBM WebSphere MQ Telemetry daemon for devices'e bağlayabilir ve cini IBM WebSphere MQ' e bağlayabilir.
- The IBM WebSphere MQ Telemetry daemon for devices is included with IBM WebSphere MQ. A separate license is required to connect it to IBM WebSphere MQ. Bkz. IBM ABD Yazılım Duyurusu 212-091.

Really Small Message Broker

- Really Small Message Broker (RSMB), IBM WebSphere MQ Telemetry daemon for devices' nin bir sürüsüdür. Ana fark kullanımda. RSMB, IBM alphaWorks' tan edinilebilir ve MQTT tabanlı çözümleri değerlendirirken veya deneylerken kullanılmak üzere hazırlanmış küçük bir test sunucudur. RSMB supports MQTT on a number of Linux platforms, on Windows XP, on Apple Mac OS X Leopard, and on Unslung (Linksys NSLU12)

IBM' dan önceki MQTT sunucuları

WebSphere Message Broker *(şimdi IBM Integration Bus olarak bilinir)*

- WebSphere Message Broker Sürüm 6, kendi MQTT sunucusunu sağladı. Destek, IBM WebSphere MQ' un telemetri bileşeni tarafından WebSphere Message Broker Sürüm 7 'de değiştirilmiştir.

Diğer MQTT sunucuları

MQTT.org , açık kaynak sunucuları da dahil olmak üzere, Hizmet olarak sunulan sayfasındaki MQTT sunucularının ve aracılarının bir listesini tutar.

İlgili görevler

“MQTT istemcilerini kullanmaya başlama” sayfa 9
MQTT istemci kitaplığını kullanan bir örnek MQTT istemci uygulaması oluşturarak ve çalıştırarak bir mobil ya da makineden makineye (M2M) uygulaması geliştirmeye başlayabilirsiniz. Örnek uygulamalar ve ilişkili istemci kitaplıkları, IBMiçindeki Mobil İleti Sistemi ve M2M Client Pack içinde bulunur. Java, JavaScriptve C dillerinde yazılan uygulamaların ve istemci kitaplıklarının sürümleri vardır. Bu uygulamaları, Apple' den Android aygıtları ve ürünleri de dahil olmak üzere birçok platformda ve aygıtta çalıştırabilirsiniz.

MQTT sunucusu olarakIBM WebSphere MQ

IBM WebSphere MQiçinde yer alan MQTT sunucusunu kullanmaya ilişkin giriş.

Başlamak için aşağıdaki makalelerdeki adımları izleyin:

- “kurmaIBM WebSphere MQ” sayfa 136
- “Configuring the MQTT service from the command line” sayfa 138
- “Configuring the MQTT service with IBM WebSphere MQ Explorer” sayfa 140

Not: Komut satırı arabirimi örneğini kullanarak hızlı bir şekilde çalışmaya başlayabilirsiniz. Ancak, yapılanışınız örnek olarak önemli ölçüde farklıysa, komut satırı arabirimini etkili bir şekilde kullanmak için daha fazla bilgi ve beceriye gereksinim duyarsınız. Hem başlangıç olarak hem de standart yapılandırma görevlerini kolayca yapmak için IBM WebSphere MQ Explorer arabirimini kullanın.

IBM WebSphere MQ Telemetry bileşeniyle ilgili temel kavramsal bilgiler için IBM WebSphere MQ ürün belgelerindeki aşağıdaki makalelere bakın:

- Telemetri aygıtlarının kuyruk yöneticisine bağlanması
- Telemetri (MQXR) hizmeti
- Telemetri kanalları

İlgili bilgiler

Linux ve AIXüzerinde telemetri için kuyruk yöneticisi yapılandırılması
Windows üzerinde telemetri için kuyruk yöneticisi yapılandırılması
MQTT istemcilerine ileti göndermek için dağıtımlı kuyruklama yapılandır
WebSphere MQ TelemetryYönetimi

kurmaIBM WebSphere MQ

Follow these instructions to obtain and install IBM WebSphere MQ and configure IBM WebSphere MQ Telemetry on Windows or Linux.

Başlamadan önce

IBM WebSphere MQüzerinde çalışan MQTT hizmeti tarafından desteklenen işletim sistemleri için bkz. IBM WebSphere MQ Telemetri sistemi gereksinimleri.

Aşağıdaki yöntemlerden birini kullanarak IBM WebSphere MQ kuruluş malzemelerinin ve lisansın bir kopyasını alın:

1. Kuruluş malzemeleri için IBM WebSphere MQ yöneticinize sorun ve lisans sözleşmesini kabul edebilir misiniz?
2. IBM WebSphere MQ' un 90 günlük değerlendirme kopyasını alın. Bkz. Değerlendirme: IBM WebSphere MQ.
3. Buy IBM WebSphere MQ. Bkz. IBM WebSphere MQ ürün sayfası.

Bu görev hakkında

Install IBM WebSphere MQ as `root` on Linux, and as an administrator on Windows. At install time, select the additional options `Telemetri Hizmeti` and `Telemetri Müşterileri` to install the IBM

WebSphere MQ Telemetry component. IBM WebSphere MQ' u denetlemek için bir kullanıcı kimliği oluşturun ve konuk kullanıcı kimliğinin tanımlı olup olmadığını denetleyin. The guest user ID is used in the sample MQTT service configuration to authorize MQTT access to IBM WebSphere MQ.

After installing IBM WebSphere MQ, start the MQTT service by doing the steps in “Configuring the MQTT service from the command line” sayfa 138 or “Configuring the MQTT service with IBM WebSphere MQ Explorer” sayfa 140.

Yordam

1. Log on as `root` on Linux, or as an administrator on Windows.
2. IBM WebSphere MQ' yi kurun.

   Follow the instructions in Installing WebSphere MQ server on Linux or Windows üzerinde WebSphere MQ sunucusunun kurulması. IBM WebSphere MQ Telemetry bileşenini kurmak için `Telemetri Hizmeti` ve `Telemetri İstemcileri` seçeneğini belirleyin.

   Linux' ta, kuruluş birincil ürününüzü yapmak için "Sıradaki işlem" bölümündeki yönergeyi not alın. Bu kuruluş iş istasyonunuzdaki tek IBM WebSphere MQ kuruluşu olsa bile, birincil iş istasyonunu birincil kullanıcı yapın. Bkz. Single installation of WebSphere MQ Version 7.1, or later, configured as the primary installation.

   Örnek yapılandırma yönergelerini tam olarak izlemek için, kuruluşu birincil olarak yapmanız gerekir.

   Birden çok kuruluş: Birincil olmayan bir kuruluşla çalışmak istiyorsanız, setmqenv komutunu çalıştırın. Bu, iş istasyonunuzdaki bir komut penceresinde IBM WebSphere MQ ortamını ayarlar. Bkz. Birden çok kuruluş.

   Kuruluş programı tarafından sunulan varsayılan kuruluş konumunu kabul etmiş olduğunuz varsayılarak, IBM WebSphere MQ aşağıdaki dizinlerde kurulu olmalıdır:

   Linux 64 bit

   ```
   /opt/mqm
   ```

   Windows 32 bit

   ```
   C:\Program Files\IBM\WebSphere MQ
   ```

   Windows 64 bit

   ```
   C:\Program Files (x86)\IBM\WebSphere MQ
   ```

   Kuruluş dizini *MQ_INSTALLATION_PATH*olarak gösterilir.
3. İsteğe bağlı: Add the user you are going to administer IBM WebSphere MQ with to the `mqm` group on this workstation.

   This step is optional on Windows because you can administer IBM WebSphere MQ as a Windows administrator. Bkz. UNIX ve Windows sistemlerinde WebSphere MQ yönetimi yetkisi.

   Windows iş istasyonunuz bir etki alanının üyesiyse, Varsayılan değer olmayan Windows 2000 etki alanı ya da Windows 2003 ve Windows Server 2008 etki alanı varsayılan, güvenlik izinleribaşlıklı konuya bakın.

   On Linux, the installation program creates a user `mqm`, as a member of the group `mqm`. Bu kullanıcıya bir parola girin ya da birincil grubu olarak `mqm` ile başka bir kullanıcı yaratın.
4. İsteğe bağlı: Sign on with the user that you made a member of the `mqm` group.

   This step is optional on Windows because you can administer IBM WebSphere MQ as a Windows administrator.
5. Konuk kullanıcı kimliğinin iş istasyonunuzda tanımlandığından emin olun.

   The guest user ID is `"guest"` on Windows and `"nobody"` on Linux. Konuk kullanıcı kimliği, işletim sistemi izinlerini ya da haklarını gerektirmez.

Sonuçlar

Birincil IBM WebSphere MQ kuruluşu olarak iş istasyonunuza IBM WebSphere MQ ' yi kurdun ve `mqm`grubunu oluşturdun. Kuruluş programı, IBM WebSphere MQ ürününü `mqm` grubunun üyelerine yönetme izni verir. Members of the administrators group on Windows also have authority to administer IBM WebSphere MQ.

Sonraki adım

1. Configure the MQTT service from the command line or IBM WebSphere MQ Explorer; see “Configuring the MQTT service with IBM WebSphere MQ Explorer” sayfa 140 or “Configuring the MQTT service from the command line” sayfa 138.
2. Android, iOS, WebSockets, Javave "C" MQTT istemcilerinizi test edin.
3. When you finish testing, remove the queue manager and MQTT service by running the command *MQ_INSTALLATION_PATH*`\mqxr\samples\CleanupMQM.bat` on Windows and *MQ_INSTALLATION_PATH*`/mqxr/samples/CleanupMQM.sh` on Linux.

İlgili bilgiler

WebSphere MQ Telemetrykuruluyor

Installing WebSphere MQ server on Linux

Windows üzerinde WebSphere MQ sunucusunun kurulması

Configuring the MQTT service from the command line

Örnek IBM WebSphere MQ Telemetry uygulamalarını çalıştırmak için komut satırını kullanarak IBM WebSphere MQ bu yönergeleri izleyin. Bu adımlarda, `MQXR_SAMPLE_Q`Madlı yeni bir kuyruk yöneticinde MQTT hizmeti yaratmak için komut dosyasının nasıl çalıştırılacağı gösterilmektedir.

Başlamadan önce

MQTT hizmetini ayarlamak için bir IBM WebSphere MQ kuyruk yöneticisine yönetici erişiminiz olmalıdır. Bir kuyruk yöneticisine erişim elde etmek için birkaç yönteminiz vardır:

1. IBM WebSphere MQ ' un bir kopyasını alın ve kendi Linux ya da Windows iş istasyonunuzda bir kuyruk yöneticisi yaratın. IBM WebSphere MQdosyasını edinmek ve kurmak için “kurmaIBM WebSphere MQ” sayfa 136 içindeki yönergeleri izleyin. Kuruluş sırasında `Telemetry Service` ve `Telemetry Clients` öğelerini de seçmeniz gerektiğini göz önünde bulundurun. Bu seçenekleri eklemek için, var olan bir kuruluşu da değiştirebilirsiniz.
2. Bir IBM WebSphere MQ yöneticisiyle iletişim kurun ve bir seçenek olarak IBM WebSphere MQ Telemetry kurulu bir sunucuda kuyruk yöneticisine yönetici erişimi isteyin. V 7.5.0.1 Kuyruk yöneticisi adına ek olarak, MQTT için en az iki TCP/IP kapısı ve WebSocketsüzerinden MQTT için gereklidir. Güvenli istemcilere bağlanmayı planlıyorsanız, en az iki bağlantı noktası gereklidir.

Görevdeki adımları tam olarak açıklandığı gibi yapmak için, `MQXR_SAMPLE_Q`Madlı bir kuyruk yöneticisi yaratabilmeli ve 1883 numaralı TCP/IP bağlantı noktası kullanılmamalıdır.

Bu görev hakkında

Bu görevde, kuyruk yöneticisi yaratan bir komut dosyası çalıştırıyorsunuz ve daha sonra, 1883 numaralı bağlantı noktasında MQTT V3.1 istemcisi bağlantılarını dinlemek için MQTT hizmetini yapılandırıyorsunuz. Yapılandırma, herkese herhangi bir konuya abone olma ve abone olma izni verir. Güvenlik ve erişim denetimi yapılandırması en alt düzeydir ve yalnızca kısıtlı erişime sahip güvenli bir ağda bulunan bir kuyruk yöneticisi için tasarlanmıştır. IBM WebSphere MQ ve MQTT ' yi güvenli olmayan bir ortamda çalıştırmak için güvenliği yapılandırmanız gerekir. IBM WebSphere MQ ve MQTTgüvenliğini yapılandırmak için, bu görevin sonundaki ilgili bağlantılara bakın.

Yordam

1. Log on with a user ID that has administrative authority to IBM WebSphere MQ.

   Denetim yetkisi bulunan bir kullanıcı kimliğini IBM WebSphere MQolarak tanımlamak için, “kurmaIBM WebSphere MQ” sayfa 136içindeki 3. adıma bakın.
2. Bir komut penceresi açın ve MQXR_SAMPLE_QM adlı örnek kuyruk yöneticisini ve MQTT hizmetini yaratmak ve başlatmak için örnek komut komut dosyasını çalıştırın.

   Örnek komut dosyasının yolu, Windows üzerinde %MQ_FILE_PATH% \mqxr\samples\SampleMQM.bat ve Linuxüzerinde *MQ_INSTALLATION_PATH*/mqxr/samples/ SampleMQM.sh olur.

   Kuyruk yöneticisini yaratmak ve yapılandırmak için aşağıdaki komutu yazın:

   - Windows

     ```
     "%MQ_FILE_PATH%\mqxr\samples\SampleMQM.bat"
     ```

   - Linux

     ```
     MQ_INSTALLATION_PATH/mqxr/samples/SampleMQM.sh
     ```

Sonuçlar

Örnek, Windowsüzerinde şu özelliklerle PlainText adında bir MQTT kanalı yaratır:

```
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
```

Linux üzerindeki kanal özellikleri, com.ibm.mq.MQXR.UserName=nobodyile aynı Windowsile aynıdır.

MQTT V3.1 clients that connect to port 1883 access IBM WebSphere MQ with the user ID set in the variable *com.ibm.mq.MQXR.UserName*. Örnek komut dosyası, kullanıcı kimliğine aşağıdaki IBM WebSphere MQ komutlarıyla yetkilendirir:

```
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
```

İlk komut, kullanıcı yetkisini temel konu başlığından devralan konulara abone olma ve bu konulara abone olma yetkisi verir. İkinci komut, kullanıcıya SYSTEM.MQTT.TRANSMIT.QUEUE iletim kuyruğuna ileti koymak için gereken yetkisi verir. The MQTT service sends messages on the SYSTEM.MQTT.TRANSMIT.QUEUE as publications to MQTT subscribers.

Komut dosyası, 1883 numaralı bağlantı noktasındaki bağlantıları dinlemek için kuyruğun üzerinde MQTT hizmetini başlatır.

Sonraki adım

Örnek MQTT V3.1 Java uygulamasını çalıştırarak bağlantıyı test etmek için bu adımları izleyin.

Örnek Java uygulaması için kaynak MQTTV3Sample.java dosyasında yer alıyor.

Örneği çalıştırmak için iki komut penceresi gereklidir. Örneği bir penceredeki bir abone olarak ve diğer pencerede yayınlayıcı olarak çalıştırın.

- Windows Aboneyi başlatmak için komutu çalıştırın.

  ```
  "%MQ_FILE_PATH%\mqxr\samples\RunMQTTV3Sample.bat" -a subscriber
  ```

Yayınlamak için şu komutu çalıştırın:

```
"%MQ_FILE_PATH%\mqxr\samples\RunMQTTV3Sample.bat"
```

- Linux Aboneyi başlatmak için komutu çalıştırın.

```
MQ_INSTALLATION_PATH/mqxr/samples/RunMQTTV3Sample.sh -a subscriber
```

Yayınlamak için şu komutu çalıştırın:

```
MQ_INSTALLATION_PATH/mqxr/samples/RunMQTTV3Sample.sh
```

Yayıncı ve abonenin komut pencerelerine çıkış yazılması:

```
Connected to tcp://localhost:1883
Publishing to topic "MQTTV3Sample/Java/v3" qos 2
Disconnected
Press any key to continue . . .
```

Şekil 27. Yayınlayıcıdan çıkış

```
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
```

Şekil 28. Aboneden çıkış

Sunucu, MQTT V3.1 uygulamanızı sınmanız için hazırdır.

İlgili görevler

MQTT hizmetini WebSphere MQ Explorer ile yapılandırma

Örnek IBM WebSphere MQ Telemetry istemcilerini çalıştırmak için IBM WebSphere MQ Explorer komutunu kullanarak IBM WebSphere MQ konfigürasyonunu tanımlamak için bu yönergeleri izleyin. Bu adımlarda, `Define sample` yapılanış sihirbazını çalıştırarak MQTT hizmetini nasıl yaratacağını gösterir.

İlgili bilgiler

WebSphere MQ Telemetry

WebSphere MQ Telemetryiçin uygulama geliştirilmesi

WebSphere MQ TelemetryYönetimi

WebSphere MQ Telemetry güvenliği

Configuring the MQTT service with IBM WebSphere MQ Explorer

Örnek IBM WebSphere MQ Telemetry istemcilerini çalıştırmak için IBM WebSphere MQ Explorer komutunu kullanarak IBM WebSphere MQ konfigürasyonunu tanımlamak için bu yönergeleri izleyin. Bu adımlarda, `Define sample` yapılanış sihirbazını çalıştırarak MQTT hizmetini nasıl yaratacağını gösterir.

Başlamadan önce

MQTT hizmetini ayarlamak için bir IBM WebSphere MQ kuyruk yöneticisine yönetici erişiminiz olmalıdır. Bir kuyruk yöneticisine erişim elde etmek için birkaç yönteminiz vardır:

1. IBM WebSphere MQ ' un bir kopyasını alın ve kendi Linux ya da Windows iş istasyonunuzda bir kuyruk yöneticisi yaratın. IBM WebSphere MQdosyasını edinmek ve kurmak için “kurmaIBM WebSphere MQ” sayfa 136 içindeki yönergeleri izleyin. Kuruluş sırasında `Telemetry Service` ve `Telemetry Clients` öğelerini de seçmeniz gerektiğini göz önünde bulundurun. Bu seçenekleri eklemek için, var olan bir kuruluşu da değiştirebilirsiniz.

2. Bir IBM WebSphere MQ yöneticisiyle iletişim kurun ve bir seçenek olarak IBM WebSphere MQ Telemetry kurulu bir sunucuda kuyruk yöneticisine yönetici erişimi isteyin. V 7.5.0.1 Kuyruk yöneticisi adına ek olarak, MQTT için en az iki TCP/IP kapısı ve WebSocketsüzerinden MQTT için gereklidir. Güvenli istemcilere bağlanmayı planlıyorsanız, en az iki bağlantı noktası gereklidir.

Görevdeki adımları tam olarak açıklandığı gibi yapmak için, `MQXR_SAMPLE_QM`adlı bir kuyruk yöneticisi yaratabilmeli ve 1883 numaralı TCP/IP bağlantı noktası kullanılmamalıdır.

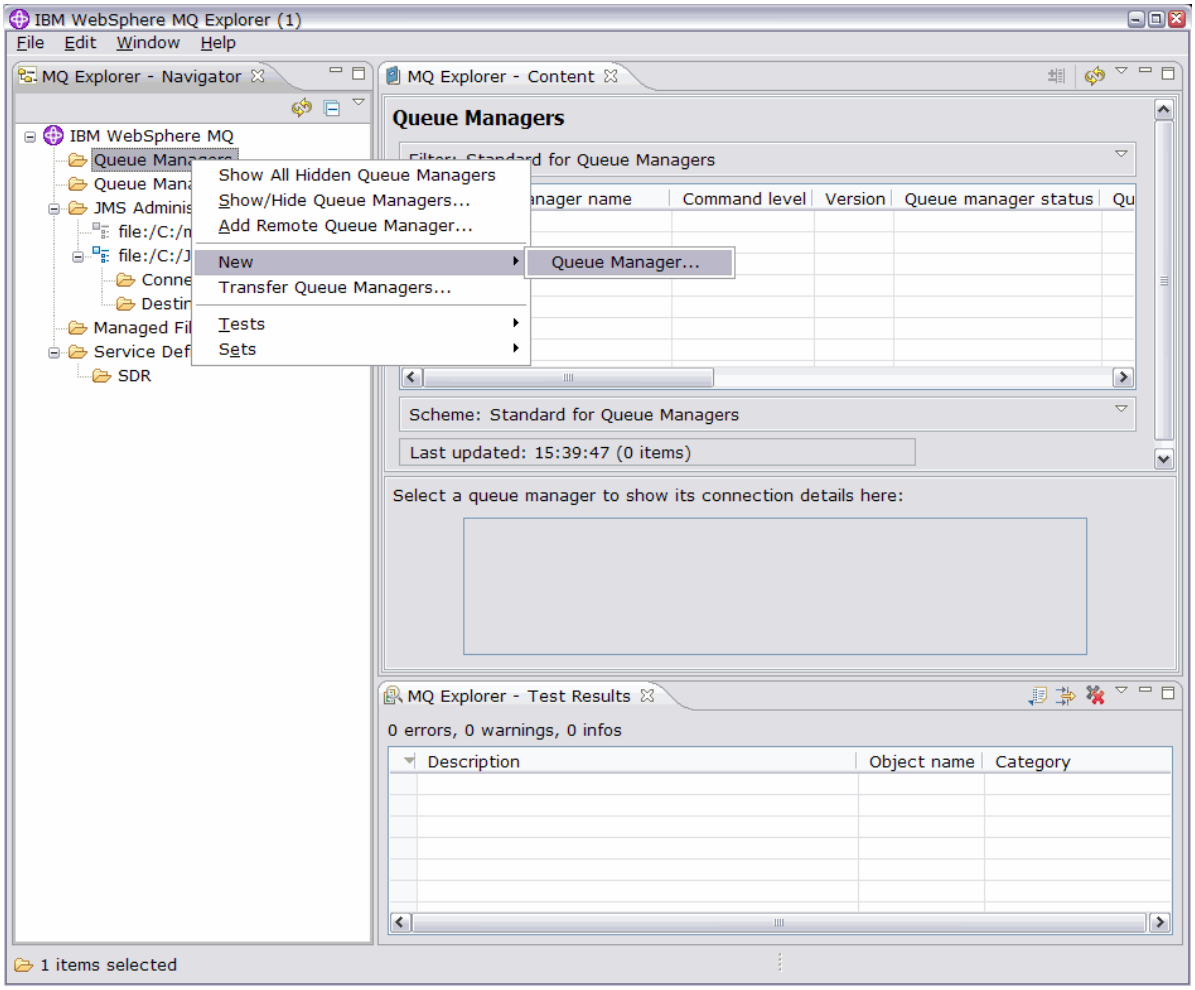
Bu görev hakkında

Bu görevde, 1883 numaralı bağlantı noktasında MQTT V3.1 istemci bağlantılarını dinlemek üzere bir MQTT hizmeti yaratmak için IBM WebSphere MQ Explorer `Define sample` yapılandırma sihirbazını çalıştırıyorsunuz. Yapılandırma, herkese herhangi bir konuya abone olma ve abone olma izni verir. Güvenlik ve erişim denetimi yapılandırması en alt düzeydir ve yalnızca kısıtlı erişime sahip güvenli bir ağda bulunan bir kuyruk yöneticisi için tasarlanmıştır. IBM WebSphere MQ ve MQTT ' yi güvenli olmayan bir ortamda çalıştırmak için güvenliği yapılandırmanız gerekir. IBM WebSphere MQ ve MQTTgüvenliğini yapılandırmak için, bu görevin sonundaki ilgili bağlantılara bakın.

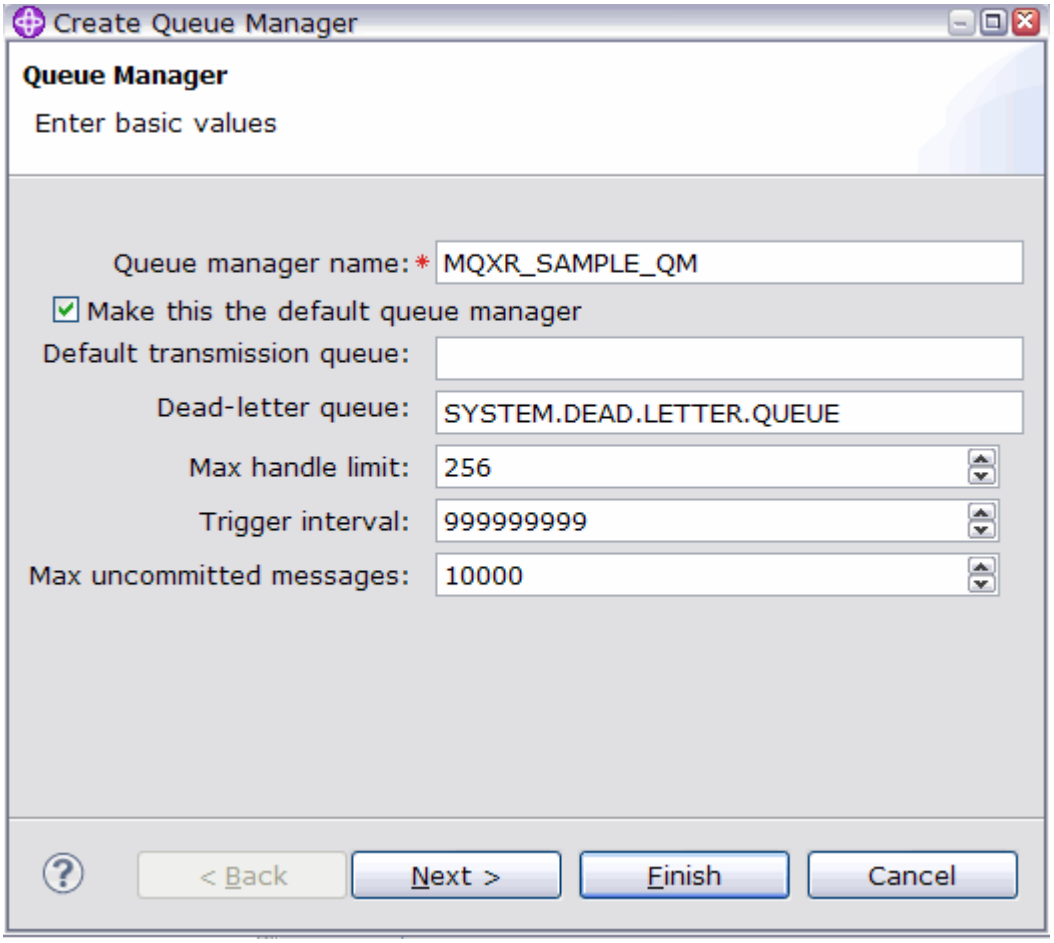
Yordam

1. Log on with a user ID that has administrative authority to IBM WebSphere MQ.

   Denetim yetkisi bulunan bir kullanıcı kimliğini IBM WebSphere MQolarak tanımlamak için, “kurmaIBM WebSphere MQ” sayfa 136içindeki 3. adıma bakın.
2. Open a command window, and run the IBM WebSphere MQ Explorer command **strmqcfg** to start IBM WebSphere MQ Explorer.
3. Kuyruk yöneticisi yarat
   a) **Yeni Kuyruk Yöneticisi** sihirbazını başlat

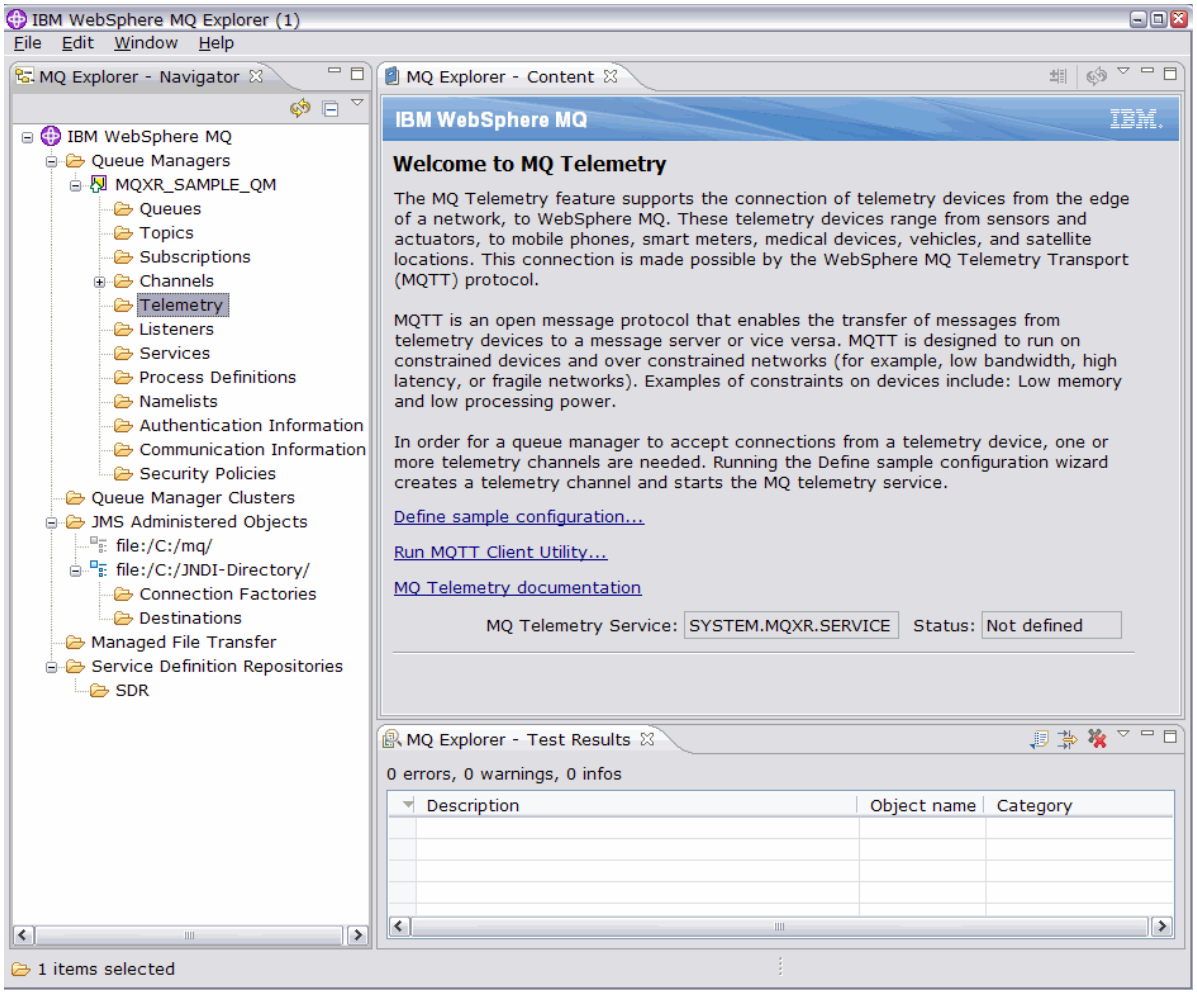


b) Bir **Kuyruk yöneticisi adı**yazın ve **Dead-letter queue**(Ölmeyen iletiler kuyruğu) adını yazın. Kolaylık sağlamak için, bunu varsayılan kuyruk yöneticisi yapın. **Bitir**'i tıklatın.



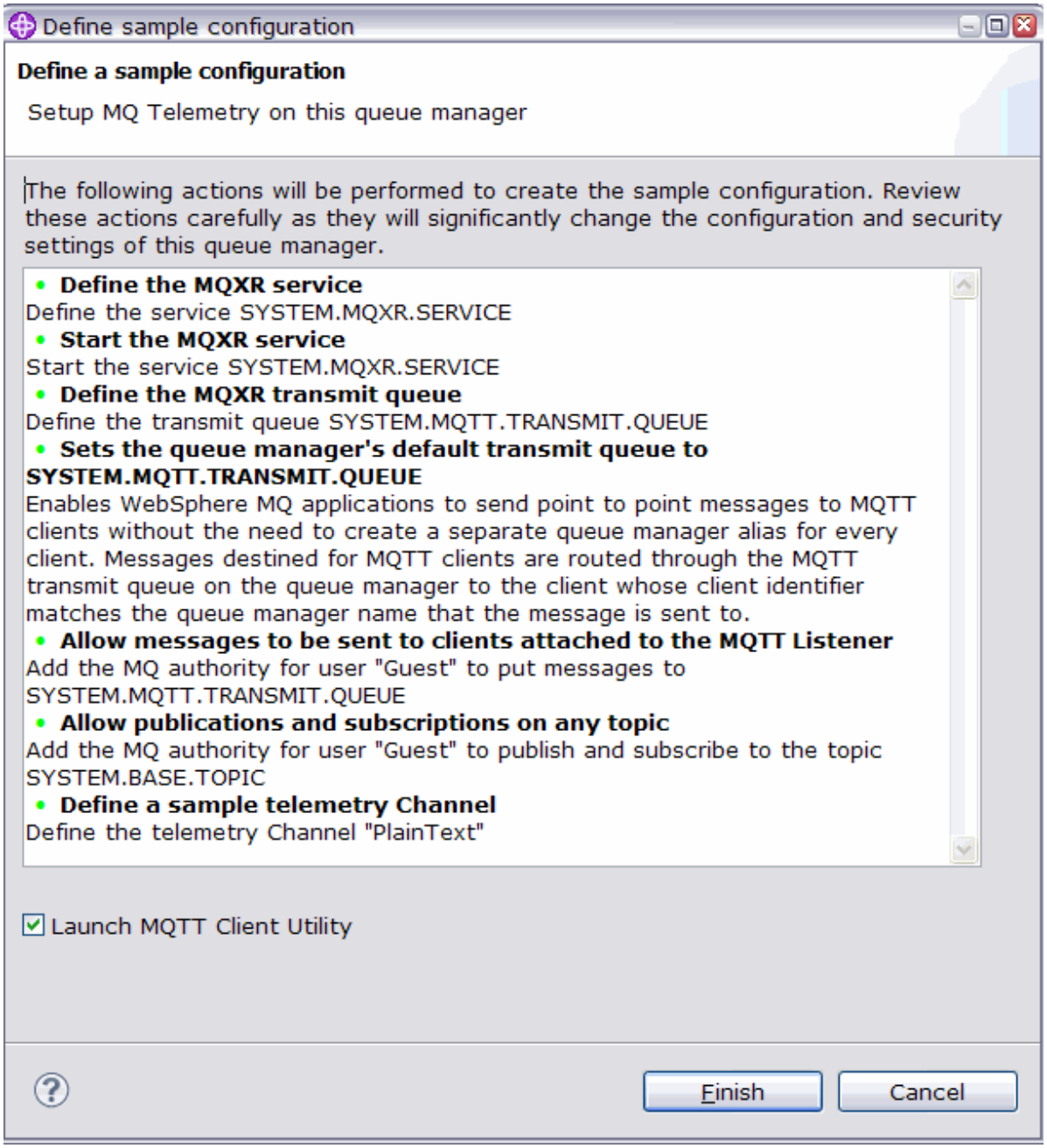
IBM WebSphere MQ Explorer kuyruk yöneticisini yaratır ve başlatır.

4. Telemetry **Örnek yapılandırma tanımla** sihirbazını çalıştırın.
   a) Kuyruk yöneticisi için Telemetry klasörünü açın.



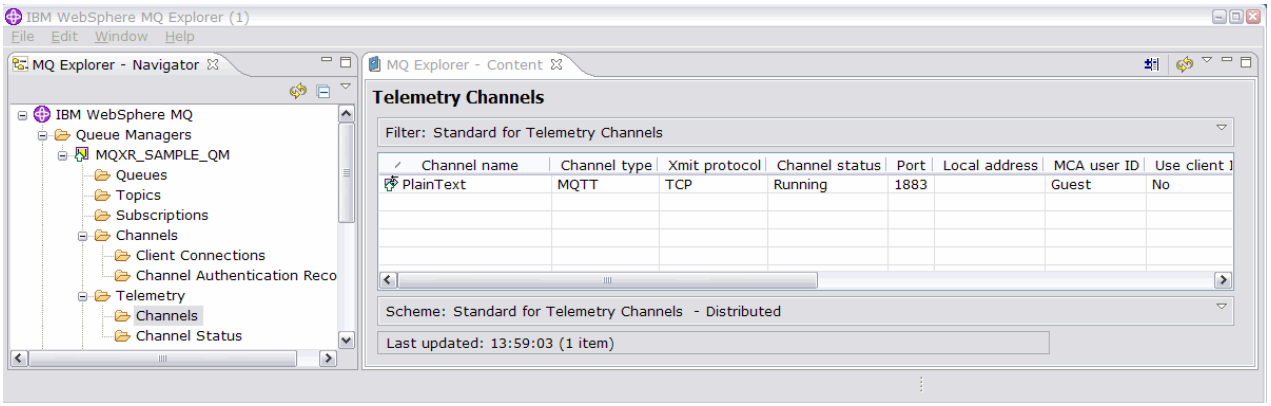
b) Sihirbazı başlatmak için **Örnek yapılandırma tanımla** öğesini tıklatın.

c) Telemetri hizmetini yaratmak ve MQTT Client Utility programını çalıştırmak için **Son** düğmesini tıklatın.



Sonuçlar

Örnek kanallarını listelemek için Telemetri Kanalları klasörünü açın.



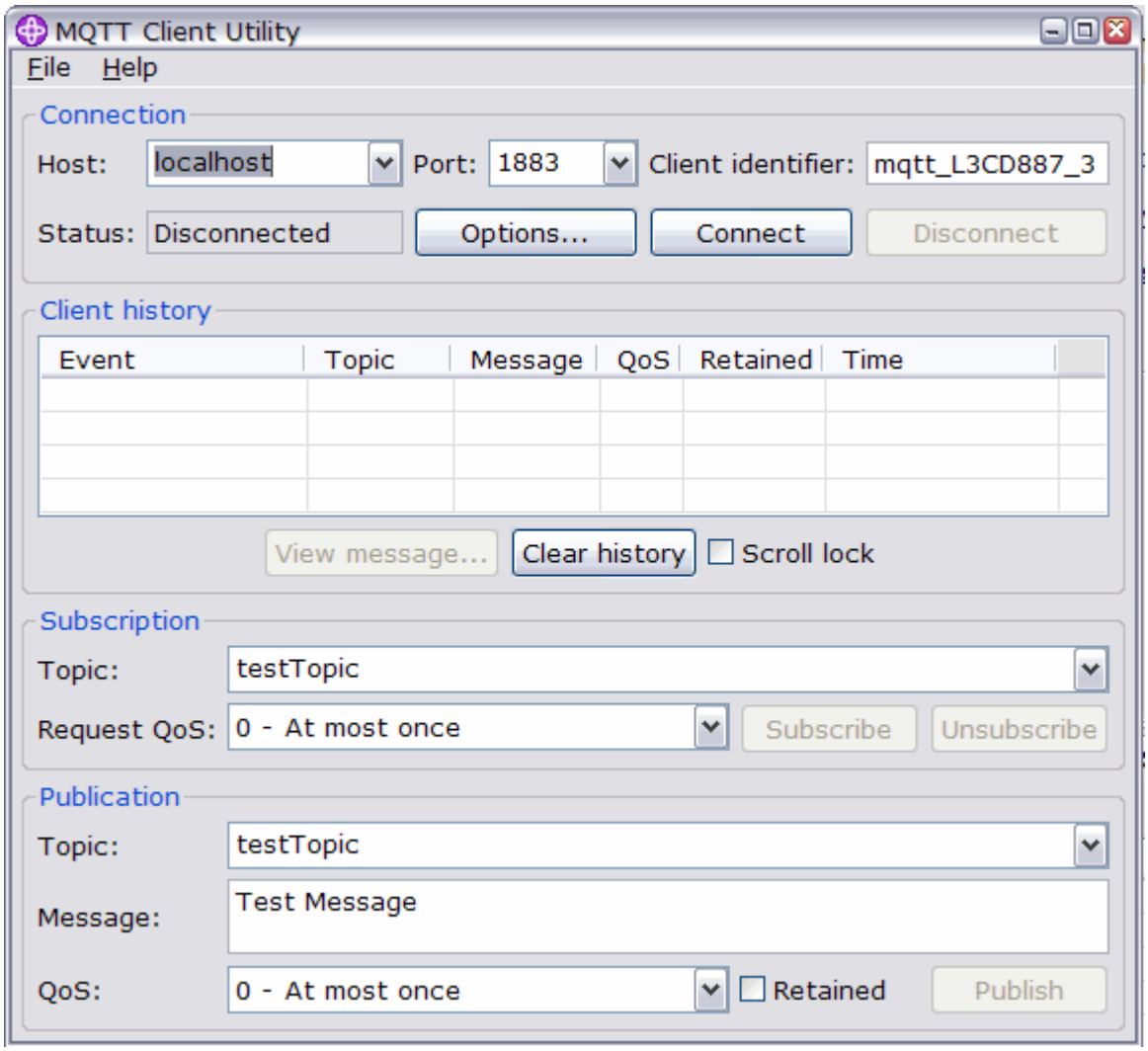
Bu kanalın özelliklerini değiştirebilir ve bu pencereye kanal ekleyebilir ve bu pencerelerde kanal silebilirsiniz.

Sonraki adım

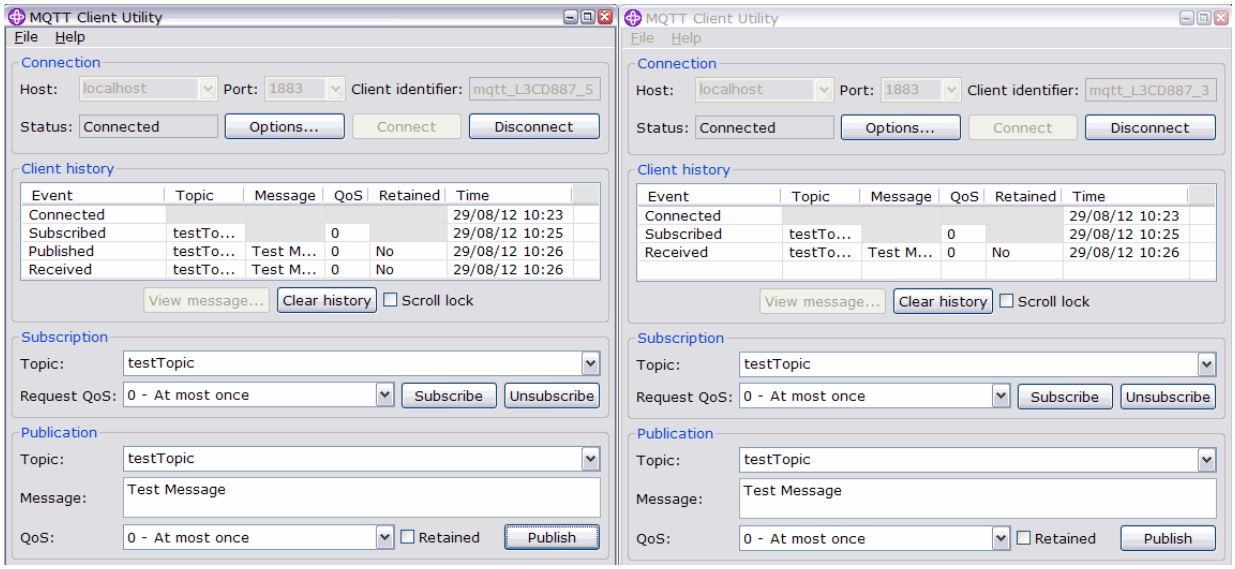
MQTT Client Utility programını çalıştırarak bağlantıyı test edin.

1. İstemci yardımcı programını başlatmak için, **Telemetry** klasörünü açın ve **MQTT Client Utility Prograsını Çalıştır** seçeneğini iki kez tıklatın.

   İki **MQTT İstemci Yardımcı Programı** penceresi açık, aynı, ancak farklı istemci tanıtıcıları için.



2. Her iki pencerede de **Bağlan** seçeneğini tıklatın.
3. Her iki pencerede **Abone Ol** ' u tıklatın.
4. Her iki pencerede **Yayınla** ' yı tıklatın. Sonuçlar Şekil 29 sayfa 147içinde gösterilir.



Şekil 29. Sonuçlar

5. Her iki pencerede **Bağlantıyı Kes** seçeneğini tıklatın.

Sunucu, MQTT V3.1 uygulamanızı sınmanız için hazırdır.

İlgili görevler

MQTT hizmetinin komut satırından yapılandırılması

Örnek IBM WebSphere MQ Telemetry uygulamalarını çalıştırmak için komut satırını kullanarak IBM WebSphere MQ bu yönergeleri izleyin. Bu adımlarda, MQXR_SAMPLE_QMadlı yeni bir kuyruk yöneticinde MQTT hizmeti yaratmak için komut dosyasının nasıl çalıştırılacağı gösterilmektedir.

WebSphere MQ TelemetryYönetimi

İlgili bilgiler

WebSphere MQ Telemetry

WebSphere MQ Telemetry olanağını WebSphere MQ Explorer ile yönetme

WebSphere MQ Telemetryiçin uygulama geliştirilmesi

Güvenlik

WebSphere MQ Telemetry güvenliği

Aygıtlar kavramları içinIBM WebSphere MQ Telemetry cini

Aygıtlar için IBM WebSphere MQ Telemetry cini gelişmiş bir MQTT V3 istemcisi uygulamasıdır. İletileri diğer MQTT istemcilerinden saklamak ve iletmek için kullanın. Bu, IBM WebSphere MQ ile bir MQTT istemcisi gibi bağlanır, ancak diğer MQTT istemcilerini de bu istemciye bağlayabilirsiniz.

Yardımcı program bir yayınlama/abone olma aracısıdır. MQTT V3 istemcileri, yayınlamak ve konuları yayınlamak için konu dizgileri ve abone olmak için konu süzgeçlerini kullanarak, konulara abone olur ve bu istemcilere abone olabilirler. The topic string is hierarchical, with topic levels divided by /. Konu süzgeçleri, tek düzeyli + genel arama karakterlerini ve çok düzeyli bir # genel arama karakterini, konu dizgisinin son bölümü olarak içerebilen konu dizgileridir.

Not: Yardımcı program için genel arama karakterleri, WebSphere Message Broker 'ın daha kısıtlayıcı kuralları (v6) izler. IBM WebSphere MQ farklı. Birden çok düzeyli birden çok genel arama karakteri destekler; genel arama karakterleri, sıradüzenin herhangi bir sayıdaki düzeyi için, konu dizgisinin herhangi bir yerinde durabilir.

Birden çok MQTT v3 istemcisi, dinleyici kapısını kullanarak yardımcı programa bağlanır. Varsayılan dinleyici kapısı değiştirilebilir. Birden çok dinleyici kapısı tanımlayabilir ve bunlara farklı ad alanları ayırabilrsiniz, bkz. “Aygıtlar dinleyici kapıları içinWebSphere MQ Telemetry cini” sayfa 155. Yardımcı programın kendisi bir MQTT v3 istemcisi olabilir. Yardımcı programı başka bir yardımcı programın dinleyici kapısına ya da bir WebSphere MQ Telemetry (MQXR) hizmetine bağlamak için bir yardımcı program köprüsü bağlantısı yapılandırın.

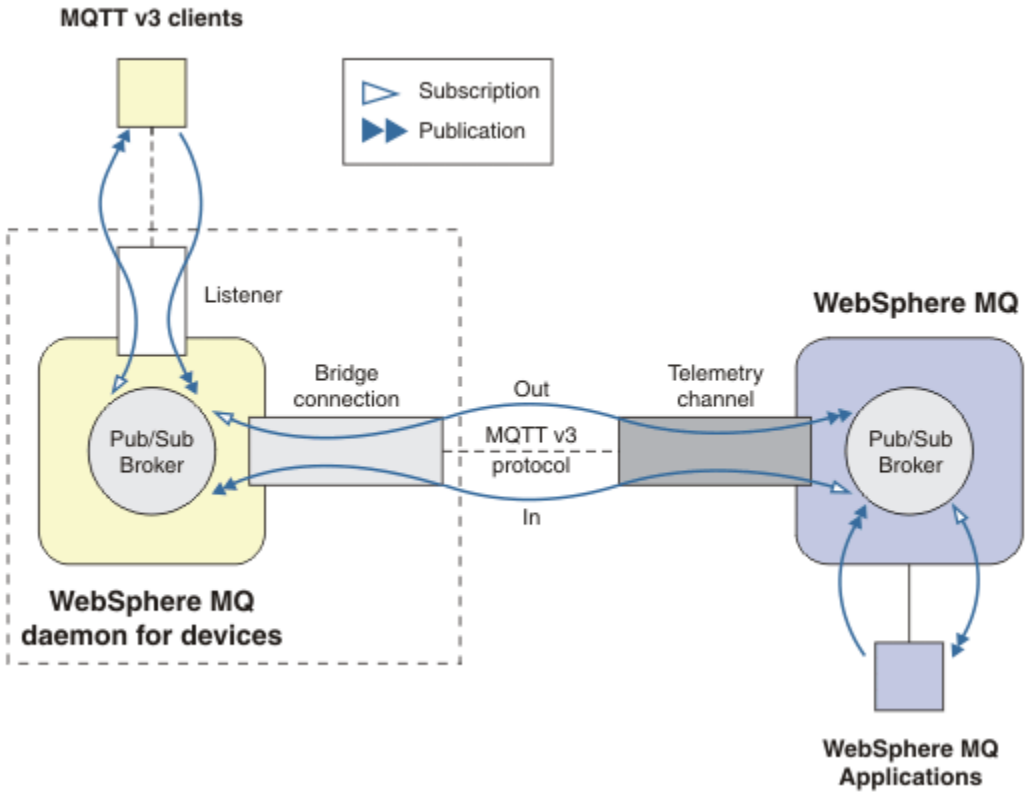
Aygıtlar için WebSphere MQ Telemetry cini için birden çok köprü yapılandırabilirsiniz. Yayınları değiş tokuş edebilen bir dizi yardımcı programı birbirine bağlamak için köprüleri kullanın.

Her bir köprü, yerel yardımcı programındaki konulara abone olabilir ve bu konulara abone olabilir. Ayrıca, başka bir yardımcı program, bir WebSphere MQ yayınlama/abone olma/abone olma aracısındaki ya da bağlı olduğu başka bir MQTT v3 aracısındaki konuları yayınlayabilir ve bunlara abone olabilir. Bir konu süzgecini kullanarak, yayınların bir aracıdan başka bir aracıya dağıtılmasını seçebilirsiniz. Yayınları her iki yöne de geçirebilirsiniz. You can propagate publicaitons from the local daemon to each of its attached remote brokers, or from any of the attached brokers to the local daemon; see “Aygıtlar köprüleri içinIBM WebSphere MQ Telemetry cini” sayfa 148.

Aygıtlar köprüleri içinIBM WebSphere MQ Telemetry cini

Bir IBM WebSphere MQ Telemetry cini for devices bridge, MQTT v3 iletişim kuralını kullanarak iki yayınlama/abone olma aracısını bağlar. Köprü, yayınları bir aracıdan diğerine, her iki yönde de yaylar. Bir uçta, aygıtlar köprü bağlantısı için bir WebSphere MQ Telemetry cini ve diğerinde bir kuyruk yöneticisi ya da başka bir yardımcı program olabilir. Bir kuyruk yöneticisi, telemetri kanalı kullanarak köprü bağlantısına bağlanır. Yardımcı program dinleyicisi kullanılarak köprü bağlantısına bir yardımcı program bağlanır.

IBM WebSphere MQ Telemetry cini aygıtları, diğer aracılara bir ya da daha fazla eşzamanlı bağlantıyı destekler. Yardımcı programdaki bağlantılar köprü adı verilir ve yardımcı program yapılandırma dosyasındaki bağlantı girişleriyle tanımlanır. IBM WebSphere MQ ile kurulan bağlantılar, aşağıdaki şekilde gösterildiği gibi IBM WebSphere MQ telemetri kanalları kullanılarak yapılır:



Şekil 30. IBM WebSphere MQ Telemetry daemon for devices ile IBM WebSphere MQarasında bağlantı kuruluyor

Bir köprü, cini başka bir aracıya MQTT v3 istemcisi olarak bağlar. Köprü parametreleri, bir MQTT v3 istemcisinin özniteliklerini ikizlemektedir.

Köprü bir bağlantıdan daha fazlasıdır. Bu, iki yayınlama/abone olma aracısı arasında yer alan bir yayınlama ve abone olma aracısı görevi görür. Yerel aracı, aygıtlar için IBM WebSphere MQ Telemetry yardımcı

programsıdır ve uzak aracı, MQTT v3 iletişim kuralını destekleyen bir yayınlama/abone olma aracısıdır. Genellikle uzak aracı başka bir yardımcı program ya da IBM WebSphere MQ' dir.

Köprünün işi, iki komisyoncu arasındaki yayınları yaymak. Köprü iki yönlü. Yayınları her iki yönde de yayıyor. Şekil 30 sayfa 148 , köprünün IBM WebSphere MQ Telemetry cinini aygıtlar için IBM WebSphere MQ' e nasıl bağlatacağı şekilde gösterir. “Köprüye ilişkin örnek konu ayarları” sayfa 149 , köprüyü yapılandırmak için konu değiştirgenin nasıl kullanılacağını göstermek için örnekleri kullanır.

Şekil 30 sayfa 148 içindeki In ve Out okları, köprünün iki yönlülüğü olduğunu belirtir. Okun bir ucunda abonelik yaratılır. Abonelikle eşleşen yayınlar, okun ters ucundaki aracıya yayınlanır. Ok, yayınların akışına göre etiketlenir. Yayın akışı Giriş cinine ve yardımcı programdan Çıkış ' a (Windows) akışını sağlar. Etiketlerin önemi, komut sözdiziminde kullanılırlar. In (Giriş) ve Out (Çıkış) başlıklı konuda, yayınların gönderileceği yere değil, aboneliğin gönderileceği yere başvurmamasını unutmayın.

Other clients, applications, or brokers might be connected either to IBM WebSphere MQ or to WebSphere MQ Telemetry daemon for devices. Bunlar, bağlı oldukları aracıdaki konuları yayınlayıp abone olarak yayınlarlar. Aracı IBM WebSphere MQise, konular kümelenmiş ya da dağıtılmış olabilir ve yerel kuyruk yöneticisinde belirtik olarak tanımlanmaz.

Köprülerin kullanımı

Yardımcı bilgi işlem bağlarını, köprü bağlantılarını ve dinleyicilerini kullanarak birbirine bağlayın. Köprü bağlantılarını ve telemetri kanallarını kullanarak yardımcı bilgi işlem ve kuyruk yöneticilerini bir araya bağlar. Birden çok aracıyı bir araya bağladığınızda, döngüler yaratılabilir. Dikkatli olun: Yayınlar, durmaksızın aracıların döngülerinin çevresinde dolaştırılabilir, algısız olarak dağıtılabilir.

IBM WebSphere MQ ' e köprülü yardımcı programların kullanılmasının bazı nedenleri şunlardır:

MQTT istemci bağlantılarının sayısını WebSphere MQolarak azaltın.
Yardımcı programların sıradüzenini kullanarak birçok istemciyi WebSphere MQ' ya bağlayabilirsiniz; tek bir kuyruk yöneticisinin aynı anda bağlanabildiği sayı sayısından daha fazla istemci kullanılabilir.

MQTT istemcileri ile WebSphere MQarasında iletileri saklama ve iletme
İstemciler kendi depolamalarına sahip değilse, istemciler ve IBM WebSphere MQarasındaki sürekli bağlantıların korunmasını önlemek için mağazana ve ileriye doğru ilerleyebilirsiniz. MQTT istemcisi ile WebSphere MQarasında birden çok bağlantı tipi kullanabilir; bkz. İzleme ve denetim için telemetri kavramları ve senaryoları.

MQTT istemcileri ile WebSphere MQarasında değiş tokuş edilen yayınlara süzgeç uygula
Yaygın olarak, yayınlar yerel olarak işlenen iletilere ve diğer uygulamaları içeren iletilere bölünmesini sağlar. Yerel yayınlar, algılayıcılar ve çalıştırıcılar arasında denetim akışlarını içerebilir ve uzak yayınlar okuma, durum ve yapılandırma komutlarına ilişkin istekleri içerir.

Yayınların konu alanlarını değiştirme
Farklı dinleyici kapılarına bağlı istemcilerden gelen konu dizgillerini birbiriyle çakışmadan kaçının. Bu örnek, farklı binalardan gelen ölçüm okuma değerlerini etiketlemek için yardımcı programı kullanır; bkz. Farklı istemci gruplarının konu alanlarını ayrışıma.

Köprüye ilişkin örnek konu ayarları

Uzak aracıya her şeyi yayınla-varsayılan değerler kullanılıyor

Varsayılan yön out(çıkış) olarak adlandırılır ve köprü konuları uzak aracıya yayınlamaktadır. konu değiştirgesi, konu süzgeçlerini kullanarak hangi konuların dağıtılmasını denetler.

Köprü, yerel cini MQTT istemcilerine ya da diğer aracılara göre yayınlanan her şeye abone olmak için Şekil 31 sayfa 150 içindeki konu değiştirgesini kullanır. Köprü, konuları köprünün bağlı olduğu uzak aracıya yayınlar.

```
connection Daemon1
topic #
```

Şekil 31. Her şeyi uzak aracıya yayınla

Her şeyi uzak aracıya yayınla-belirtik

Aşağıdaki kod parçasındaki konu ayarı, varsayılan değerlerin kullanılmasından sonra aynı sonucu verir. Tek fark, **direction** parametresinin açık olması olabilir. Yerel aracıya, yardımcı programa abone olmak ve uzak aracıya yayınlamak için `out` yönünü kullanın. Köprünün abone olduğu yerel yardımcı program üzerinde oluşturulan yayınlar, uzak aracıda yayınlanır.

```
connection Daemon1
topic # out
```

Şekil 32. Her şeyi uzak aracıya yayınla-belirtik

Her şeyi yerel aracıya yayınla

`dışarı`yönünü kullanmak yerine, ters yönü (`içinde`) ayarlayabilirsiniz. Aşağıdaki kod parçası, köprünün, köprünün bağlı olduğu uzak aracıda yayınlanan her şeye abone olacak şekilde yapılandırılarak yapılandırılarak yapılandırılabilirir. Köprü, konuları yerel aracıya, yardımcı programdan yayınlar.

```
connection Daemon1
topic # in
```

Şekil 33. Her şeyi yerel aracıya yayınla

Yerel aracıdaki dışa aktarma konusundan uzak aracıdaki içe aktarma konusuna ilişkin her şeyi yayınla

Use two additional topic parameters, **local_prefix** and **remote_prefix**, to modify the topic filter, # in the previous examples. Abonelikte kullanılan konu süzgecini değiştirmek için bir değiştirge kullanılır; diğer değiştirge ise, yayının yayınlandığı konuyu değiştirmek için kullanılır. Bu etki, diğer aracıdaki başka bir konu dizgisi olan bir aracıda kullanılan konu dizgisinin başlangıcının yerini alır.

Konunun yönlerine bağlı olarak, **local_prefix** ve **remote_prefix** ' nin anlamı geri çevrilmesine neden olur. Yön `out`ise, varsayılan değer olan **local_prefix** , konu aboneliğinin bir parçası olarak kullanılır ve **remote_prefix** , uzak yayındaki konu dizgisinin **local_prefix** bölümünün yerini alır. Yön `in`ise, **remote_prefix** uzak aboneliğin bir parçası olur ve **local_prefix** , konu dizgisinin **remote_prefix** kısmını değiştirir.

Bir konu dizgisinin ilk bölümü genellikle bir konu alanı tanımlama olarak düşünülemektedir. Bir konunun yayınlandığı konu alanını değiştirmek için ek parametreleri kullanın. Bu işlemi, konunun hedef aracıdaki başka bir konuyla çakışmasını önlemek ya da bir bağlama noktası konu dizesini kaldırmak için bunu yapabilirsiniz.

Örnek olarak, aşağıdaki kod parçasında, yardımcı programdaki `export/`# konu dizgisine ilişkin tüm yayınlar uzak aracıda `import/`# olarak yeniden yayınlanır.

```
topic # out export/ import/
```

Şekil 34. Yerel aracıdaki dışa aktarma konusundan uzak aracıdaki içe aktarma konusuna ilişkin her şeyi yayınla

Uzak aracıdaki dışa aktarma konusundan yerel aracıdaki içe aktarma konusuna ilişkin her şeyi yayınla

Aşağıdaki kod parçası yapılandırmayı tersine çevrilmiş olarak gösterir; köprü, uzak aracıdaki `export/#` konu dizgisiyle yayınlanan her şeye abone olur ve yerel aracıda bunu `import/#` olarak yayınlar.

```
connection Daemon1
topic # in import/ export/
```

Şekil 35. Uzak aracıdaki dışa aktarma konusundan yerel aracıdaki içe aktarma konusuna ilişkin her şeyi yayınla

Publish everything from the 1884/ mount point to the remote broker with the original topic strings

Aşağıdaki kod parçasında, köprü yerel yardımcı programdaki 1884/ sisteme bağlama noktasına bağlı istemciler tarafından yayınlanan her şeye abone olur. Köprü, bağlama noktasına yayınlanan her şeyi uzak aracıya yayınlar. The mount point string 1884/ is removed from the topics published to the remote broker. The *yerel_öneki* is the same as the mount point string 1884/, and the *remote_prefix* is a blank string.

```
listener 1884
mount_point 1884/
connection Daemon1
topic # out 1884/ ""
```

Şekil 36. 1884/ bağlama noktasından, özgün konu dizgileriyle uzak aracıya her şeyi yayınlayın.

Farklı istemcilere bağlı farklı istemcilerin konu alanlarını ayırma

Bir bina için ölçüm okuma değerlerini yayınlayabilmek için elektrik enerjisi sayaçları için bir uygulama yazıldığını varsayın. Okuma değerleri, MQTT istemcileri kullanılarak aynı binada barındırılan bir yardımcı programa yayınlanır. Yayınlar için seçilen konu `power`. Aynı uygulama, karmaşık bir alan içinde bir dizi binaya da dağıtılır. Site izleme ve veri depolama için, tüm binalardan gelen okuma değerleri köprü bağlantıları kullanılarak toplanır. Bağlantılar, oluşturma yardımcı programı merkezi bir yerdeki WebSphere MQ ' ya bağlar.

Tüm binalarda aynı istemci uygulaması kullanılır. Bu uygulama, `power`başlıklı konuya yayınlar. Ancak veriler oluşturularak farklılaştırılmalıdır. Bu işlem, bina numarasını konu adına önek olarak ekleyen her bir bina için yardımcı program tarafından yapılır. The bridge from the first building in the complex uses the prefix `meters/building01/`, from building two the prefix is `meters/building02/`. Diğer binalardan gelen okumalar aynı modeli takip ediyor. WebSphere MQ bu nedenle, `meters/building01/power`gibi konuları içeren okumaları alır.

Her bir yardımcı program için yapılandırma dosyası, aşağıdaki kod parçasındaki örüntüden sonra gelen bir konu deyimine sahiptir:

```
connection Daemon1
topic power out "" meters/building01/
```

Şekil 37. Farklı yardımcı dillere bağlı istemcilerin konu alanlarını birbirinden ayırın

Önceki kod parçasında, boş dizgi, kullanılmayan local_prefix parametresi için yer tutucudur.

Not: Bu örnek, bir şekilde yapay ve yalnızca bir şekil olarak tasarlanmıştır. Uygulamada, uygulamanın yayınlandığı konu alanının yapılandırılabilir olma olasılığı yüksektir.

Aynı yardımcı programa bağlı istemcilerin konu alanlarını ayır

Tüm güç ölçümlerini bağlamak için tek bir yardımcı program kullanıldığını varsayın. Uygulamada farklı kapılara bağlanmak üzere yapılandırılabildiğine göre, aşağıdaki kod parçasındaki gibi metreleri farklı binalardan farklı dinleyici kapılarına bağlayarak binaları ayırt edebilirsiniz. Yine, örnek, bağlama noktalarının nasıl kullanılabileceğini gösterir.

```
listener 1884
mount_point meters/building01/
listener 1885
mount_point meters/building02/
connection Daemon1
topic meters/+/power out
```

Şekil 38. Aynı yardımcı programa bağlı istemcilerin konu alanlarını ayır

Her iki yönde akan yayınlar için farklı konuları yeniden eşleyin

In the configuration in the following code fragment, the bridge subscribes to the single topic b at the remote broker and forwards publications about b to the local daemon, changing the topic to a. The bridge also subscribes to the single topic x at the local broker and forwards publications about x to the remote broker, changing the topic to y.

```
connection Daemon1
topic "" in a b
topic "" out x y
```

Şekil 39. Her iki yönde akan yayınlar için farklı konuları yeniden eşleyin

Bu örneğe ilişkin önemli bir nokta, her iki aracıda da farklı konuların abone olunması ve yayınlanma noktalarının yayınlanması ile ilgilidir. Her iki aracıdaki konu alanları da dağılır.

Her iki yönde akan yayınlar için de aynı konuları yeniden eşleyin (döngüleme)

Önceki örnekten farklı olarak, Şekil 40 sayfa 152içindeki yapılandırma, genel olarak bir döngüdeki sonuçlarla sonuçlanır. `topic "" in a` badlı konu deyiminde, köprü b 'a uzaktan abone olur ve yerel olarak a ' a yayınlar. In the other topic statement, the bridge subscribes to a locally, and publishes to b remotely. Aynı yapılandırma, Şekil 41 sayfa 153içinde gösterildiği gibi yazılabilir.

The general result is that if a client publishes to b remotely, the publication is transferred to the local daemon as a publication on topic a. However, on being published by the bridge to the local daemon on the topic a, the publication matches the subscription made by the bridge to local topic a. Abonelik `topic "" out a b`. As a result, the publication is transferred back to the remote broker as a publication on topic b. The bridge is now subscribed to the remote topic b, and the cycle begins again.

Bazı aracılar, döngülerin gerçekleşmesini önlemek için döngü algılamasını gerçekleştirir. Ancak, farklı tipte aracılar birleştiğinde döngü algılama mekanizmasının çalışması gerekir. Loop detection does not work if WebSphere MQ is bridged to the WebSphere MQ Telemetry daemon for devices. Aygıtlar için iki IBM WebSphere MQ Telemetry yardımcı programı birbirine sıkışmışsa, bu iş çalışır. Varsayılan döngü algılaması açık olarak açıktır; bkz. try_private.

```
connection Daemon1
topic "" in a b
topic "" out a b
```

*Şekil 40. **!**Her iki yönde akan yayınlar için aynı konuları yeniden eşleyin*

```
connection Daemon1
topic "" both a b
```

*Şekil 41. **!**bothkullanarak, her iki yönde akan yayınlarla aynı konuları yeniden eşleyin.*

Şekil 39 sayfa 152 içindeki yapılandırma, Şekil 40 sayfa 152ile aynıdır.

IBM WebSphere MQ Telemetry daemon for devices köprüsü bağlantılarının kullanılabilirliği

Var olan ilk uzak aracıya bağlanmak için birden çok IBM WebSphere MQ Telemetry daemon for devices köprüsü bağlantı adresi yapılandırın. Aracı çok eşgörünümli bir kuyruk yöneticisiyse, her iki TCP/IP adresini de belirtin. Kullanılabilir olduğunda birincil sunucuya bağlanmak ya da yeniden bağlanmak için bir birincil bağlantı yapılandırın.

Bağlantı köprüsü parametresi (adresler), TCP/IP yuva adreslerinden oluşan bir listedir. Köprü, başarılı bir bağlantı kuruncaya kadar sırayla her adrese bağlanmayı dener. yuvarlak _robin ve start_type bağlantı parametreleri, başarılı bir bağlantı yapıldıktan sonra adreslerin nasıl kullanıldığını denetler.

start_type , auto(otomatik), manual(otomatik) ya da lazy(tembel) ise, bağlantı başarısız olursa, köprü yeniden bağlanmayı dener. Her bir adresi sırayla kullanır, her bağlantı girişimi arasında yaklaşık 20 saniyelik bir gecikme olur. `start_type` , once(kez) ise, bağlantı başarısız olursa, köprü otomatik olarak yeniden bağlanmayı denemez.

round_robin trueise, köprü bağlantısı, listedeki ilk adresten başlar ve listedeki her adresi sırayla dener. Liste çok tükendiğinde, ilk adresten başlar. Listede yalnızca bir adres varsa, her 20 saniyede bir yeniden dener.

round_robin değeri falseise, listedeki birincil sunucu adı verilen ilk adres tercihtir. İlk sunucuya bağlanma girişimi başarısız olursa, köprü arka plandaki birincil sunucuya yeniden bağlanmayı denemeye devam eder. Aynı zamanda köprü, listedeki diğer adresleri kullanarak bağlanmayı dener. Arka plan birincil sunucuya bağlanmayı denediğinde, köprü geçerli bağlantıdan bağlantıyı keser ve birincil sunucu bağlantısına geçiş yapar.

Bir bağlantının kendi isteğiyle bağlantısı kesilirse (örneğin, bir **connection_stop** komutu yayınlayarak), bağlantı yeniden başlatılırsa, aynı adresi kullanmaya devam eder. Bağlanmadaki bir hata nedeniyle bağlantı kesilirse ya da uzak aracıya bağlantıyı bıraktıysa, köprü 20 saniye bekler. Daha sonra listede yalnızca bir adres varsa, listedeki bir sonraki adrese ya da aynı adrese bağlanmaya çalışır.

Çok eşgörünümli bir kuyruk yöneticiyle bağlantı kurulması

Çok eşgörünümli kuyruk yöneticisi yapılanışında, kuyruk yöneticisi farklı IP adresleriyle iki farklı sunucuda çalışır. Tipik olarak telemetri kanalları, belirli bir IP adresi olmadan yapılandırılır. Bunlar yalnızca bir kapı numarasıyla yapılandırılırlar. telemetri kanalı başlatıldığında, varsayılan olarak yerel sunucudaki ilk kullanılabilir ağ adresini seçer.

Kuyruk yöneticisi tarafından kullanılan iki IP adresi ile köprü bağlantısının adresler parametresini yapılandırın. round_robin öğesini true olarak ayarlayın.

Etkin kuyruk yöneticisi yönetim ortamı başarısız olursa, kuyruk yöneticisi yedek yönetim ortamına geçiş yapar. Yardımcı program, etkin eşgörünümle bağlantının bozuk olduğunu saptar ve yedek yönetim ortamına yeniden bağlanmayı dener. Köprü bağlantısı için yapılandırılan adresler listesinde yer alan diğer IP adresini kullanır.

Köprünün bağlandığı kuyruk yöneticisi hala aynı kuyruk yöneticisindir. Kuyruk yöneticisi kendi durumunu kurtarır. cleansession false olarak ayarlanırsa, köprü bağlantı oturumu, hata durumunda yedek denetleyiciye geçiş işleminden önce aynı duruma geri yüklenir. Bir gecikmeden sonra bağlantı devam eder. Messages with "en az bir kez" or "en çok bir kez" quality of service are not lost, and subscriptions continue to work.

Yeniden bağlanma süresi, yedek yönetim ortamı başlatıldığında yeniden başlatılan kanal sayısına ve istemcilerin sayısına ve ne kadar ileti ışığının yandığını bağlıdır. Köprü bağlantısı, bağlantı yeniden oluşturulmadan önce her iki IP adresine yeniden bağlanmayı deneyebilir.

Belirli bir IP adresine sahip çok eşgörünümli bir kuyruk yöneticisi telemetri kanalı yapılandırmayın. IP adresi yalnızca tek bir sunucu üzerinde geçerlidir.

IP adresini yöneten alternatif bir yüksek kullanılabilirlik çözümü kullanıyorsanız, bir telemetri kanalını belirli bir IP adresiyle yapılandırmak için doğru olabilir.

cleansession

Köprü bağlantısı, MQTT v3 istemci oturumdur. Bir bağlantının yeni bir oturum mı başlatılacağını, yoksa varolan bir oturumu mı geri yükleyeceğini denetleyebilirsiniz. Var olan bir oturumu geri yüklerse, köprü bağlantısı abonelikleri korur ve önceki oturumdan yayınları alıkoyar.

adresler birden çok IP adresi listeleniyorsa ve IP adresleri farklı kuyruk yöneticileri tarafından barındırılan telemetri kanallarına ya da farklı telemetri yardımcı kanallarına bağlanıyorsa, temizoturum ' yi false olarak ayarlamayın. Oturum durumu, kuyruk yöneticileri ya da damasonlar arasında aktarılmaz. Farklı bir kuyruk yöneticisinde ya da yardımcı programında varolan bir oturumu yeniden başlatmaya çalışmak, yeni bir oturum başlatılmakta olan bir oturumla sonuçlanır. Belirsiz iletiler kaybedilir ve abonelikler beklendiği gibi davranmayabilir.

bildirimler

Bir uygulama, bildirimler kullanılarak köprü bağlantısının çalışıp çalışmadığına ilişkin izi tutabilir. Bildirim, `1`değerine, bağlı ya da `0`değerine sahip, bağlantısı kesik olan bir yayındır. Bu, `notification_topic` parametresi tarafından tanımlanan *topicString* olarak yayınlanır. Varsayılan değer olan *topicString* , $SYS/`broker/connection/` *clientIdentifier*`/state`' dir. Varsayılan *topicString* , $SYSönekini içerir. Subscribe to topics beginning with $SYS by defining a topic filter beginning with $SYS. #konu süzgeci, her şeye abone olabilir, yardımcı program üzerinde $SYS ile başlayan konulara abone olmaz. Uygulama konusu alanından farklı bir özel sistem konu alanı tanımlanırken $SYS ' i düşünün.

Bildirimler , bir köprü bağlandığında ya da bağlantısı kesildiğinde MQTT istemcilerine bildirimde bulunmak için IBM WebSphere MQ Telemetry daemon for devices ' i etkinleştirin.

Keepalive_interval

Keepalive_interval köprü bağlantı parametresi, uzak sunucuya TCP/IP ping komutu gönderen köprü arasındaki aralığı belirler. Varsayılan aralık 60 saniyedir. Ping, TCP/IP oturumunun uzak sunucu tarafından ya da bir güvenlik duvarı tarafından kapatılmasını önler, bu da bağlantıda bir etkinlik dışı durum olduğunu saptar.

clientid

Bir köprü bağlantısı, bir MQTT v3 istemci oturumdur ve clientidköprü bağlantı parametresi tarafından ayarlanan bir `clientIdentifier` değerine sahiptir. If you intend reconnections to resume a previous session by setting the temizoturum parameter to yanlış, the `clientIdentifier` used in each session must be the same. Varsayılan değer olan clientid , *hostname.connectionName*değeridir. Bu değer aynı kalır.

Aygıtlara ilişkin WebSphere MQ Telemetry cininin kurulması, doğrulanması, yapılandırılması ve denetlenmesi

Yardımcı programın kuruluşu, yapılanışı ve denetimi dosya tabanlıdır.

Yardımcı programı, Software Development Kit 'i, yardımcı programı çalıştıracadiğiniz aygıta kopyalayarak kurun.

Örnek olarak, MQTT istemci yardımcı programını çalıştırın ve yayınlama/abone olma aracısı olarak aygıtlara ilişkin WebSphere MQ Telemetri yardımcı programına bağlanın; bkz. Yayınlama/abone olma aracısı olarak aygıtlar için WebSphere MQ Telemetry yardımcı programını kullanın.

Bir yapılandırma dosyası oluşturarak cini yapılandırın; bkz. Aygıtlar yapılandırma dosyası içinWebSphere MQ Telemetry cini.

Control a running daemon by creating commands in the file, `amqtdd.upd`. Yardımcı programın dosyayı okuduğu her 5 saniyede bir, komutları çalıştırır ve dosyayı siler; bkz. Aygıtlar komut dosyası içinWebSphere MQ Telemetry cini.

Aygıtlar dinleyici kapıları içinWebSphere MQ Telemetry cini

Dinleyici kapıları kullanan aygıtlar için MQTT V3 istemcilerini WebSphere MQ Telemetry cinine bağlayın. Dinleyici kapısını bir bağlama noktası ve bağlantı sayısı üst sınırı ile niteleyebilirsiniz.

Bir dinleyici kapısı, bu kapıya bağlanan bir istemcinin MQTT istemcisi `connect(serverURI)` yönteminde belirtilen kapı numarasına karşılık gelmelidir. Varsayılan değer olarak hem istemci, hem de yardımcı program için 1883' a ayarlanır.

You can change the default port for the daemon by setting the global definition `kapı` in the daemon configuration file. Yardımcı program yapılandırma dosyasına bir `dinleyici` tanımlaması ekleyerek belirli kapıları ayarlayabilirsiniz.

Varsayılan kapı dışında her bir dinleyici kapısı için, istemcileri yalıtmak için bir bağlama noktası belirtebilirsiniz. Bağlama noktası bulunan bir kapıya bağlı istemciler diğer istemcilerden yalıtılır; bkz. “Aygıtlara bağlama noktaları içinWebSphere MQ Telemetry cini” sayfa 155.

Herhangi bir kapıya bağlanabilecek istemcilerin sayısını sınırlayabilirsiniz. Set the global definition `bağlantı_sayısı` to limit connections to the default port, or qualify each listener port with `bağlantı_sayısı`.

Örnek

An example of a configuration file that changes the default port from 1883 to `1880`, and limits connections to port 1880 to `10000`. 1884 bağlantı noktasına yönelik bağlantılar `1000`ile sınırlıdır. 1884 kapısına bağlanan istemciler, diğer kapılara bağlı istemcilerden yalıtılır.

```
port 1880
max_connections 10000
listener 1884
mount_point 1884/
max_connections 1000
```

Aygıtlara bağlama noktaları içinWebSphere MQ Telemetry cini

Bir bağlama noktasını, aygıtlar için bir WebSphere MQ Telemetry cinine bağlanmak için MQTT istemcileri tarafından kullanılan bir dinleyici kapısıyla ilişkilendirebilirsiniz. Bir sisteme bağlama noktası, farklı bir dinleyici kapısına bağlı MQTT istemcilerinden bir dinleyici kapısını kullanarak, MQTT istemcilerinin değiş tokuş edilen yayınlarını ve abonelikleri yalıtır.

Bir bağlama noktasıyla bir dinleyici kapısına bağlanan istemciler, diğer dinleyici kapılarına bağlı istemcilerle konuları doğrudan değiştiremez. Bir bağlama noktası olmayan bir dinleyici kapısına bağlanan istemciler, herhangi bir istemcinin konularına yayınlayabilir ya da bu konuları abone olabilirler. Müşteriler, bir bağlama noktası aracılığıyla bağlı olup olmadıkları konusunda bilgi sahibi değildir; müşteriler tarafından yaratılan konu dizgileriyle hiçbir fark yaratmaz.

Bağlama noktası, yayın ve aboneliklerin konu dizgisine önek olarak eklenen bir metin dizesidir. Dinleyici kapısına bir bağlama noktası olan istemciler tarafından yaratılan tüm konu dizgilerinin başına önek olarak eklenir. Metin dizesi, dinleyici kapısına bağlı istemcilere gönderilen tüm konu dizgilerinden kaldırılır.

Bir dinleyici kapısında bağlama noktası yoksa, kapıya bağlı istemciler tarafından oluşturulan ve alınan yayınların ve aboneliklerin konu dizgileri değiştirilmez.

Sonda /olan bağlama noktası dizgileri oluşturun. Bu şekilde bağlama noktası, bağlama noktası için konu ağacının üst konudur.

Örnek

Yapılandırma dosyası, aşağıdaki dinleyici kapılarını içerir:

```
listener 1883
mount_point 1883/
listener 1884 127.0.0.1
mount_point 1884/
listener 1885
```

1883kapısına bağlı bir istemci, MyTopiciçin bir abonelik oluşturur. Yardımcı program aboneliği 1883/MyTopicolarak kaydettirir. Another client attached to port 1883 publishes a message on the topic, MyTopic. Yardımcı program, konu dizesini 1883/MyTopic olarak değiştirir ve eşleşen abonelikleri arar. The subscriber on port 1883 receives the publication with the original topic string MyTopic. Yardımcı program, bağlama noktası önekini konu dizgisinden kaldırdı.

Another client, attached to port 1884, also publishes on the topic MyTopic. Bu kez, yardımcı programı konuyu 1884/MyTopicolarak kaydettirir. Farklı bağlama noktası, farklı bir konu dizgisine sahip bir abonelikte sonuçlandığından, 1883 numaralı kapıdaki abone yayını almaz.

A client, attached to port 1885, publishes on the topic, 1883/MyTopic. Yardımcı program konu dizgisini değiştirmiyor. The subscriber on port 1883 receives the publication to MyTopic.

Hizmet kalitesi, dayanıklı abonelikler ve alıkonan yayınlar içinWebSphere MQ Telemetry cini

Hizmet kalitesi ayarları, yalnızca çalışan bir yardımcı program için geçerlidir. Bir yardımcı program durdurursa, denetimli bir şekilde ya da bir hata nedeniyle beliren iletilerin durumu kaybedilir. Bir iletinin teslim edilmesi en az bir kez ya da en azından bir kez durdurulacaksa garanti edilemez. Aygıtlar içinWebSphere MQ Telemetry cini sınırlı kalıcılığı destekler. Yardımcı program sona erdirildiğinde alıkonan yayınları ve abonelikleri saklamak için **retained_persistence** yapılanış değiştirgesini ayarlayın.

Unlike WebSphere MQ, the WebSphere MQ Telemetry daemon for devices does not journal persistent data. Oturum durumu, ileti durumu ve alıkonan yayınlar iletsel olarak kaydedilmez. Varsayılan olarak, yardımcı program durduğunda tüm verileri atar. Abonelikleri düzenli olarak kontrol etmek ve yayınları alıkonmak için bir seçenek ayarlayabilirsiniz. Yardımcı program durduğunda ileti durumu her zaman kaybedilir. Alıkonmayan tüm yayınlar kaybedilir.

Alıkonan yayınları belirli aralıklarla bir dosyaya kaydetmek için yardımcı program yapılandırma seçeneğini Retained_persistence öğesini trueolarak ayarlayın. Yardımcı program yeniden başlatıldığında, en son otomatik olarak kaydedilen yayınların geri alınması gerekir. Varsayılan olarak, istemciler tarafından yaratılan alıkonan iletiler, yardımcı program yeniden başlatıldığında yeniden yürürlüğe alınmaz.

Kalıcı bir oturumda oluşturulan abonelikleri bir dosyaya düzenli olarak kaydetmek için, yardımcı program yapılandırma seçeneğini Retained_persistence trueolarak ayarlayın. Retained_peristence değeri trueolarak ayarlanırsa, istemcilerin CleanSession ile false, bir "kalıcı oturum"olarak ayarlanmış bir oturumda oluşturulacağı abonelikler geri yüklenir. Yardımcı program, yayınları yeniden başlattığında, bu abonelikleri yeniden saklar. The client receives the publications when it restarts with CleanSession to false. By default, client session state is not saved when a daemon stops, and so subscriptions are not restored, even if the client sets CleanSession to false.

Retained_persistence , otomatik kaydetme mekanizmasıdır. En son tutulan yayınlar ya da abonelikler saklanmayabilir. Tutulan yayınların ve aboneliklerin hangi sıklıkta kaydedileceğini değiştirebilirsiniz. autosave_on_changes yapılandırma seçeneklerini ve autosave_intervalyapılandırma seçeneklerini kullanarak, kaydetme arasındaki aralığı ya da kaydetme arasındaki değişiklik sayısını ayarlayın.

Kalıcılık ayarına ilişkin örnek yapılandırma

```
# 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
```

Aygıt güvenliği içinWebSphere MQ Telemetry cini

Aygıtlar için WebSphere MQ Telemetry cini, buna bağlanan istemcilerin kimliklerini doğrulayabilir, diğer aracılara bağlanmak için kimlik bilgilerini kullanabilir ve konulara erişimi denetleyebilir. Yardımcı programın sağladığı güvenlik, SSL desteği sağlamayan WebSphere MQ Telemetry C istemcisi kullanılarak oluşturularak sınırlandırılır. Sonuç olarak, yardımcı programa bağlantılar şifrelenmez ve sertifikalar kullanılarak kimliği doğrulanamaz.

Varsayılan değer olarak, hiçbir güvenlik açık değildir.

İstemcilerin kimlik doğrulaması

MQTT istemcileri, `MqttConnectOptions.setUserName` ve `MqttConnectOptions.setPassword`yöntemlerini kullanarak bir kullanıcı adı ve parola ayarlayabilirler.

Parola dosyasındaki girişlere karşı istemci tarafından sağlanan kullanıcı adını ve parolayı denetleyerek yardımcı programa bağlanan bir istemciyi doğrulayın. Kimlik doğrulamasını etkinleştirmek için, bir parola dosyası oluşturun ve yardımcı program yapılandırma dosyasında `password_file` parametresini ayarlayın; bkz. password_file.

İstemci adı ya da parola olmadan bağlanan istemcilerin kimlik doğrulaması yapan bir yardımcı programa bağlanmasına izin vermek için, yardımcı program yapılandırma dosyasında `allow_anonymous` değiştirgesini ayarlayın; bkz. allow_adonymous. Bir istemci kullanıcı adı ya da parola sağlıyorsa, `password_file` parametresi ayarlandıysa, bu kullanıcı her zaman parola dosyası ile karşılaştırılır.

Yardımcı program yapılandırma dosyasındaki `clientid_prefixes` parametresini, belirli istemcilerle bağlantıları sınırlandırmak için ayarlayın. İstemcilerin `client_prefixes` değiştirgesinde listelenen öneklerden biriyle başlayan `clientIdentifiers` (clientIdentifiersclientIdentifiers)haveolmalıdır; bkz. clientid_prefixes.

Köprü bağlantısı güvenliği

Aygıtlar köprü bağlantısı için her WebSphere MQ Telemetry cini bir MQTT V3 istemcidir. Her bir köprü bağlantısı için kullanıcı adı ve parolayı, yardımcı program yapılandırma dosyasında bir köprü bağlantı parametresi olarak ayarlayabilir; bkz. kullanıcı adı ve parola. Daha sonra bir köprü kendisini bir aracıya doğrulayabilir.

Konulara ilişkin erişim denetimi

İstemcilerin kimliği doğrulanırsa, yardımcı program her kullanıcıya ilişkin konulara denetim erişimi de sağlayabilir. Yardımcı program, bir istemcinin erişim denetimi dosyasındaki bir erişim konu dizgisiyle yayınlanmasını ya da abone olduğu konuyu eşleştirmeye dayalı olarak erişim denetimi verir; bkz. acl_file.

Erişim denetimi listesinin iki bölümü vardır. İlk parça, anonim istemciler de dahil olmak üzere tüm istemcilere erişimi denetler. İkinci bölüm, parola dosyasındaki herhangi bir kullanıcı için bir bölüm içerir. Her kullanıcı için belirli erişim denetimini listeler.

Örnek

Güvenlik değiştirgeleri aşağıdaki örnekte gösterilmiştir.

```
acl_file c:\WMQTDaemon\config\acl.txt
password_file c:\WMQTDaemon\config\passwords.txt
allow_anonymous true
connection Daemon1
username daemon1
password deamonpassword
```

Şekil 42. Yardımcı program yapılandırma dosyası

```
Fred:Fredpassword
Barney:Barneypassword
```

Şekil 43. Parola dosyası, 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/#
```

Şekil 44. Erişim denetimi dosyası, acl.txt

MQTT istemcilerinde sorun giderme

Çalışan MQTT istemcileriyle ilgili bir sorunu çözmenize yardımcı olacak bir sorun giderme görevi arayın.

İlgili görevler

"MQTT (Paho) Java istemcisine ilişkin izleme ve hata ayıklama" sayfa 167
Varsayılan günlüğe kaydedici, `java.util.logging` (JSR47) olarak bilinen standart Java günlüğe kaydetme olanağını kullanır. Bunu bir yapılanış kütüğü ya da programl olarak kullanarak konfigrasyonunu tanmlayabilirsiniz.

"MQTT JavaScript istemcisinin izlenmesi" sayfa 169
Bağlı istemci nesnesindeki yöntemleri çağırmak için istemci web uygulamasını değiştirerek izleme toplamak için JavaScript istemcisini kullanabilirsiniz.

"Telemetri (MQXR) hizmetinin izlenmesi" sayfa 162
Telemetri hizmetinin izini başlatmak için bu yönergeleri izleyin, izleme işlemini denetleyen parametreleri ayarlayın ve izleme çıkışını bulun.

"MQTT v3 Java istemcisinin izlenmesi" sayfa 164
Bir MQTT Java istemcisi izlemesi yaratmak ve çıkışını denetlemek için bu yönergeleri izleyin.

"C için MQTT istemcisini izleme" sayfa 165
Bir MQTT istemcisi C uygulamasını izlemek için `MQTT_C_CLIENT_TRACE` ortam değişkenini ayarlayın.

"Sorun çözülüyor: MQTT istemcisi bağlanmıyor" sayfa 176
Telemetry (MQXR) hizmetine bağlanmamakta başarısız olan bir MQTT istemci programının sorununu çözün.

"Sorun çözülüyor: MQTT istemci bağlantısı atıldı" sayfa 178
Kısa ya da uzun bir süre başarıyla bağlanıp çalıştıktan sonra, bir istemcinin beklenmeyen `ConnectionLost` kural dışı durumları yayınlanmasına neden olan nedir öğrenin.

"Sorun çözümleniyor: MQTT uygulamasında iletiler kaybedildi" sayfa 179
Bir iletiyi kaybetmenin sorununu çözün. İleti kalıcı değil, yanlış yere gönderildi ya da hiçbir zaman gönderilmedi mi? Yanlış bir şekilde kodlanmış bir istemci programı iletileri kaybedebilir.

“Sorun çözülüyor: Telemetry (MQXR) hizmeti başlamaz” sayfa 181
Başlatılamayan telemetri (MQXR) hizmetinin sorununu çözün. WebSphere MQ Telemetry kuruluşunu denetleyin ve herhangi bir dosya eksik, taşınmaz ya da yanlış izinlere sahip değildir. Telemetri (MQXR) hizmeti tarafından kullanılan yolları denetleyerek telemetri (MQXR) hizmet programlarını bulun.

“Sorun çözülüyor: JAAS oturum açma modülü telemetri hizmeti tarafından çağrılmadı” sayfa 182
Find out if your JAAS login module is not being called by the telemetry (MQXR) service, and configure JAAS to correct the problem.

“Sorun çözümleniyor: Yardımcı program başlatılıyor ya da çalıştırılıyor” sayfa 185
Operations Console günlüğü için IBM WebSphere MQ Telemetry cinine bakın, izlemeyi açın ya da yardımcı program ile ilgili sorunları gidermek için bu baştaki belirti çizelgesini kullanın.

“Sorun çözülüyor: MQTT istemcileri cine bağlanmıyor” sayfa 186
İstemciler yardımcı programa bağlanmıyor ya da yardımcı program diğer yardımcı programlara ya da bir WebSphere MQ telemetry kanalına bağlanmıyor.

İlgili başvurular

“Telemetrik günlüklerin, hata günlüklerinin ve yapılandırma dosyalarının konumu” sayfa 159
IBM WebSphere MQ Telemetry tarafından kullanılan günlükleri, hata günlüklerini ve yapılandırma dosyalarını bulun.

“MQTT v3 Java istemcisi neden kodları” sayfa 161
Bir MQTT v3 Java istemcisi kural dışı durumu ya da throwable içindeki neden kodlarının nedenlerine bakın.

“System requirements for using SHA-2 cipher suites with MQTT clients” sayfa 170
- IBM, SR13 ' den başlayarak Java 6 için, MQTT kanallarınızı ve istemci uygulamalarınızı güvenli kılmak için SHA-2 şifreleme takımlarını kullanabilirsiniz. However, SHA-2 cipher suites are not enabled by default until Java 7 from IBM, SR4 onwards, so in earlier versions you must specify the required suite. Kendi JRE ' nizle bir MQTT istemcisi çalıştırıyorsanız, bunun SHA-2 şifreleme takımlarını desteklediğinden emin olmanız gerekir. İstemci uygulamalarınızın SHA-2 şifreleme takımlarını kullanması için, istemcinin SSL bağlamını Transport Layer Security (TLS) sürüm 1.2' yi destekleyen bir değere ayarlaması gerekir.

“SSL üzerinden mobil ileti sistemi web uygulamaları için tarayıcı desteğindeki kısıtlamalar” sayfa 171
Farklı tarayıcılar üzerinde farklı tarayıcılar arasında yetenek farklılıkları vardır. Understanding these differences helps you configure your apps, certificate authorities (CAs) and client certificates to connect using the MQTT messaging client for JavaScript over SSL and WebSockets.

Telemetrik günlüklerin, hata günlüklerinin ve yapılandırma dosyalarının konumu

IBM WebSphere MQ Telemetry tarafından kullanılan günlükleri, hata günlüklerini ve yapılandırma dosyalarını bulun.

Not: Örnekler, Windowsiçin kodlanmıştır. Change the syntax to run the examples on Linux

Sunucu tarafındaki günlükler

IBM WebSphere MQ Telemetry kuruluş sihirbazı, kuruluş günlüğüne ileti yazar:

```
WMQ program directory\mqxr
```

Telemetri (MQXR) hizmeti, WebSphere MQ kuyruk yöneticisi hata günlüğüne ve FDC dosyalarının IBM WebSphere MQ hata dizinine iletileri yazar:

```
WMQ data directory\Qmgrs\qMgrName\errors\AMQERR01.LOG
WMQ data directory\errors\AMQnnn.n.FDC
```

Ayrıca, telemetri (MQXR) hizmeti için bir günlük yazar. Günlük, hizmetin başlatıldığı özellikleri ve bir MQTT istemcisi için yetkili sunucu olarak işlev bulduğu hataları görüntüler. Örneğin, istemcinin oluşturmadığı bir abonelikten aboneliği kaldırma. Günlük yolu:

```
WMQ data directory\Qmgrs\qMgrName\errors\mqxr.log
```

The IBM WebSphere MQ telemetry sample configuration created by IBM WebSphere MQ Explorer starts the telemetry service using the command **runMQXRService**. **runMQXRService** , *WMQ Telemetry install directory*\biniçinde. Bu, şunları yazar:

```
WMQ data directory\Qmgrs\qMgrName\mqxr.stdout
WMQ data directory\Qmgrs\qMgrName\mqxr.stderr
```

Telemetri (MQXR) hizmeti için yapılandırılmış yolları görüntülemek ya da telemetri (MQXR) hizmetini başlatmadan önce kullanıma hazırlamak için **runMQXRService** işlemini değiştirin.

Sunucu tarafı yapılandırma dosyaları

Telemetri kanalları ve telemetri (MQXR) hizmeti

Sınırlama: Telemetri kanalı yapılandırma dosyasının biçimi, konumu, içeriği ve yorumu, gelecekteki yayınlarda değişebilir. Telemetri kanallarını yapılandırmak için IBM WebSphere MQ Explorer 'ı kullanmanız gerekir.

IBM WebSphere MQ Explorer, telemetri yapılandırmalarını Windows' ta `mqxr_win.properties` dosyasında ve Linuxüzerindeki `mqxr_unix.properties` dosyasında saklar. Özellikler dosyaları telemetri yapılandırma dizinine kaydedilir:

```
WMQ data directory\Qmgrs\qMgrName\mqxr
```

Şekil 45. Telemetri yapılandırma dizini: Windows

```
/var/mqm/qmgrs/qMgrName/mqxr
```

Şekil 46. Linuxüzerinde telemetri yapılandırma dizini

JVM

Set Java properties that are passed as arguments to the telemetry (MQXR) service in the file, `java.properties`. Dosyadaki özellikler doğrudan telemetri (MQXR) hizmeti çalıştıran JVM ' ye iletilir. Bunlar, Java komut satırında ek JVM özellikleri olarak geçirilir. Properties set on the command line take precedence over properties added to the command line from the `java.properties` file.

Telemetri yapılandırmalarıyla aynı klasörde bulunan `java.properties` dosyasını bulun, bkz. Şekil 45 sayfa 160 ve Şekil 46 sayfa 160.

Her özelliği ayrı bir satır olarak belirterek `java.properties` ' i değiştirin. Her bir özelliği, özelliği JVM ' ye bir bağımsız değişken olarak geçirmeniz için tam olarak biçimlemenizi sağlar; örneğin:

```
-Xmx1024m
-Xms1024m
```

JAAS

The JAAS configuration file is described in Telemetri kanalı JAAS yapılandırması , which includes the sample JAAS configuration file, JAAS.config, shipped with IBM WebSphere MQ Telemetry.

JAAS' ı yapılandırırsanız, kullanıcıların standart JAAS kimlik doğrulama yordamlarını değiştirmeleri için kimlik doğrulaması yapmak üzere bir sınıf yazacaksın.

`Login` sınıfınızı, telemetri (MQXR) hizmet sınıfı yolu tarafından kullanılan sınıf yoluna eklemek için, bir WebSphere MQ `service.env` yapılandırma dosyası sağlayın.

`service.env`içindeki JAAS `LoginModule` için sınıf yolunu ayarlayın. You cannot use the variable, `%classpath%` in `service.env`. `service.env` içindeki sınıf yolu, telemetri (MQXR) hizmet tanımında önceden ayarlanmış olan sınıf yoluna eklenir.

Display the class paths that are being used by the telemetry (MQXR) service by adding `echo set classpath` to `runMQXRService.bat`. Çıkış, `mqxr.stdout`' e gönderilir.

`service.env` dosyası için varsayılan konum şudur:

```
WMQ data directory\service.env
```

Aşağıdaki her kuyruk yöneticisi için bu ayarları `service.env` kütüğüyle geçersiz kılın:

```
WMQ data directory\Qmgrs\qMgrName\service.env
```

Şekil 47 sayfa 161 , örnek `LoginModule.class`' i kullanmak için örnek bir `service.env` dosyasını gösterir.

```
CLASSPATH=WMQ Install Directory\mqxr\samples
```

Not: `service.env` herhangi bir değişken içermemelidir. *`WMQ Install Directory`*' un gerçek değerini değiştirin.

Şekil 47. Örnek `service.env` for Windows

İz

Bir IBM hizmet mühendisi izlemeyi yapılandırmanızı isteyebilir; bkz. “Telemetri (MQXR) hizmetinin izlenmesi” sayfa 162. Konfigrasyonu tanmlanmŸ izleme parametreleri iki ktkten saklanmŸ:

```
WMQ data directory\Qmgrs\qMgrName\mqxr\trace.config
WMQ data directory\Qmgrs\qMgrName\mqxr\mqxrtrace.properties
```

İstemci tarafı günlük dosyaları

IBM WebSphere MQ Telemetry ile birlikte sağlanan Java SE MQTT istemcisinde varsayılan dosya sürekliliği sınıfı, istemci çalışma dizininde *clientIdentifier*`-tcp`*hostNamekapı* ya da *clientIdentifier*`-ssl`*hostNamekapı* adlı bir klasör yaratır. Klasör adı, bağlantı girişiminde kullanılan `hostName` ve `kapı` ' yi belirtir.. Klasör, kalıcılık sınıfı tarafından saklanmış iletiler içeriyor. İletiler başarıyla teslim edildiklerinde silinir.

Bir istemci, temiz bir oturuma sahip bir istemci sona erdiğinde silinir.

İstemci izleme açıksa, biçimlenmemiş günlük, istemci çalışma dizininde saklanan varsayılan olarak olur. İzleme dosyası `mqtt-`*n*`.trc`olarak adlandırılır.

İstemci tarafı yapılandırma dosyaları

Java özellik dosyalarını kullanarak MQTT Java istemcisine ilişkin izleme ve SSL özelliklerini ayarlayın ya da özellikleri programsal olarak ayarlayın. JVM `-D` anahtarını kullanarak özellikleri MQTT Java istemcisine iletin: örneğin,

```
Java -Dcom.ibm.micro.client.mqttv3.trace=c:\\MqttTrace.properties
     -Dcom.ibm.ssl.keyStore=C:\\MyKeyStore.jks
```

Bkz. “MQTT v3 Java istemcisinin izlenmesi” sayfa 164.MQTT istemci kitaplıklarına ilişkin istemci API belgelerine ilişkin bağlantılar için bkz. MQTT istemci programlama başvurusu.

MQTT v3 Java istemcisi neden kodları

Bir MQTT v3 Java istemcisi kural dışı durumu ya da throwable içindeki neden kodlarının nedenlerine bakın.

Çizelge 5. MQTT v3 Java istemcisi neden kodları

Neden Kodu	Değer	Neden
REASON_CODE_BROKER_UNAVAILABLE	3	
REASON_CODE_CLIENT_ALREADY_CONNECTED	32100	İstemci zaten bağlı.
REASON_CODE_CLIENT_ALREADY_DISCONNECTED	32101	İstemcinin bağlantısı zaten kesildi.
REASON_CODE_CLIENT_DISCONNECT_PROHIBITED	32107	`MqttCallback`' da bir yöntem içinden `MqttClient.disconnect` çağrılmaya çalışıldığında ortaya atılmıştır.
REASON_CODE_CLIENT_DISCONNECTING	32102	İstemci şu anda bağlantıyı kesiyor ve yeni bir iş kabul edemiyor.
REASON_CODE_CLIENT_EXCEPTION	0	İstemci bir kural dışı durumla karşılaştı.
REASON_CODE_CLIENT_NOT_CONNECTED	32104	İstemci sunucuya bağlı değil.
REASON_CODE_CLIENT_TIMEOUT	32000	Sunucu, sunucudan yanıt beklerken zaman aşımına uğradı.
REASON_CODE_FAILED_AUTHENTICATION	4	Hatalı bir kullanıcı adı ya da parola nedeniyle, sunucuyla kimlik doğrulaması başarısız oldu.
REASON_CODE_INVALID_CLIENT_ID	2	Sunucu, belirtilen istemci tanıtıcısını reddetti.
REASON_CODE_INVALID_PROTOCOL_VERSION	1	İstenen protokol sürümü sunucu tarafından desteklenmiyor.
REASON_CODE_NO_MESSAGE_IDS_AVAILABLE	32001	Yeni ileti tanıtıcıları kullanılamamasına neden olan iç hata.
REASON_CODE_NOT_AUTHORIZED	5	İstenen işlemi gerçekleştirmek için yetkili değil.
REASON_CODE_SERVER_CONNECT_ERROR	32103	Sunucuyla bağlantı kurulamıyor.
REASON_CODE_SOCKET_FACTORY_MISMATCH	32105	Sunucu URI 'si ve sağlanan SocketFactory eşleşmiyor.
REASON_CODE_SSL_CONFIG_ERROR	32106	SSL yapılandırma hatası.
REASON_CODE_UNEXPECTED_ERROR	6	Beklenmeyen bir hata oluştu.

Telemetri (MQXR) hizmetinin izlenmesi

Telemetri hizmetinin izini başlatmak için bu yönergeleri izleyin, izleme işlemini denetleyen parametreleri ayarlayın ve izleme çıkışını bulun.

Başlamadan önce

İzleme bir destek işlevidir. Bir IBM hizmet mühendisi size telemetri (MQXR) hizmetini izlemenizi isterse, bu yönergeleri izleyin. Ürün belgeleri izleme dosyasının biçimini ya da bir istemciyi hata ayıklamak için nasıl kullanılacağını belgelemez.

Bu görev hakkında

IBM WebSphere MQ izlemesini başlatmak ve durdurmak için IBM WebSphere MQ **strmqtrc** ve **endmqtrc** komutlarını kullanabilirsiniz. **strmqtrc** , telemetri (MQXR) hizmetine ilişkin izlemeyi yakalar. **strmqtrc** kullanıldığında, telemetri hizmeti izleme başlatılmadan önce birkaç saniye içinde bir gecikme süresi vardır. IBM WebSphere MQ izlemesine ilişkin ek bilgi için İzlemenin kullanılmasıbaşlıklı konuya bakın. Diğer bir seçenek olarak, aşağıdaki yordamı kullanarak telemetri (MQXR) hizmetini izleyebilirsiniz:

Yordam

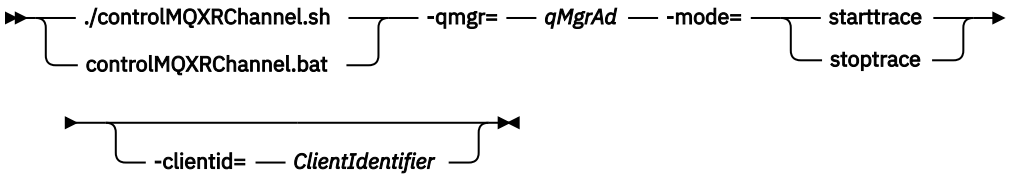
1. İz miktarını ve izleme büyüklüğünü denetlemek için izleme seçeneklerini belirleyin. Seçenekler, **strmqtrc** ya da **controlMQXRChannel** komutuyla başlatılan bir izleme için geçerlidir.

   Aşağıdaki dosyalarda izleme seçeneklerini belirleyin:

   ```
   mqxrtrace.properties
   trace.config
   ```

   Dosyalar şu dizinde yer alıyor:

   - Windowsüzerinde, *WebSphere MQ data directory*\qmgrs*qMgrName* \mqxr.
   - Linuxüzerinde, var/mqm/qmgrs/ *qMgrName*/mqxr.
2. Aşağıdaki dizinde bir komut penceresi açın:
   - Windows sistemlerinde *WebSphere MQ installation directory*\mqxr\bin .
   - Linux sistemlerinde /opt/mqm/mqxr/bin.
3. Bir SYSTEM.MQXR.SERVICE izlemesi başlatmak için aşağıdaki komutu çalıştırın:



   Zorunlu parametreler

   qmgr=*qmgrName*
   Set *qmgrName* to the queue manager name

   mode=starttrace| stoptrace
   İzlemeyi başlatmak için starttrace ' ı ya da izlemeyi sona erdirmek için süreölçer olarak ayarlayın

   İsteğe bağlı parametreler

   clientid=*ClientIdentifier*
   ClientIdentifier 'u bir istemcinin ClientIdentifier ' ine ayarlayın. clientid , izlemeyi tek bir istemciye süzer. Birden çok istemciyi izlemek için izleme komutunu birkaç kez çalıştırın.

   Örneğin:

   ```
   /opt/mqm/mqxr/bin/controlMQXRChannel.sh -qmgr=QM1 -mode=starttrace -clientid=
   problemclient
   ```

Sonuçlar

İzleme çıkışını görüntülemek için, aşağıdaki dizine gidin:

- Windows üzerinde, *WebSphere MQ data directory*\trace.
- Linuxüzerinde, /var/mqm/trace .

İzleme dosyaları mqxr_PPPPP.trcadını taşır; burada PPPPP işlem tanıtıcısıdır.

İlgili başvurular
strmqtrc

MQTT v3 Java istemcisinin izlenmesi

Bir MQTT Java istemcisi izlemesi yaratmak ve çıkışını denetlemek için bu yönergeleri izleyin.

Başlamadan önce

Bu konu yalnızca IBM WebSphere MQ sürüm 7.5.0.0için geçerlidir. Daha sonraki sürümler için Java istemcisini izlemeyi açıklayan bilgi için bkz. "MQTT (Paho) Java istemcisine ilişkin izleme ve hata ayıklama" sayfa 167.

İzleme bir destek işlevidir. Bir IBM hizmet mühendisi, MQTT Java istemcinizi izlemenizi isterse, bu yönergeleri izleyin. Ürün belgeleri izleme dosyasının biçimini ya da bir istemciyi hata ayıklamak için nasıl kullanılacağını belgelemez.

İzleme yalnızca WebSphere MQ Telemetry Java istemcisi için çalışır.

Bu görev hakkında

Not: Örnekler, Windows için kodlanmıştır. Change the syntax to run the examples on Linux[2].

Yordam

1. İzleme yapılandırmasını içeren bir Java özellikler dosyası yaratın.

   Özellikler dosyasında, aşağıdaki isteğe bağlı özellikleri belirtin. Bir özellik anahtarı bir kereden fazla belirtilirse, son geçiş özelliği özelliği belirler.

   a) com.ibm.micro.client.mqttv3.trace.outputName

   İzleme dosyasını yazabilmek için kullanılan dizin. Varsayılan değer olarak istemci çalışma dizini kullanılır. The trace file is called mqtt-*n*.trc.

   ```
   java com.ibm.micro.client.mqttv3.trace.outputName=c:\\MQTT_Trace
   ```

   b) com.ibm.micro.client.mqttv3.trace.count

   Yazılacak izleme kütüklerinin sayısı. Varsayılan değer, sınırsız boyuttan bir dosyadır.

   ```
   java com.ibm.micro.client.mqttv3.trace.count=5
   ```

   c) com.ibm.micro.client.mqttv3.trace.limit

   Yazılacak dosya boyutu üst sınırı, varsayılan değer 500000 'dir. Bu sınır yalnızca, birden çok izleme kütüğü istenirse geçerlidir.

   ```
   java com.ibm.micro.client.mqttv3.trace.limit=100000
   ```

   d) com.ibm.micro.client.mqttv3.trace.client.*clientIdentifier*.status

   İstemci başına izlemeyi açık ya da kapalı konuma getirin. *clientIdentifier*=*ise, izleme programı tüm istemciler için açık ya da kapalı olur. Varsayılan olarak, izleme tüm istemciler için kapatılır.

[2] Java doğru yol sınırlayıcısını kullanır. Bir özellik dosyasında sınırlayıcıyı '/' ya da '\\'olarak kodlayabilirsiniz; '\' kaçış karakteridir.

```
java com.ibm.micro.client.mqttv3.trace.client.*.status=on
```

```
java com.ibm.micro.client.mqttv3.trace.client.Client10.status=on
```

2. Bir sistem özelliği kullanarak izleme özellikleri dosyasını JVM ' ye geçirin.

```
java -Dcom.ibm.micro.client.mqttv3.trace=c:\\MqttTrace.properties
```

3. İstemciyi çalıştırın.
4. İzleme dosyasını ikili kodlamadan metne ya da `.html`biçimine dönüştürün. Aşağıdaki komutu kullanın:

```
java com.ibm.micro.client.mqttv3.internal.trace.TraceFormatter [-i traceFile] [-o
outputFile] [-h] [-d
time]
```

bağımsız değişkenlerin bulunduğu yer:

-?
yardımı görüntüler

-i traceFile
Gereklidir. Giriş dosyasında geçer (örneğin, `mqtt-0.trc`).

-o outputFile
Gereklidir. Çıkış dosyasını tanımlar (örneğin, `mqtt-0.trc.html` ya da `mqtt-0.trc.txt`).

-h
Çıkış HTML olarak. Çıkış dosyaları uzantısı `.html`olmalıdır. Belirlenmezse, çıkış düz metindir.

-d time
Milisaniye cinsinden zaman farkı, (> =) saatinden büyük ya da ona eşit olduğunda * ile bir satırı girintilendirir. HTML çıkışı için geçerli değildir.

Aşağıdaki örnek, izleme dosyasını HTML biçiminde yazacaktır.

```
java com.ibm.micro.client.mqttv3.internal.trace.TraceFormatter -i mqtt-0.trc -o
mqtt-0.trc.html -h
```

The second example will output the trace file as plain text, with any consecutive timestamps that have milliseconds with a difference of 50 or greater indented with an asterisk (*).

```
java com.ibm.micro.client.mqttv3.internal.trace.TraceFormatter -i mqtt-0.trc -o
mqtt-0.trc.txt -d 50
```

Son örnek, izleme dosyasını düz metin olarak çıkışa yazacaktır:

```
java com.ibm.micro.client.mqttv3.internal.trace.TraceFormatter -i mqtt-0.trc -o
mqtt-0.trc.txt
```

C için MQTT istemcisini izleme

Bir MQTT istemcisi C uygulamasını izlemek için `MQTT_C_CLIENT_TRACE` ortam değişkenini ayarlayın.

Başlamadan önce

C izlemesi içinMQTT istemcisi, C kitaplıkları için önceden oluşturulmuş Windows ve Linux MQTT istemcisi için ve kendi oluşturdugunuz iOS kitaplıkları için kullanılabilir.

Bu görev hakkında

`MQTT_C_CLIENT_TRACE` ortam değişkenini, izleme çıkışının içereceği bir dosyaya giden bir yola ayarlayın. İzleme çıkışı dosyaya yazılır.

Yordam

MQTT istemcisi C uygulamanızı çalıştırmadan önce `MQTT_C_CLIENT_TRACE=mqttcclient.log` seçeneğini ayarlayın.

a) For example, modify the sample script in “C için MQTT istemcisiyle çalışmaya başlama” sayfa 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) Komut dosyasını `%sdkroot%/sdk/client/c/samples` dizininden çalıştırın.

Sonuçlar

İzleme çıkışı dosyaları aşağıdaki satırlarla başlar:

```
=============================================================
                    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
```

İlgili görevler

“C için MQTT istemcisiyle çalışmaya başlama” sayfa 25

C kaynağını derleyebileceğiniz herhangi bir platform üzerinde C için örnek MQTT istemcisiyle çalışır ve çalışır. Örnek MQTT Client for C ' yi either IBM MessageSight or IBM WebSphere MQ as the MQTT serverile çalıştırabildiğinizi doğrulayın.

MQTT (Paho) Java istemcisine ilişkin izleme ve hata ayıklama

Varsayılan günlüğe kaydedici, `java.util.logging` (JSR47) olarak bilinen standart Java günlüğe kaydetme olanağını kullanır. Bunu bir yapılanış kütüğü ya da programl olarak kullanarak konfigrasyonunu tanmlayabilirsiniz.

Bu görev hakkında

Not: Paho Java istemcisi yalnızca IBM WebSphere MQ sürümleri 7.5.0.1 ve sonraki sürümleri için geçerlidir. For information describing tracing the Java client in IBM WebSphere MQ version 7.5.0.0, see “MQTT v3 Java istemcisinin izlenmesi” sayfa 164.

Not: İzleme bir destek işlevidir. IBM hizmet mühendisi, MQTT Java istemcinizi izlemenizi isterse, bu yönergeleri izleyin. Ürün belgeleri izleme dosyasının biçimini ya da bir istemciyi hata ayıklamak için nasıl kullanılacağını belgelemez. İzleme yalnızca IBM WebSphere MQ Telemetry Java istemcisi için çalışır.

The simplest method to use a configuration file is to specify its name in the property `java.util.logging.config.file`.

A working property file `jsr47min.properties` is provided in package `org.eclipse.paho.client.mqttv3.logging`

JSR47 günlüğe kaydetme olanağı aşağıdaki şekillerde kullanılabilir:

- Seçilen bir paket kümesinden ileti toplamak için
- Bir günlük düzeyinin aşağısından ve aşağısından ileti toplamak için
- Günlük iletileri için birden çok hedef seçmek için
- Bir dosyaya yazan ve kullanılan dosyaların sayısını ve sayısını denetleyen yerleşik bir günlüğe kaydedici sağlayarak
- Belleğe yazan ve bellek iletilerinin bir tetikleyiciye dayalı olarak yazılmasına olanak sağlayan yerleşik bir günlüğe kaydedici sağlayarak
- MQTT istemci kitaplığını kullanan uygulama JSR47kullanılarak da işlemden geçirilirse, uygulamadaki ve istemci kitaplığından gelen iletiler birbirine karışılır

Hata ayıklama bilgilerinin toplanmasına yardımcı olmak için bir yardımcı program sınıfı sağlanır. Bu sınıf, daha önce açıklanan günlük ve izleme iletilerini içerir, ancak Java sistem özellikleri ve Paho istemcisinin içinden değişkenlerin değeri gibi bilgileri toplayabilir.

The debug facility is provided in the public class Debug, that is part of the package `org.eclipse.paho.client.mqttv3.util`. Hata ayıklama eşgörünümü, hem zamanuyumsuz hem de zamanuyumlu MQTT istemci nesnelerine ilişkin getDebug() yöntemi kullanılarak elde edilebilir.

Örneğin:

```
MqttClient cl = new MqttClient();
Debug d = cl.getDebug();
```

dumpClientdebug (hata ayıklama) yöntemi (), hata ayıklama bilgileri üst sınırını içerir. Günlük olanağı, tam hata ayıklama bilgilerini yakalamak için etkinleştirilmelidir; bu bilgiler buna yazılır. Tam hata ayıklama bilgilerini yakalamak için, sorunun oluştuğu bilindiğinde bir döküm yöntemi çağırın; örneğin, belirli bir kural dışı durum oluştuktan sonra.

Yordam

1. Bir yapılandırma dosyası oluşturun ya da sağlanan jsr47min.properties dosyasını kullanın.

   Sağlanan özellikler dosyasını kullanıyorsanız, itme tetikleyicinin hata düzeyini düzeltmek için ayarlandığından emin olun. Varsayılan olarak bu, önemli bir düzey hatasına ayarlanır; ancak, bir hata

ortaya çıkana kadar izlemeyi belleğimde tutmak yerine sürekli olarak dosyaya yazmak gerekebilir. Bunu yapmak için şunu değiştirin:

```
java.util.logging.MemoryHandler.push=SEVERE
```

-

```
java.util.logging.MemoryHandler.push=ALL
```

2. Bir sistem özelliği kullanarak izleme yapılandırma dosyasını JVM ' ye geçirin.

   jsr4min.properties dosyasını kullanıyorsanız, şu adresi kullanın:

   ```
   java -Djava.util.logging.config.file=C:\temp\jsr47min.properties
   ```

3. İstemciyi çalıştırın.

Sonuçlar

Bir kural dışı durum ya da sorun ortaya çıktığında, Paho Hata Ayıklama sınıfı, yapılandırılan dosya hedefine bellek içi izini yazar.

İzleme otomatik olarak dosyaya yazıldığı için yazılmaz; bu durum yalnızca, push tetikleyicisini vurduğunda ya da hata ayıklama sınıfı izlemenin yazılmasına neden olduğunda oluşur. İkinci durum, uygulama kodunda değişiklik yapılmasını gerektirebilir.

Her satır, yaratıldığı gibi dosya işleyiciye yazılır. Bir FileHandlerolanağını yapılandırarak iletilerin yazıldığı biçimi denetleyebilirsiniz. A custom file handler is provided with Paho that writes out more than the SimpleHandler and less than the XMLHandler provided with the JRE. Paho günlük biçimleyicisini kullanan izleme kayıtları aşağıdaki biçimlerden biri olabilir:

```
Level   Data and Time   Class   Method   Thread   clientID   Message
```

Örnek

A working property file `jsr47min.properties` is provided. Bu dosya, Paho MQTT istemcisine ilişkin sorunların çözülmesine yardımcı olan izlemeyi toplamak için önerilen bir yapılandırma içerir. İzlemeyi, başarım üzerinde en az etkiye sahip belleğde sürekli olarak toplanacak şekilde yapılandırır. İtme tetikleyicisi oluştuğunda ya da itmek için belirli bir istek yapıldığında, bellek içi izleme konfigürasyonu tanımlanmış hedef işleyiciye itilir. Varsayılan itme tetikleyicisi, bozuk bir bağlantı olan bir Severe düzey iletidir. Varsayılan olarak, bellede toplanan izleme, belirtilen dosyaya bu noktada yazılır. Varsayılan olarak bu dosya standart `java.util.logging.FileHandler`dosyasıdır. Paho Hata Ayıklama sınıfını kullanarak bellek izini hedefine itebilirsiniz.

Full details of JSR47 can be found in the Javadoc for package 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
```

Sonraki adım

İzlemeyi programlı olarak toplamak için, hata ayıklama bilgilerinin toplanmasına yardımcı olmak için bir yardımcı program sınıfı sağlanır. Bu sınıf, daha önce açıklanan günlük ve izleme iletilerini içerir, ancak Java sistem özellikleri ve Paho istemcisinin içinden değişkenlerin değeri gibi bilgileri toplayabilir.

The debug facility is provided in the public class Debug, that is part of the package `org.eclipse.paho.client.mqttv3.util`. Hata ayıklama eşgörünümü, hem zamanuyumsuz hem de zamanuyumlu MQTT istemci nesnelerine ilişkin getDebug() yöntemi kullanılarak elde edilebilir.

Örneğin:

```
MqttClient cl = new MqttClient();
Debug d = cl.getDebug();
```

dumpClientdebug (hata ayıklama) yöntemi (), hata ayıklama bilgileri üst sınırını içerir. Günlük olanağı, tam hata ayıklama bilgilerini yakalamak için etkinleştirilmelidir; bu bilgiler buna yazılır. Tam hata ayıklama bilgilerini yakalamak için, sorunun oluştuğu bilindiğinde bir döküm yöntemi çağırın; örneğin, belirli bir kural dışı durum oluştuktan sonra.

MQTT JavaScript istemcisinin izlenmesi

Bağlı istemci nesnesindeki yöntemleri çağırmak için istemci web uygulamasını değiştirerek izleme toplamak için JavaScript istemcisini kullanabilirsiniz.

Bu görev hakkında

İzlemeyi toplamak için aşağıdaki yöntemleri kullanabilirsiniz:

- `client.startTrace()` istemciye ilişkin izlemeyi başlatır.
- `client.stopTrace()` istemciye ilişkin izlemeyi durdurur.
- `client.getTraceLog()` , yürürlükteki izleme arabelleğini döndürür.

You can output the trace buffer to send to IBM Software Support. Bunu yapmak için bir dizi yol var. Örnekte, izlemenin başlatılmakta olduğu, sonra hem konsola, hem de belirli bir e-posta adresine gönderilen çıkışın ve sonunda izleme durdurulduğu gösterilmektedir.

```
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();
 };
```

Örnek çıkış:

```
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 System requirements for using SHA-2 cipher suites with MQTT clients

- IBM, SR13 ' den başlayarak Java 6 için, MQTT kanallarınızı ve istemci uygulamalarınızı güvenli kılmak için SHA-2 şifreleme takımlarını kullanabilirsiniz. However, SHA-2 cipher suites are not enabled by default until Java 7 from IBM, SR4 onwards, so in earlier versions you must specify the required suite. Kendi JRE ' nizle bir MQTT istemcisi çalıştırıyorsanız, bunun SHA-2 şifreleme takımlarını desteklediğinden emin olmanız gerekir. İstemci uygulamalarınızın SHA-2 şifreleme takımlarını kullanması için, istemcinin SSL bağlamını Transport Layer Security (TLS) sürüm 1.2' yi destekleyen bir değere ayarlaması gerekir.

For Java 7 from IBM, SR4 onwards, SHA-2 cipher suites are enabled by default. For Java 6 from IBM, SR13 and later service releases, if you define an MQTT channel without specifying a cipher suite, the channel will not accept connections from a client using a SHA-2 cipher suite. SHA-2 şifreleme takımlarını kullanmak için, kanal tanımında gerekli olan takımı belirtmeniz gerekir. This makes the MQTT server enable the suite before making connections. Ayrıca, belirtilen takımı kullanan istemci uygulamalarının bu kanala bağlanabileceği anlamına da gelir.

Javaiçin MQTT istemcisiiçin de benzer bir sınırlama vardır. İstemci kodu IBM' den bir Java 1.6 JRE üzerinde çalışıyorsa, gerekli olan SHA-2 şifreleme takımları belirtik olarak etkinleştirilmelidir. Bu paketleri kullanmak için, istemci, SSL bağlamını Transport Layer Security (TLS) protokolünün 1.2 sürümünü destekleyen bir değere de ayarlamalıdır. Örneğin:

```
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);
```

Haziran 2013 'te olduğu gibi, Internet Explorer 10, MQTT messaging client for JavaScript ile çalışan ve TLS 1.2 iletişim kuralını da destekleyen tek tarayıcıdır; bu nedenle, JavaScript istemcisiyle SHA-2 bağlantıları yapmak istiyorsanız kullanabileceğiniz tek tarayıcı budur.

Şu anda desteklenmekte olan şifreleme takımlarının listesi için ilgili bağlantıları kullanın.

İlgili kavramlar

“SSL kullanan istemci kimlik doğrulaması için MQTT istemcisi yapılanışı” sayfa 103

SSL kullanan MQTT istemcisini doğrulamak için, istemci SSL kullanarak bir telemetri kanalına bağlanır. SSL istemcilerini doğrulamak için yapılandırılmış bir telemetri kanalına karşılık gelen bir TCP kapısı belirtmelidir.

“SSL kullanan kanal kimlik doğrulaması için MQTT istemcisi yapılanışı” sayfa 106
Telemetri kanalını SSL kullanarak doğrulamak için, istemci, telemetri kanalına SSL kullanarak bağlanmalıdır. Bu, SSL için yapılandırılmış bir telemetri kanalına karşılık gelen bir kapı belirtmelidir. Yapılandırma, sunucunun özel olarak imzalanmış dijital sertifikasını içeren bir geçiş tümcecik korumalı anahtar deposu içermelidir.

V 7.5.0.1 SSL üzerinden mobil ileti sistemi web uygulamaları için tarayıcı desteğindeki kısıtlamalar

Farklı tarayıcılar üzerinde farklı tarayıcılar arasında yetenek farklılıkları vardır. Understanding these differences helps you configure your apps, certificate authorities (CAs) and client certificates to connect using the MQTT messaging client for JavaScript over SSL and WebSockets.

Mobile messaging using JavaScript over SSL is fairly new, so it is not surprising that different browser and platform combinations have implemented the capability in slightly different ways, and to different extents. Aşağıdaki tablo, şu anda ne işe yaradığını ve tarayıcı (Firefox, Chrome, Internet Explorer,andve Safari) ve platformlarının (Windows, Linux, Mac, iOSve Android) birleşimleri için çalışmadığı hakkında genel bir bakış sağlar.

Çizelge 6. Platform ve tarayıcı ile SSL desteği. Her tarayıcı ve platform birleşimi için tablo, SSL Anonim ve Anonim Olmayan bağlantıların desteklenip desteklenmediğini ve tarayıcının tüm Sertifika Yetkilileri (CA ' lar) ve istemci sertifikaları ile birlikte çalıştığı kapsamı belirtir.

Tarayıcı	**SSL Desteği (Y/N)**	**SSL, herhangi bir CA (Y/N) ile çalışır**	**Daha fazla bilgi**
Firefox masaüstü.	SSL Anonim-Evet SSL Anonim değil-Evet	SSL Anonim-Evet SSL Anonim değil-Evet	CA ' yı ve istemci sertifikasını tarayıcıya ekleyin. Firefox , kendi sertifika deposunu kullanır. Bir CA sertifikasını içe aktarmak için, **Araçlar** > **Seçenekler** > **Gelişmiş** > **Şifreleme** > **Sertifikaları Görüntüle** > **Yetkililer** > **Al**seçeneklerini tıklatın Bir istemci sertifikasını almak için, **Araçlar** > **Seçenekler** > **Gelişmiş** > **Şifreleme** > **Sertifikaları Görüntüle** > **Sertifikalarınız** > **İçe Aktar**seçeneklerini tıklatın Güvenli bir bağlantıyı etkinleştirmek için URL ' de `https://` değerini belirtin. Firefox , size otomatik olarak bir sertifika seçme ya da size her zaman sorarak bir sertifika seçme seçeneği sunar. Firefox , ayrıca size SSL 3.0 ya da TLS 1.0kullanma seçeneğini de verir; her ikisinin de seçildiğinden emin olun.

Çizelge 6. Platform ve tarayıcı ile SSL desteği. Her tarayıcı ve platform birleşimi için tablo, SSL Anonim ve Anonim Olmayan bağlantıların desteklenip desteklenmediğini ve tarayıcının tüm Sertifika Yetkilileri (CA ' lar) ve istemci sertifikaları ile birlikte çalıştığı kapsamı belirtir. *(devamı var)*

Tarayıcı	**SSL Desteği (Y/N)**	**SSL, herhangi bir CA (Y/N) ile çalışır**	**Daha fazla bilgi**
Chrome masaüstü.	SSL Anonim-Evet SSL Anonim değil-Evet	SSL Anonim-Evet SSL Anonim değil-Evet	CA ve istemci sertifikasını, diğer yazılımlarla paylaşılan işletim sistemi sertifika deposuna eklemek için tarayıcıyı kullanın. Bir CA sertifikasını içe aktarmak için, **Ayarlar** > **Gelişmiş Ayarları Göster** > **Sertifikaları Yönet** > **Güvenilen Kök Sertifika Yetkilileri** > **Al**seçeneklerini tıklatın. Bir istemci sertifikasını almak için, **Ayarlar** > **Gelişmiş Ayarları Göster** > **Sertifikaları Yönet** > **Kişisel** > **İçe Aktar**seçeneklerini tıklatın Güvenli bir bağlantıyı etkinleştirmek için URL ' de `https://` değerini belirtin. Chrome sizden birkaç seçenek ister; Anonymous ya da Anonymous olmayan bir bağlantı yapılandırıp yapılandırmadığınızı bağlı olarak doğru olanı seçin.

Çizelge 6. Platform ve tarayıcı ile SSL desteği. Her tarayıcı ve platform birleşimi için tablo, SSL Anonim ve Anonim Olmayan bağlantıların desteklenip desteklenmediğini ve tarayıcının tüm Sertifika Yetkilileri (CA ' lar) ve istemci sertifikaları ile birlikte çalıştığı kapsamı belirtir. *(devamı var)*

Tarayıcı	**SSL Desteği (Y/N)**	**SSL, herhangi bir CA (Y/N) ile çalışır**	**Daha fazla bilgi**
Internet Explorer.	SSL Anonim-Evet SSL Anonim değil-Evet	SSL Anonim-Evet SSL Anonim değil-Evet	Anonim olmayan bir SSL bağlantısı yaparsanız, doğru istemci sertifikasını seçmeniz istenir. Internet Explorer , diğer yazılımlarla paylaşılan Windows sertifika deposunu kullanır. Bir CA sertifikasını içe aktarmak için, **Araçlar** > **İnternet Seçenekleri** > **İçerik** > **Sertifikalar** > **Güvenilen Kök Sertifika Yetkilileri** > **Al**seçeneklerini tıklatın. Bir istemci sertifikasını içe aktarmak için **Araçlar** > **İnternet Seçenekleri** > **İçerik** > **Sertifikalar** > **Kişisel** > **İçe Aktar**öğelerini tıklatın.
Safari masaüstü.	SSL Anonim-Evet SSL Anonim değil-Evet	SSL Anonim-Evet SSL Anonim değil-Evet	CA ve istemci sertifikasını, diğer yazılımlarla paylaşılan işletim sistemi sertifika deposuna eklemek için tarayıcıyı kullanın.

Çizelge 6. Platform ve tarayıcı ile SSL desteği. Her tarayıcı ve platform birleşimi için tablo, SSL Anonim ve Anonim Olmayan bağlantıların desteklenip desteklenmediğini ve tarayıcının tüm Sertifika Yetkilileri (CA ' lar) ve istemci sertifikaları ile birlikte çalıştığı kapsamı belirtir. *(devamı var)*

Tarayıcı	**SSL Desteği (Y/N)**	**SSL, herhangi bir CA (Y/N) ile çalışır**	**Daha fazla bilgi**
AndroidüzerindeFirefox	SSL Anonim-Evet SSL Anonim değil-Evet	SSL Anonim-Evet SSL Anonim-Hayır	Anonim Olmayan: Client certificates do not work, because you cannot meet the requirement to add your CA to the list in Firefox. Bir istemci sertifikasını içe aktarmak için **Ayarlar** > **Güvenlik** > **Kimlik Bilgileri Deposu**simgesini tıklatın. Sertifikanız, listede güvenilir bir sertifika kuruluşu tarafından imzalandıysa, güvenli bir bağlantı yapabilirsiniz.
AndroidüzerindeChrome	SSL Anonim-Evet SSL Anonim değil-Evet	SSL Anonim-Evet SSL Anonim-Hayır	Anonim Olmayan: Client certificates do not work, because you cannot meet the requirement to add your CA to the list in Chrome. **Not:** Google plan to support this in Version 27 of Chrome. Sürüm 18 'den bu yana açık bir hata ortaya çıktı. Bir istemci sertifikasını içe aktarmak için **Ayarlar** > **Güvenlik** > **Kimlik Bilgileri Deposu**simgesini tıklatın. Sertifikanız, listede güvenilir bir sertifika kuruluşu tarafından imzalandıysa, güvenli bir bağlantı yapabilirsiniz.

Çizelge 6. Platform ve tarayıcı ile SSL desteği. Her tarayıcı ve platform birleşimi için tablo, SSL Anonim ve Anonim Olmayan bağlantıların desteklenip desteklenmediğini ve tarayıcının tüm Sertifika Yetkilileri (CA ' lar) ve istemci sertifikaları ile birlikte çalıştığı kapsamı belirtir. *(devamı var)*

Tarayıcı	SSL Desteği (Y/N)	SSL, herhangi bir CA (Y/N) ile çalışır	Daha fazla bilgi
iOSüzerindeSafari	SSL Anonim-Evet SSL Anonim değil-Evet	SSL Anonim-Evet SSL Anonim-Hayır	Anonim Olmayan: Aygıt, CA sertifikası aynı anda kurulu olsa da istemci sertifikasına güvenmez. Safari , aygıt sertifikası deposunu kullanır. Bu mağazaya (içe) aktarmak için **Ayarlar** > **Genel** > **Profil**' i tıklatın ve CA ya da istemci sertifikasını bir web sayfasından gönderin ya da kendinize e-posta ile gönderin.
iOSüzerindeChrome	SSL Anonim-Evet SSL Anonim-Hayır	SSL Anonim-Hayır SSL Anonim-Hayır	Anonim: iOS sistem kök deposuna yalnızca Apple uygulamaları erişebilir. Bu nedenle Chrome , ekleyemeydiğiniz kendi CA listesini kullanmalıdır. Anonim Olmayan: CA ' nizi listeye ekleme gereksinimini karşılayamadığınız için, istemci sertifikaları çalışmaz.

İlgili görevler

“MQTT messaging client for JavaScript ile SSL ve WebSocketsarasındaki bağlantı kurulması” sayfa 76
Connect your web app securely to IBM WebSphere MQ by using the MQTT messaging client for JavaScript sample HTML pages with SSL and the WebSocket protocol.

İlgili bilgiler

Mozilla: (SSL) Firefox, Android CA depolamayı kullanır mı, yoksa kendi mi?

Krom: Sayı 134418-İstemci sertifikası desteğini uygula

Https sitesi, ie10üzerinde güvenilir olmayan sertifikan ile açılamıyor

Sorun çözülüyor: MQTT istemcisi bağlanmıyor

Telemetry (MQXR) hizmetine bağlanmamakta başarısız olan bir MQTT istemci programının sorununu çözün.

Başlamadan önce

Sorun sunucuda mı, istemcide mi, yoksa bağlantıyla mı ilgili? Kendi MQTT v3 iletişim kuralı işleme istemcinizi ya da C ya da Java WebSphere MQTT istemcilerini kullanan bir MQTT istemcisi uygulaması yazdınız mı?

Sunucuda WebSphere MQ Telemetry ile birlikte verilen doğrulama uygulamasını çalıştırın ve telemetri kanalı ile telemetri (MQXR) hizmetinin doğru şekilde çalıştığından emin olun. Daha sonra doğrulama uygulamasını istemciye aktarın ve doğrulama uygulamasını orada çalıştırın.

Bu görev hakkında

Bir MQTT istemcisinin bağlanmamasının ya da telemetri sunucusuna bağlı olmadığı sonucuna varabileceğiniz bir dizi nedeni vardır.

Yordam

1. Telemetri (MQXR) hizmetinin `MqttClient.Connect`' e geri döndürdüğü neden kodundan çıkarımlar çizilebileceğini göz önünde bulundurun. Bağlantı başarısızlığı tipi nedir?

Seçenek	Açıklama
REASON_CODE_INVALID_PROTOCOL_VERSION	Yuva adresinin bir telemetri kanalına karşılık geldiğinden emin olun ve başka bir aracı için aynı yuva adresini kullanmadığınızdan emin olun.
REASON_CODE_INVALID_CLIENT_ID	İstemci tanıtıcısının 23 byte 'tan uzun olmadığını ve yalnızca aralıklardan gelen karakterleri içerdiğini doğrulayın: `A-Z, a-z, 0-9, './_%`
REASON_CODE_INVALID_DESTINATION	İstemci tanıtıcısının kuyruk yöneticisi adıyla aynı olmadığından emin olun.
REASON_CODE_SERVER_CONNECT_ERROR	Telemetri (MQXR) hizmetinin ve kuyruk yöneticisinin olağan bir şekilde çalıştığından emin olun. Yuva adresinin başka bir uygulamaya ayrılmamış olup olmadığını denetlemek için **netstat** seçeneğini kullanın.

   IBM WebSphere MQ Telemetry tarafından sağlanan kitaplıklardan birini kullanmak yerine bir MQTT istemci kitaplığı yazdıysanız, `CONNACK` dönüş koduna bakın.

   Bu üç hatadan, istemcinin telemetri (MQXR) hizmetine bağlı olduğunu, ancak hizmetin bir hata bulduğunu ortaya çıkartabilirsiniz.

2. Telemetri (MQXR) hizmeti yanıt vermediğinde, müşterinin ürettiği neden kodlarından hangi çıkarımlar çizilebileceğini göz önünde bulundurun:

Seçenek	Açıklama
REASON_CODE_CLIENT_EXCEPTION **REASON_CODE_CLIENT_TIMEOUT**	Sunucuda FDC dosyası olup olmadığını görmek için bkz. “Sunucu tarafındaki günlükler” sayfa 159. Telemetri (MQXR) hizmeti, istemcinin zamanaşımına uğradığını algıladığında, ilk hata verilerini yakalama (FDC) dosyası yazar. Bu dosya, bağlantı beklenmedik bir şekilde kesildiğinde FDC dosyası yazar.

   Telemetri (MQXR) hizmeti istemciye yanıt vermemiş olabilir ve istemcideki zamanaşımı süresi sona ermiş olabilir. The WebSphere MQ Telemetry Java client only hangs if the application has set an indefinite timeout. `MqttClient.Connect` için ayarlanan zamanaşımı süresi tanılanmamış bir bağlantı sorunuyla sona erdikten sonra istemci bu kural dışı durumlardan birini yayınlıyor.

Bağlantı hatasıyla ilintili bir FDC dosyası bulamazsanız, istemciyi sunucuya bağlanmaya çalıştıranı çıkartamazsınız:

a) İstemcinin bir bağlantı isteği gönderdiğini doğrulayın.

Check the TCPIP request with a tool such as **tcpmon**, available from https://java.net/projects/tcpmon

b) İstemci tarafından kullanılan uzak yuva adresi, telemetri kanalı için tanımlanan yuva adresi ile eşleşiyor mu?

IBM WebSphere MQ Telemetry ile birlikte sağlanan Java SE MQTT istemcisinde varsayılan dosya sürekliliği sınıfı, istemci çalışma dizininde *clientIdentifier*-tcp*hostNamekapı* ya da *clientIdentifier*-ssl*hostNamekapı* adlı bir klasör yaratır. Klasör adı, bağlantı girişiminde kullanılan hostName ve kapı ' yi belirtir.; bkz. "İstemci tarafı günlük dosyaları" sayfa 161.

c) Uzak sunucu adresine ping komutu gönderebiliyor musunuz?

d) Sunucuda **netstat** , telemetri kanalını istemcinin de bağlıyor olduğu kapıda çalışıyor mu?

3. Telemetri (MQXR) hizmetinin istemci isteğinde bir sorun olup olmadığını denetleyin.

The telemetry (MQXR) service writes errors it detects into mqxr.log, and the queue manager writes errors into AMQERR01.LOG; see

4. Başka bir istemci çalıştırılarak sorunu yalıtmaya çalışın.

- Aynı telemetri kanalını kullanarak MQTT örnek uygulamasını çalıştırın.
- Bağlantıyı doğrulamak için **wmqttSample** GUI istemcisini çalıştırın. SupportPac IA92olanağını karşıdan yükleyerek **wmqttSample** ' i edinin.

Not: IA92 ' ın eski sürümleri MQTT v3 Java istemcisi kitaplığını içermez.

Ağ bağlantısıyla ilgili belirsizlikleri ortadan kaldırmak için sunucu platformunda örnek programları çalıştırın ve daha sonra, istemci altyapısındaki örnekleri çalıştırın.

5. Denetlemeniz gereken diğer şeyler:

a) Aynı anda bağlantı kurmaya çalışan on binlerce MQTT istemcisi var mı?

Telemetri kanalları, gelen bağlantıların arka günlüğünü arabelleğe almak için bir kuyruğa sahiptir. Bağlantılar, saniyenin 10.000 'i fazlalık olarak işlenir. Arka günlük arabelleğindeki büyüklük, IBM WebSphere MQ Explorer 'da telemetri kanalı sihirbazı kullanılarak yapılandırılabilir. Varsayılan boyutu 4096 'tır. Birikim günlüğünün düşük bir değere yapılandırılmadığından emin olun.

b) Telemetri (MQXR) hizmeti ve kuyruk yöneticisi hala çalışıyor mu?

c) İstemcinin, TCPIP adresini değiştirmiş olan yüksek kullanılabilirlikli bir kuyruk yöneticisine bağlı olması mı?

d) Bir güvenlik duvarı, giden ya da dönen veri paketlerine seçmeli olarak süzgeç uygulıyor mu?

Sorun çözülüyor: MQTT istemci bağlantısı atıldı

Kısa ya da uzun bir süre başarıyla bağlanıp çalıştıktan sonra, bir istemcinin beklenmeyen ConnectionLost kural dışı durumları yayınlanmasına neden olan nedir öğrenin.

Başlamadan önce

MQTT istemcisi başarıyla bağlandı. Müşteri uzun bir süre ayakta kalabilirdi. İstemciler aralarındaki yalnızca kısa bir aralıkla başlıyorsanız, bağlantı kurma arasındaki süre ile bağlantı atılma süresi kısa olabilir.

Atılan bir bağlantının başarıyla yapılmış bir bağlantıdan ayırt edilmesi zor değildir ve daha sonra atılır. Atılan bir bağlantı, MqttCallback.ConnectionLost yöntemini çağırarak MQTT istemcisi tarafından tanımlanır. Yöntem, yalnızca bağlantı başarıyla kurulduktan sonra çağrılır. Belirti, eksi alındı bildirimi aldıktan sonra ya da zamanaşımına uğradıktan sonra kural dışı durum yayınlaması için MqttClient.Connect değerine farklıdır.

MQTT istemcisi uygulaması, IBM WebSphere MQtarafından sağlanan MQTT istemci kitaplıklarını kullanmıyorsa, belirti istemciye bağlıdır. MQTT v3 protokolünde, belirti sunucuya gönderilen bir isteğin zamansız yanıtı ya da TCP/IP bağlantısının başarısız olmayışıdır.

Bu görev hakkında

MQTT istemcisi `MqttCallback.ConnectionLost` çağrısı, pozitif bir bağlantı alındı bildirimi alındıktan sonra karşılaşılan sunucu tarafı sorunlarına yanıt olarak atılabilir bir özel durumlacallsçağrılır. When an MQTT client returns from `MqttTopic.publish` and `MqttClient.subscribe` the request is transferred to an MQTT client thread that is responsible for sending and receiving messages. Sunucu tarafındaki hatalar, `ConnectionLost` geri bildirme yönteminde bir throwable kural dışı durumu iletilerek zamanuyumsuz olarak raporlanır.

Telemetri (MQXR) hizmeti, bağlantıyı bıraktıysa her zaman ilk hata verilerini yakalama dosyası yazar.

Yordam

1. Aynı `ClientIdentifier`' ı kullanan başka bir istemci başlatıldı mı?

   İkinci bir istemci başlatılırsa ya da aynı `ClientIdentifier`kullanılarak aynı istemci yeniden başlatılırsa, ilk istemciyle olan ilk bağlantı atılır.
2. İstemci, yayınlama ya da abone olma yetkisi olmayan bir konuya erişmiş mi?

   Any actions the telemetry service takes on behalf of a client that return `MQCC_FAIL` result in the service dropping the client connection.

   Neden kodu istemciye döndürülemez.

   - İstemcinin bağlı olduğu kuyruk yöneticisi için `mqxr.log` ve `AMQERR01.LOG` dosyalarında günlük iletileri olup olmadığını görmek için bkz. “Sunucu tarafındaki günlükler” sayfa 159.
3. TCP/IP bağlantısı kesildi mi?

   Bir güvenlik duvarı, bir TCPIP bağlantısını etkin değil olarak işaretlemek için düşük bir zamanaşımı ayarına sahip olabilir ve bağlantıyı düşürebilir.

   - Shorten the inactive TCPIP connection time using `MqttConnectOptions.setKeepAliveInterval`.

Sorun çözümleniyor: MQTT uygulamasında iletiler kaybedildi

Bir iletiyi kaybetmenin sorununu çözün. İleti kalıcı değil, yanlış yere gönderildi ya da hiçbir zaman gönderilmedi mi? Yanlış bir şekilde kodlanmış bir istemci programı iletileri kaybedebilir.

Başlamadan önce

Gönderdiğin mesaj ne kadar eminsin? Kayboldu mu? İleti alınmadığı için bir iletinin kaybolduğunu erteleyemez misiniz? İleti bir yayınsa, hangi ileti kaybedilir: yayıncı tarafından gönderilen ileti ya da aboneye gönderilen ileti? Ya da abonelik kaybedildi ve aracı, aboneye bu aboneliğin yayınlarını göndermiyor mu?

Çözüm, dağıtılmış yayınlama/abone olma, kümeleri ya da yayınlama/abone olma sıradüzenlerini içeriyorsa, kayıp bir iletinin görünmesine neden olabilecek çok sayıda yapılandırma sorunu vardır.

"En az bir kez" ya da "En çok bir kez" hizmet kalitesi ile bir ileti gönderdiyseniz, kaybolduğunu düşündüğünüz iletinin beklediğiniz şekilde teslim edilmemesinin olasılığı yüksektir. İletinin yanlış bir şekilde sistemden silindiği olasılığı düşük. Yayını ya da beklediğiniz aboneliği oluşturamamış olabilir.

Kayıp iletilerin sorun saptamasını yapmak için yaptığınız en önemli adım, iletinin kaybolduğunu doğrulamandır. Senaryoyı yeniden yaratın ve daha fazla ileti kaybedin. "En az bir kez" ya da "En çok bir kez" hizmet kalitesi, iletilerin atılması gereken tüm iletileri ortadan kaldırmak için kullanılır.

Bu görev hakkında

Kayıp bir mesajı teşhis etmek için dört bacak vardır.

1. "Yangın ve unut" mesajları, tasarlandığı gibi çalışıyor. "Yangın ve unut" mesajları bazen sistem tarafından atılır.
2. Yapılandırma: Dağıtılmış bir ortamdaki doğru yetkilerle yayınlama/abone olma ayarı basit değildir.
3. İstemci programlama hataları: İleti tesliminin sorumluluğu yalnızca IBMtarafından yazılan kodun sorumluluğu değildir.
4. Tüm bu olanakları tüketmiş olduğunuz takdirde, IBM hizmetini içermeye karar verebilirsiniz.

Yordam

1. Kayıp ileti "Yangın ve Unutun" hizmet kalitesine sahipse, "En az bir kez" ya da "En fazla bir kez" hizmet kalitesine ayarlayın. İletiyi yeniden kaybetmeyi deneyin.
   - "Fire and forget" (Fire and forget) hizmetine gönderilen iletiler, IBM WebSphere MQ tarafından bir dizi koşulda uzağa atılır:
     – İletişim kaybı ve kanal durduruldu.
     – Kuyruk yöneticisi sona erdirilsin.
     – Çok sayıda ileti.
   - "Yangın ve unutun" iletilerinin teslimatı, TCP/IP ' nin güvenilirliğine bağlıdır. TCP/IP, teslim edilinceye kadar veri paketlerini yeniden göndermeye devam eder. TCP/IP oturumu bozuksa, "Fire and forget" (Fire ve forget) hizmet kalitesine sahip iletiler kaybedilir. Oturum istemci ya da sunucu kapatılarak kapatılabilir, bir iletişim sorunu ya da oturumun bağlantısını kesmiş bir güvenlik duvarı olabilir.
2. Teslim edilmemiş iletileri "En az bir kez" ya da "En çok bir kez" hizmet kalitesiyle yeniden göndermek için, istemcinin önceki oturumu yeniden başlattıktan sonra yeniden başlatıldığını doğrulayın.
   a) If the client app is using the Java SE MQTT client, check that it sets `MqttClient.CleanSession` to `false`
   b) Farklı istemci kitaplıkları kullanıyorsanız, bir oturumun doğru olarak başlatılıp başlatılmadığından emin olun.
3. İstemci uygulamasının aynı oturumu yeniden başlatıyor olduğunu ve yanlışlıkla farklı bir oturum başlatmadığına bakın.

   Aynı oturumu yeniden başlatmak için `cleanSession = false`ve `Mqttclient.clientIdentifier` ve `MqttClient.serverURI` , önceki oturumla aynı olmalıdır.
4. Bir oturum zamanından önce kapanırsa, iletinin yeniden göndermek için istemcinin saklama deposunda kullanılabilir olup olmadığını denetleyin.
   a) İstemci uygulaması Java SE MQTT istemcisini kullanıyorsa, iletinin kalıcı saklama klasöründe saklanmakta olduğunu doğrulayın; bkz. “İstemci tarafı günlük dosyaları” sayfa 161
   b) Farklı istemci kitaplıkları kullanıyorsanız ya da kendi kalıcılık mekanizmanızı uyguladıysanız, doğru biçimde çalıştığından emin olun.
5. Iletilmeden önce kimsenin iletiyi silmemesine dikkat edin.

   MQTT istemcilerine teslim bekleyen teslim edilmemiş iletiler `SYSTEM.MQTT.TRANSMIT.QUEUE`içinde depolanır. telemetri sunucusuna teslim edilmeyi bekleyen iletiler istemci kalıcılık mekanizması tarafından depolanır; bkz. MQTT istemcilerinde ileti kalıcılığı.
6. İstemcinin, almayı beklediği yayına ilişkin bir aboneliği olup olmadığını denetleyin.

   List subscriptions using WebSphere MQ Explorer, or by using **runmqsc** or PCF commands. Tüm MQTT istemci abonelikleri adlandırılır. Bu formlara bir ad verilir: *`ClientIdentifier`*:*`Topic name`*
7. Yayınlayıcının yayınlama yetkisi olup olmadığını ve yayın konusuna abone olmak için aboneyi denetlemeniz gerekir.

```
dspmqaut -m qMgr -n topicName -t topic -p user ID
```

Kümelenmiş bir yayınlama/abone olma sisteminde, abonenin bağlı olduğu kuyruk yöneticisinde konu üzerinde yetkilendirilmesi gerekir. Yayının yayınlandığı kuyruk yöneticilikindeki konuya abone olmak için abonenin yetkilendirilmesi gerekmez. Kuyruk yöneticilerine ilişkin kanallar, yetkili aboneliği geçirmek ve yayını iletmek için doğru bir şekilde yetkilendirilmelidir.

Aynı aboneliği oluşturun ve IBM WebSphere MQ Explorer 'ı kullanarak bu aboneliği yayınlayın. İstemci yardımcı programını kullanarak uygulama istemcisi yayınınızı ve abone olarak abone olmanızın benzetimini yapın. Yardımcı programı IBM WebSphere MQ Explorer 'dan başlatın ve kullanıcı kimliğini istemci uygulamanızın benimsemiş olduğu kullanıcı kimliğiyle eşleşecek şekilde değiştirin.

8. Abonenin, yayını SYSTEM.MQTT.TRANSMIT.QUEUE' e koyma izni olup olmadığını denetleyin.

```
dspmqaut -m qMgr -n queueName -t queue -p user ID
```

9. IBM WebSphere MQ noktadan noktaya iletişim uygulamasının, iletisini SYSTEM.MQTT.TRANSMIT.QUEUE' e koyma yetkisi olup olmadığını denetleyin.

```
dspmqaut -m qMgr -n queueName -t queue -p user ID
```

See "İstemciye doğrudan ileti gönderme" in MQTT istemcilerine ileti göndermek için dağıtımlı kuyruklama yapılandır.

Sorun çözülüyor: Telemetry (MQXR) hizmeti başlamaz

Başlatılamayan telemetri (MQXR) hizmetinin sorununu çözün. WebSphere MQ Telemetry kuruluşunu denetleyin ve herhangi bir dosya eksik, taşınmaz ya da yanlış izinlere sahip değildir. Telemetri (MQXR) hizmeti tarafından kullanılan yolları denetleyerek telemetri (MQXR) hizmet programlarını bulun.

Başlamadan önce

WebSphere MQ Telemetry özelliği kurulu olmalıdır. IBM WebSphere MQ Explorer 'da **IBM WebSphere MQ > Kuyruk Yöneticileri > *qMgrAdı* > Telemetry**' de Telemetry klasörü vardır. Klasör yoksa, kuruluş başarısız olur.

Başlatılacak olan Telemetri (MQXR) hizmeti yaratılmıştır. Telemetri (MQXR) hizmeti yaratılmamışsa, **Define sample configuration ...**(Örnek yapılandırma tanımla) komutunu çalıştırın. `Telemetry` klasöründeki sihirbazı kullanın.

Telemetri (MQXR) hizmeti önceden başlatıldıysa, `Telemetry` klasörü altında ek **Kanallar** ve **Kanal Durumu** klasörleri oluşturulur. Telemetri hizmeti, SYSTEM.MQXR.SERVICE, **Hizmetler** klasöründe yer alıyor. Sistem Nesnelerini göstermek için Gezgin radyo düğmesi tıklatılırsa bu görünür.

Hizmeti başlatmak ve durdurmak, durumunu göstermek ve kullanıcı kimliğinizin hizmeti başlatmak için yetkisinin olup olmadığını görüntülemek için SYSTEM.MQXR.SERVICE seçeneğini sağ tıklatın.

Bu görev hakkında

SYSTEM.MQXR.SERVICE telemetrisi (MQXR) hizmeti başlatılamıyor. Başlatılamaması, iki farklı şekilde ortaya çıkar:

1. Başlatma komutu hemen başarısız olur.
2. Başlatma komutu başarılı olur ve hizmetin durdurularak hemen durdurulmasına neden olur.

Yordam

1. Hizmeti başlat

Sonuç

Hizmet hemen durdurulur. Bir pencerede hata iletisi görüntülenir; örneğin:

```
WebSphere MQ cannot process the request because the
executable specified cannot be started. (AMQ4160)
```

Neden

Kurulumdan dosyalar eksik ya da kurulu dosyalardaki izinler yanlış ayarlanmış.

IBM WebSphere MQ Telemetry özelliği, yalnızca yüksek kullanılabilirlikli kuyruk yöneticisi çiftlerinden birine kurulur. Kuyruk yöneticisi yönetim ortamı bir yedek veritabanına geçiyorsa, SYSTEM.MQXR.SERVICEişlemini başlatmaya çalışır. Telemetri (MQXR) hizmeti yedek veritabanında kurulu olmadığı için hizmeti başlatma komutu başarısız olur.

Araştırma

Hata günlüklerine bakın; bkz. "Sunucu tarafındaki günlükler" sayfa 159.

İşlemler

WebSphere MQ Telemetry özelliğini kurun ya da kaldırın ve yeniden kurun.

2. Hizmeti başlatın; 30 saniye bekleyin; Explorer 'ı yenileyin ve hizmet durumunu denetleyin.

Sonuç

Hizmet başlar ve durur.

Neden

SYSTEM.MQXR.SERVICE komutu **runMQXRService** komutunu başlattı, ancak komut başarısız oldu.

Araştırma

Hata günlüklerine bakın; bkz. "Sunucu tarafındaki günlükler" sayfa 159.

Yalnızca tanımlanan örnek kanalla ilgili bir sorun olup olmadığını görün. Backup and the clear the contents of the *WMQ data directory*\Qmgrs*qMgrName*\mqxr\ directory. Örnek yapılanış sihirbazını çalıştırın ve hizmeti başlatmayı deneyin.

İşlemler

İzin ve yol sorunlarını arayın.

Sorun çözülüyor: JAAS oturum açma modülü telemetri hizmeti tarafından çağrılmadı

Find out if your JAAS login module is not being called by the telemetry (MQXR) service, and configure JAAS to correct the problem.

Başlamadan önce

You have modified *WMQ installation directory*\mqxr\samples\LoginModule.java to create your own authentication class *WMQ installation directory*\mqxr\samples\samples\LoginModule.class. Diğer bir seçenek olarak, kendi JAAS kimlik doğrulama sınıflarınızı yazdınız ve bunları istediğiniz bir dizine yerleştirdiniz. Telemetri (MQXR) hizmetiyle yapılan ilk testten sonra, kimlik doğrulama sınıfınızın telemetri (MQXR) hizmeti tarafından çağrılmadığından kuşkulanıyorsunuz.

Not: Guard against the possibility that your authentication classes might be overwritten by maintenance being applied to WebSphere MQ. Use your own path for authentication classes, rather than a path within the WebSphere MQ directory tree.

Bu görev hakkında

Görev, sorunun çözümünün nasıl çözülecağını göstermek için bir senaryo kullanır. Senaryoda, security.jaas adlı bir paket, JAASLogin.classadlı bir JAAS kimlik doğrulama sınıfı içerir. Bu, C:\WMQTelemetryApps\security\jaasyolunda depolanır. IBM WebSphere MQ Telemetry için JAAS 'nin yapılandırılmasına ilişkin yardım için Telemetri kanalı JAAS yapılandırması ' e bakın. "Örnek JAAS yapılanışı" sayfa 183 örnek bir yapılandırmadır.

Yordam

1. `javax.security.auth.login.LoginException`tarafından yayınlanan bir kural dışı durum için `mqxr.log` içine bakın.

   See “Sunucu tarafındaki günlükler” sayfa 159 for the path to `mqxr.log`, and Şekil 54 sayfa 185 for an example of the exception listed in the log.
2. Correct your JAAS configuration by comparing it with the worked example in “Örnek JAAS yapılanışı” sayfa 183.
3. Replace your login class by the sample `JAASLoginModule`, after refactoring it into your authentication package and deploy it using the same path. `loggedIn` değerini `true` ile `false`arasında değiştirin.

   If the problem goes away when `loggedIn` is `true`, and appears the same when `loggedIn` is `false`, the problem lies in your login class.
4. Sorunun kimlik doğrulaması değil, yetkilendirmeyle birlikte olup olmadığını denetleyin.
   a) Telemetri kanalı tanımlamasını, sabit bir kullanıcı kimliği kullanarak yetki denetimi gerçekleştirmek için değiştirin. `mqm` grubunun üyesi olan bir kullanıcı kimliği seçin.
   b) İstemci uygulamasını yeniden çalıştırın.

      Sorun kaybolursa, çözüm, yetkilendirme için iletilmekte olan kullanıcı kimliği ile yatar. Kullanıcı adı ne iletiliyor? Oturum açma modülünden dosyaya yazdırın. Check its access permissions using IBM WebSphere MQ Explorer, or **dspmqauth**.

Örnek JAAS yapılanışı

Telemetri kanalını yapılandırmak için WebSphere MQ Explorer 'da **Yeni telemetri kanalı** sihirbazını kullanın. Müşteri, 1884 numaralı bağlantı noktasına bağlanır ve `JAASMCAUser` telemetri kanalına bağlanır. Şekil 48 sayfa 183 , telemetri sihirbazı tarafından oluşturulan telemetri özellikleri dosyasına bir örnek gösterir. Bu dosyayı doğrudan düzenlemeyin. The channel authenticates using JAAS, using the configuration called `JAASConfig`. Once the client has authenticated, it uses the user ID `Admin` to authorize its access to IBM WebSphere MQ objects.

```
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
```

Şekil 48. WMQ Installation directory\data\qmgrs\qMgrName\mqxr\mqxr_win.properties

The JAAS configuration file has a stanza named `JAASConfig` that names the Java class `security.jaas.JAASLogin`, which JAAS is to use to authenticate clients.

```
JAASConfig {
  security.jaas.JAASLogin required debug=true;
};
```

Şekil 49. WMQ Installation directory\data\qmgrs\qMgrName\mqxr\jaas.config

SYSTEM.MQTT.SERVICE başlatıldığında, Şekil 50 sayfa 183 içindeki yolu sınıf yoluna (classpath) ekler.

```
CLASSPATH=C:\WMQTelemtryApps;
```

Şekil 50. WMQ Installation directory\data\qmgrs\qMgrName\service.env

Şekil 51 sayfa 184 , Şekil 50 sayfa 183 içindeki ek yolu, telemetri (MQXR) hizmeti için ayarlanan sınıf yoluna (classpath) eklemiş olur.

```
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;
```

Şekil 51. runMQXRService.bat sınıfından sınıf yolu çıkışı

Şekil 52 sayfa 184 içindeki çıkış, telemetri (MQXR) hizmetinin Şekil 48 sayfa 183içinde gösterilen kanal tanımlamasıyla başladığını gösterir.

```
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
```

Şekil 52. WMQ Installation directory\data\qmgrs\qMgrName\errors\mqxr.log

When the client app connects to the JAAS channel, if `com.ibm.mq.MQXR.JAASConfig=JAASWrongConfig` does not match the name of a JAAS stanza in the `jaas.config` file, the connection fails, and the client throws an exception with a return code of `0`; see Şekil 53 sayfa 184. İkinci kural dışı durum (`Client is not connected (32104)`), istemci bağlanmadığında bağlantıyı kesmeyi denediği için yayınlandı.

```
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)
```

Şekil 53. com.ibm.mq.id.PubAsyncRestartablebağlanırken kural dışı durum yayınlandı

`mqxr.log` , Şekil 53 sayfa 184içinde gösterilen ek çıktıyı içerir.

Hata,causeile `javax.security.auth.login.LoginException` , nedeni `No LoginModules configured for JAAS`olan JAAS tarafından algılanır. Hatalı bir yapılandırma adıyla Şekil 54 sayfa 185' ta olduğu gibi bu durum ortaya çıkmış olabilir. JAAS , JAAS yapılanışının yüklenmesine neden olan diğer sorunların sonucu da olabilir.

JAAStarafından bir özel durum bildirilmezse, JAAS , `JAASConfig` stanza adlı `security.jaas.JAASLogin` sınıfını başarıyla yükleyerek kurmuştur.

```
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
```

Şekil 54. `mqxr.log` - JAAS yapılandırması yüklenirken hata oluştu

Sorun çözümleniyor: Yardımcı program başlatılıyor ya da çalıştırılıyor

Operations Console günlüğü için IBM WebSphere MQ Telemetry cinine bakın, izlemeyi açın ya da yardımcı program ile ilgili sorunları gidermek için bu baştaki belirti çizelgesini kullanın.

Yordam

1. Konsol günlüğünü denetleyin.

   Yardımcı program ön planda çalışıyorsa, konsol iletileri uçbirim penceresine yazılır. Yardımcı program artalanda başlatıldıysa, konsol `stdout` ' u yeniden yönlendirdiğiniz yerdir.
2. Yardımcı programı yeniden başlatın.

   Yapılandırma dosyasında yapılan değişiklikler, yardımcı program yeniden başlatılıncaya kadar etkinleştirilmez.
3. Consult Çizelge 7 sayfa 185:

Çizelge 7. Belirti çizelgesi

Sorun	**Önerilen çözüm**

Yardımcı programı Pencerelerüzerinde başlattığınızda aşağıdaki ileti görüntülenir:
`Sistem, belirtilen programı yürütemiyor` ya da
`Uygulamanın başlatılması başarısız oldu`
`Çünkü yan yana yapılandırması yanlış.`
	Microsoft Visual C++ 2008 Redistributable Package ürününü kurun.

İki ya da daha çok yardımcı sunucu ya da MQTT yetenekli sunucular bir köprü ya da köprüler tarafından birbirine bağlanır ve işlemci aşırı yük gösterir.	Bir ya da daha çok iletinin sürekli olarak bir sunucudan diğerine aktarıldığı bir ileti döngüsü vardır. Yapılanış kütüklerindeki konu değiştirgelerini inceleyin. Olanaklı olduğu yerlerde daha belirli konuları kullanın. Bağlantı döngülerinin en yaygın nedeni her iki yönde de geniş genel arama karakteridir.

Çizelge 7. Belirti çizelgesi (devamı var)	
Sorun	**Önerilen çözüm**
Köprü, diğer MQTT istemcilerinin bağlanabileceği uzak bir MQTT yetenekli sunucuya bağlanamıyor.	Uzak sunucu, uzak sunucunun aynı zamanda aygıtlar için WebSphere MQ Telemetri yardımcı programı olup olmadığını saptamak için yapılan girişimlerle uyumsuz olabilir. İleti döngülerini ortadan kaldırmak için özel işlemeyi geçersiz kılmak için **try_private** ayarını `off` (kapalı) olarak ayarlamayı deneyin.
Bir köprü yapılandırıldığında bu ileti yazdırılır: `Uyarı: Connect 1888 yuvasında ilk paket değil, CONNACK değeri alındı.`	Yerel yardımcı programa geri dönebilmek için büyük olasılıkla bir köprü yapılandırmış olmasınız. Geriçevrim desteklenmiyor.

Sorun çözülüyor: MQTT istemcileri cine bağlanmıyor

İstemciler yardımcı programa bağlanmıyor ya da yardımcı program diğer yardımcı programlara ya da bir WebSphere MQ telemetry kanalına bağlanmıyor.

Bu görev hakkında

Yardımcı program tarafından gönderilen ve alınan MQTT paketlerinin izlenmesi.

Yordam

Set the **trace_output** parameter to `protocol` in the daemon configuration file or send a command to the daemon using the `amqtdd.upd` file.

`amqtdd.upd` dosyasını kullanmaya ilişkin bir örnek için bkz. Aygıtlar için IBM WebSphere MQ Telemetry cini ile IBM WebSphere MQarasındaki aktarma iletileri .

Yardımcı program, iletişim kuralı ayarını kullanarak, gönderdiği ve aldığı her MQTT paketini açıklayan konsola bir ileti yazdırır.

Özel notlar

Bu belge, ABD'de kullanıma sunulan ürünler ve hizmetler için hazırlanmıştır.

IBM, bu belgede sözü edilen ürün, hizmet ya da özellikleri diğer ülkelerde kullanıma sunmayabilir. Bulunduğunuz yerde kullanıma sunulan ürün ve hizmetleri yerel IBM müşteri temsilcisinden ya da çözüm ortağından öğrenebilirsiniz. Bir IBM ürün, program ya da hizmetine gönderme yapılması, açık ya da örtük olarak, yalnızca o IBM ürünü, programı ya da hizmetinin kullanılabileceğini göstermez. Aynı işlevi gören ve IBM'in fikri mülkiyet haklarına zarar vermeyen herhangi bir ürün, program ya da hizmet de kullanılabilir. Ancak, IBM dışı ürün, program ya da hizmetlerle gerçekleştirilen işlemlerin değerlendirilmesi ve doğrulanması kullanıcının sorumluluğundadır.

IBM'in, bu belgedeki konularla ilgili patentleri ya da patent başvuruları olabilir. Bu belgenin size verilmiş olması, patentlerin izinsiz kullanım hakkının da verildiği anlamına gelmez. Lisansla ilgili sorularınızı aşağıdaki adrese yazabilirsiniz:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Çift byte (DBCS) bilgilerle ilgili lisans soruları için, ülkenizdeki IBM'in Fikri Haklar (Intellectual Property) bölümüyle bağlantı kurun ya da sorularınızı aşağıda adrese yazın:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japonya

Aşağıdaki paragraf, İngiltere ya da bu tür hükümlerin yerel yasalarla uyuşmadığı diğer ülkelerde geçerli değildir: INTERNATIONAL BUSINESS MACHINES CORPORATION BU YAYINI, HAK İHLALİ YAPILMAYACAĞINA DAİR GARANTİLERLE TİCARİLİK VEYA BELİRLİ BİR AMACA UYGUNLUK İÇİN ZIMNİ GARANTİLER DE DAHİL OLMAK VE FAKS BUNLARLA SINIRLI OLMAMAK ÜZERE AÇIK YA DA ZIMNİ HİÇBİR GARANTİ VERMEKSİZİN "OLDUĞU GİBİ" ESASIYLA SAĞLAMAKTADIR. Bazı ülkeler bazı işlemlerde garantinin açık ya da örtük olarak reddedilmesine izin vermez; dolayısıyla, bu bildirim sizin için geçerli olmayabilir.

Bu yayın teknik yanlışlar ya da yazım hataları içerebilir. Buradaki bilgiler üzerinde düzenli olarak değişiklik yapılmaktadır; söz konusu değişiklikler sonraki basımlara yansıtılacaktır. IBM, önceden bildirimde bulunmaksızın, bu yayında açıklanan ürünler ve/ya da programlar üzerinde iyileştirmeler ve/ya da değişiklikler yapabilir.

Bu belgede IBM dışı Web sitelerine yapılan göndermeler kullanıcıya kolaylık sağlamak içindir ve bu Web sitelerinin onaylanması anlamına gelmez. Bu Web sitelerinin içerdiği malzeme, bu IBM ürününe ilişkin malzemenin bir parçası değildir ve bu tür Web sitelerinin kullanılmasının sorumluluğu size aittir.

IBM'e bilgi ilettiğinizde, IBM bu bilgileri size karşı hiçbir yükümlülük almaksızın uygun gördüğü yöntemlerle kullanabilir ya da dağıtabilir.

(i) Bağımsız olarak yaratılan programlarla, bu program da içinde olmak üzere diğer programlar arasında bilgi değiş tokuşuna ve (ii) değiş tokuş edilen bilginin karşılıklı kullanımına olanak sağlamak amacıyla bu program hakkında bilgi sahibi olmak isteyen lisans sahipleri şu adrese yazabilirler:

IBM Corporation
Yazılım Birlikte Çalışabilirlik Koordinatörü, Bölüm 49XA
3605 Highway 52 N

Rochester, MN 55901
U.S.A.

Bu tür bilgiler, ilgili kayıt ve koşullar altında ve bazı durumlarda bedelli olarak edinilebilir.

Bu belgede açıklanan lisanslı program ve bu programla birlikte kullanılabilecek tüm lisanslı malzeme, IBM tarafından, IBM Müşteri Sözleşmesi, IBM Uluslararası Program Lisansı Sözleşmesi ya da eşdeğer herhangi bir sözleşmenin kayıt ve koşulları altında sağlanır.

Burada belirtilen performans verileri denetimli bir ortamda elde edilmiştir. Bu nedenle, başka işletim ortamlarında çok farklı sonuçlar alınabilir. Bazı ölçümler geliştirilme düzeyindeki sistemlerde yapılmıştır ve bu ölçümlerin genel kullanıma sunulan sistemlerde de aynı olacağı garanti edilemez. Ayrıca, bazı sonuçlar öngörü yöntemiyle elde edilmiş olabilir. Dolayısıyla, gerçek sonuçlar farklı olabilir. Bu belgenin kullanıcıları, kendi ortamları için geçerli verileri kendileri doğrulamalıdırlar.

IBM dışı ürünlerle ilgili bilgiler, bu ürünleri sağlayan firmalardan, bu firmaların yayın ve belgelerinden ve genel kullanıma açık diğer kaynaklardan alınmıştır. IBM bu ürünleri sınamamıştır ve IBM dışı ürünlerle ilgili performans doğruluğu, uyumluluk gibi iddiaları doğrulayamaz. IBM dışı ürünlerin yeteneklerine ilişkin sorular, bu ürünleri sağlayan firmalara yöneltilmelidir.

IBM'in gelecekteki yönelim ve kararlarına ilişkin tüm bildirimler değişebilir ve herhangi bir duyuruda bulunulmadan bunlardan vazgeçilebilir; bu yönelim ve kararlar yalnızca amaç ve hedefleri gösterir.

Bu belge, günlük iş ortamında kullanılan veri ve raporlara ilişkin örnekler içerir. Örneklerin olabildiğince açıklayıcı olması amacıyla kişi, şirket, marka ve ürün adları belirtilmiş olabilir. Bu adların tümü gerçek dışıdır ve gerçek iş ortamında kullanılan ad ve adreslerle olabilecek herhangi bir benzerlik tümüyle rastlantıdır.

YAYIN HAKKI LİSANSI:

Bu belge, çeşitli işletim platformlarında programlama tekniklerini gösteren, kaynak dilde yazılmış örnek uygulama programları içerir. Bu örnek programları, IBM'e herhangi bir ödemede bulunmadan, örnek programların yazıldığı işletim altyapısına ilişkin uygulama programlama arabirimiyle uyumlu uygulama programlarının geliştirilmesi, kullanılması, pazarlanması ya da dağıtılması amacıyla herhangi bir biçimde kopyalayabilir, değiştirebilir ve dağıtabilirsiniz. Bu örnekler her koşul altında tüm ayrıntılarıyla sınanmamıştır. Dolayısıyla, IBM bu programların güvenilirliği, bakım yapılabilirliği ya da işlevleri konusunda açık ya da örtük güvence veremez.

Bu bilgileri elektronik kopya olarak görüntülediyseniz, fotoğraflar ve renkli resimler görünmeyebilir.

Programlama arabirimi bilgileri

Programlama arabirimi bilgileri (sağlandıysa), bu programla birlikte kullanılmak üzere uygulama yazılımları yaratmanıza yardımcı olmak üzere hazırlanmıştır.

Bu kitap, müşterinin IBM WebSphere MQhizmetlerini edinmek üzere program yazmasına olanak tanıyan, amaçlanan programlama arabirimlerine ilişkin bilgiler içerir.

Ancak, bu bilgiler tanılama, değiştirme ve ayarlama bilgilerini de içerebilir. Tanılama, değiştirme ve ayarlama bilgileri, uygulama yazılımlarınızda hata ayıklamanıza yardımcı olur.

Önemli: Bu tanılama, değiştirme ve ayarlama bilgilerini bir programlama arabirimi olarak kullanmayın; bu, değişiklik söz konusu olduğunda kullanılır.

Ticari Markalar

IBM, IBM logosu, ibm.com, IBM Corporation 'ın dünya çapında birçok farklı hukuk düzeninde kayıtlı bulunan ticari markalarıdır. IBM ticari markalarının güncel bir listesini Web üzerinde "Telif hakkı ve ticari marka bilgileri"www.ibm.com/legal/copytrade.shtmladresinde bulabilirsiniz. Diğer ürün ve hizmet adları IBM'in veya diğer şirketlerin ticari markaları olabilir.

Microsoft ve Windows, Microsoft Corporation'ın ABD ve/veya diğer ülkelerdeki ticari markalarıdır.

UNIX, The Open Group şirketinin ABD ve diğer ülkelerdeki tescilli ticari markasıdır.

Linux, Linus Torvalds'ın ABD ve/ya da diğer ülkelerdeki tescilli ticari markasıdır.

Bu ürün, Eclipse Project (http://www.eclipse.org/) tarafından geliştirilen yazılımları içerir.

Java ve Java tabanlı tüm markalar ve logolar, Oracle firmasının ve/ya da iştiraklerinin markaları ya da tescilli markalarıdır.

Parça numarası:

(1P) P/N: