2018

IBM INFOSPHERE DATA REPLICATION

APACHE KAFKA AND SSL

TOMASZ ZIEJA

Contents

Introduction

There are many ways Apache Kafka can be configured to make use of SSL. In first section of the document I present one such configuration – mutual SSL authentication using self-signed CA certificate. I hope it will be helpful to quickly create test environments.

Further sections discuss ways IIDR CDC needs to be set up to accommodate different Kafka SSL configurations. This is not a comprehensive guide, but I think it covers most scenarios and can be a starting point for those not covered.

Last section discusses several common problems you might encounter, their causes and resolutions. It also presents useful debugging tools and strategies.

Some notes about SSL and Apache Kafka

SSL authentication differs from Kerberos because it does not require Key Distribution Center. Authentication is performed using certificates and decision whether a certificate is to be trusted depends on the content of a truststore.

Kafka supports both server-only authentication and mutual authentication. In the former case, only certificate presented by a server is validated. In the latter configuration, both server and client certificates are validated before conversation can occur.

Because Zookeeper does not currently support SSL you do not want to connect to it. Hence “bootstrap.servers” option is always set in presented examples.

Quick note on terminology: Secure Sockets Layer (SSL) is the predecessor of Transport Layer Security (TLS), and SSL has been deprecated since June 2015. However, for historical reasons, Kafka (like Java) uses the term SSL instead of TLS in configuration and code.

How to configure Apache Kafka to use SSL encryption and authentication

This sample configuration lets you quickly set up SSL for one node Kafka cluster. After completing the steps, you will have Kafka broker and Schema Registry set up for mutual SSL authentication with self-signed CA certificate.

Do not attempt to use this configuration in production. The steps require root access.

Following software was used when creating this example:

- Confluent Kafka 2.11
- OpenJDK Java 1.8.0
- openssl 1.0.2

Generating certificates

Before you can configure Kafka to use SSL you need to generate certificates. Three certificates are required for mutual authentication setup:

- Certificate Authority (CA) certificate
- Kafka Server certificate
- Kafka Client certificate

This task can be accomplished using “openssl” package and “keytool” command.

```
# Generate self-signed CA certificate. When prompted provide password and Distinguished Name.
openssl req -new -x509 -keyout ca-key -out ca-cert -days 1000

# Generate Kafka Server certificate and put it in new keystore.
keytool -keystore kafka.server.keystore.jks -alias localhost -validity 1000 -genkeypair -storepass
password -keypass password -dname "CN=Kafka Server"

# Generate IIDR CDC certificate and put it in new keystore.
keytool -keystore kafka.client.keystore.jks -alias localhost -validity 1000 -genkeypair -storepass
password -keypass password -dname "CN=IIDR CDC for Kafka Client"

# Create truststore files by importing CA certificate. Answer yes when prompted.
keytool -keystore kafka.client.truststore.jks -alias CARoot -importcert -file ca-cert -storepass password
keytool -keystore kafka.server.truststore.jks -alias CARoot -importcert -file ca-cert -storepass password
```

Next you need to sign the certificates using CA certificate. Before you do that, open “/etc/pki/tls/openssl.cnf” configuration file and uncomment below line. This enables extension copying.

```
copy_extensions = copy
```

Execute below commands only if you never signed any certificates on the machine:

```
touch /etc/pki/CA/index.txt && echo '1000' > /etc/pki/CA/serial
```

Sign the certificates. Make sure you replace **serverHostname** placeholder with appropriate value.

```
# Export certificate requests.
keytool -keystore kafka.server.keystore.jks -alias localhost -certreq -file cert-server-file -storepass
password -ext SAN=DNS:serverHostname
```

```
keytool -keystore kafka.client.keystore.jks -alias localhost -certreq -file cert-client-file -storepass password

# Sign certificates. Answer y when prompted.
openssl ca -cert ca-cert -keyfile ca-key -in cert-server-file -out cert-server-signed -policy policy_anything
openssl ca -cert ca-cert -keyfile ca-key -in cert-client-file -out cert-client-signed -policy policy_anything

# Import signed certificates and CA certificate back into keystores. Answer yes when prompted.

keytool -keystore kafka.server.keystore.jks -alias CARoot -importcert -file ca-cert -storepass password
keytool -keystore kafka.client.keystore.jks -alias CARoot -importcert -file ca-cert -storepass password

keytool -keystore kafka.server.keystore.jks -alias localhost -importcert -file cert-server-signed -storepass password

keytool -keystore kafka.client.keystore.jks -alias localhost -importcert -file cert-client-signed -storepass password
```

You should now have following files:

```
# Used by brokers and Schema Registry. Need to be accessible by Kafka user.
kafka.server.keystore.jks
kafka.server.truststore.jks

# Used by IIDR CDC. Need to be accessible by IIDR CDC user.
kafka.client.keystore.jks
kafka.client.truststore.jks
```

Configure Kafka brokers

Add / adjust following options in “kafka_installdir/etc/server.properties” file:

```
ssl.truststore.location=/path/to/kafka.server.truststore.jks
ssl.truststore.password=password
ssl.keystore.location=/path/to/kafka.server.keystore.jks
ssl.keystore.password=password
ssl.key.password=password
security.inter.broker.protocol=SSL
ssl.client.auth=required
listeners=PLAINTEXT://:9092,SSL://:9093
```

Configure Schema Registry

Add / adjust following options in “kafka_installdir/etc/schema-registry/schema-registry.properties” file:

```
listeners=http://0.0.0.0:8081,https://0.0.0.0:8082
ssl.truststore.location=/path/to/kafka.server.truststore.jks
ssl.truststore.password=password
ssl.keystore.location=/path/to/kafka.server.keystore.jks
ssl.keystore.password=password
ssl.key.password=password
ssl.client.auth=true
```

How to configure IIDR CDC to use SSL encryption and server authentication

This sample configuration involves authenticating Kafka brokers and Schema Registry using SSL certificates. IIDR CDC certificate is not validated, so it does not have to be signed by trusted CA.

Configure Consumer and Producer

Add below options to the “kafkaproducer.properties” and “kafkaconsumer.properties” files:

```
security.protocol=SSL
bootstrap.servers=brokerHostname:brokerPort
ssl.truststore.location=/path/to/kafka.client.truststore.jks
ssl.truststore.password=password
```

Configure Schema Registry connection (up to version 11.4.0.1-5099)

Add below options to “IIDR_installdir/instance/instancename/conf/dmts64.vmargs” file. Create the file if it does not exist.

```
-Djavax.net.ssl.trustStore=/path/to/kafka.client.truststore.jks -
Djavax.net.ssl.trustStorePassword=password
```

Configure Schema Registry connection (from version 11.4.0.1-5101)

Run “dmconfigurets” command, select “Manage encryption profiles” option, edit encryption profile used by your instance and specify the path, type and password for custom truststore file.

Configure Subscription

In Management Console, right click on the subscription, select “Kafka Properties” and tick “Encrypted” box in Schema Registry section.

How to configure IIDR CDC to use SSL encryption and SASL authentication

This sample configuration involves authenticating Kafka brokers and Schema Registry using SSL certificates. IIDR CDC authenticates to the brokers using SASL and to Schema Registry using SSL certificate. SASL PLAIN mechanism is used, which is simple user and password checking.

Configure Consumer and Producer

Add below options to the “kafkaproducer.properties” and “kafkaconsumer.properties” files:

```
security.protocol=SASL_SSL
bootstrap.servers=brokerHostname:brokerPort
ssl.truststore.location=/path/to/kafka.client.truststore.jks
ssl.truststore.password=password
sasl.mechanism=PLAIN
```

Configure Schema Registry connection and JAAS config file (up to version 11.4.0.1-5099)

Add below options to “IIDR_installdir/instance/instancename/conf/dmts64.vmargs” file. Create the file if it does not exist.

```
-Djava.security.auth.login.config=/path/to/jaas.conf -
Djavax.net.ssl.trustStore=/path/to/kafka.client.truststore.jks -
```

```
Djavax.net.ssl.trustStorePassword=password -Djavax.net.ssl.keyStore=/path/to/kafka.client.keystore.jks
-Djavax.net.ssl.keyStorePassword=password
```

Configure Schema Registry connection and JAAS config file (from version 11.4.0.1-5101)

Add below options to "IIDR_installdir/instance/instancename/conf/dmts64.vmargs" file. Create the file if it does not exist.

```
-Djava.security.auth.login.config=/path/to/jaas.conf
```

Run "dmconfigurets" command, select "Manage encryption profiles" option, edit encryption profile used by your instance and specify the path, type and password for custom truststore and keystore files.

Configure Subscription

In Management Console, right click on the subscription, select "Kafka Properties" and tick "Encrypted" box in Schema Registry section.

Create JAAS config file

Create "jaas.conf" file and populate it:

```
KafkaClient {
 org.apache.kafka.common.security.plain.PlainLoginModule required
 username="user"
 password="password";
};
```

Kerberos

For information how to configure SASL authentication with Kerberos, refer to "Configuring CDC with a Kerberized Kafka" section of following document:
https://www.ibm.com/developerworks/community/files/app#/file/6e39fdce-b6d0-4865-a4f0-345386a9783d

How to configure IIDR CDC to use SSL encryption and mutual authentication

This type of configuration involves authenticating Kafka brokers and Schema Registry using SSL certificates. IIDR CDC certificate is also validated, so it needs to be signed by trusted CA.

Configure Consumer and Producer

Add below options to the "kafkaproducer.properties" and "kafkaconsumer.properties" files

```
security.protocol=SSL
bootstrap.servers=brokerHostname:brokerPort
ssl.truststore.location=/path/to/kafka.client.truststore.jks
ssl.truststore.password=password
ssl.keystore.location=/path/to/kafka.client.keystore.jks
ssl.keystore.password=password
ssl.key.password=password
```

Configure Schema Registry connection (up to version 11.4.0.1-5099)

Add below options to "IIDR_installdir/instance/instancename/conf/dmts64.vmargs" file. Create the file if it does not exist.

```
-Djavax.net.ssl.trustStore=/path/to/kafka.client.truststore.jks -
Djavax.net.ssl.trustStorePassword=password -Djavax.net.ssl.keyStore=/path/to/kafka.client.keystore.jks
-Djavax.net.ssl.keyStorePassword=password
```

Configure Schema Registry connection (from version 11.4.0.1-5101)

Run “dmconfigurets” command, select “Manage encryption profiles” option, edit encryption profile used by your instance and specify the path, type and password for custom truststore and keystore files.

Configure Subscription

In Management Console, right click on the subscription, select “Kafka Properties” and tick “Encrypted” box in Schema Registry section.

Typical problems with SSL

Hostname does not match Subject Alternative Name on a certificate

```
198     2018-11-26 12:25:16.980     AA Kafka Target Formatter:0{107}
        com.datamirror.ts.util.TsExceptionHandler   processUnhandledException() An uncaught
exception has occurred: †org.apache.kafka.common.errors.SerializationException Error serializing Avro
message|Caused by: javax.net.ssl.SSLHandshakeException java.security.cert.CertificateException: No
name matching localhost found|   at com.ibm.jsse2.j.a(j.java:12)|   at com.ibm.jsse2.as.a(as.java:11 8)|
at com.ibm.jsse2.C.a(C.java:193)|   at com.ibm.jsse2.C.a(C.java:245)|   at
com.ibm.jsse2.D.a(D.java:242)|   at com.ibm.jsse2.D.a(D.java:56)|   at com.ibm.jsse2.C.r(C.java:69)|
at com.ibm.jsse2.C.a(C.java:580)|   at com.ibm.jsse2.as.a(as.java:512)|   at
com.ibm.jsse2.as.i(as.java:969)|   at com.ibm.jsse2.as.a(as.java:680)|   at
com.ibm.jsse2.as.startHandshake(as.java:859)|   at
com.ibm.net.ssl.www2.protocol.https.c.afterConnect(c.java:16)|   at
com.ibm.net.ssl.www2.protocol.https.d.connect(d.java:44)|   at
sun.net.www.protocol.http.HttpURLConnection.getOutputStream0(HttpURLConnection.java:1297)|   at
sun.net.www.protocol.http.HttpURLConnection.getOutputStream(HttpURLConnection.java:1272)|   at
com.ibm.net.ssl.www2.protocol.https.b.getOutputStream(b.java:99)|   at
io.confluent.kafka.schemaregistry.client.rest.RestService.sendHttpRequest(RestService.java:141)|   at
io.confluent.kafka.schemaregistry.client.rest.RestService.httpRequest(RestService.java:181)|   at
io.confluent.kafka.schemaregistry.client.rest.RestService.registerSchema(RestService.java:232)
```

Either Kafka broker or Schema Registry presented certificate with Subject Alternative Name that does not match the hostname of the server.

Use hostname that matches certificate’s Subject Alternative Name when connecting to Kafka broker (“bootstrap.servers” option) or to Schema Registry (Subscription’s Kafka Properties).

If you are not sure what Subject Alternative Names are on the certificate, refer to section “Debugging SSL”.

Schema Registry certificate not trusted

```
117     2018-11-26 14:17:44.095     AA Kafka Target Formatter:0{80}
        com.datamirror.ts.util.TsExceptionHandler   processUnhandledException() An uncaught
exception has occurred: †org.apache.kafka.common.errors.SerializationException Error serializing Avro
message|Caused by: javax.net.ssl.SSLHandshakeException com.ibm.jsse2.util.h: PKIX path building
failed: java.security.cert.CertPathBuilderException: PKIXCertPathBuilderImpl could not build a valid
CertPath.; internal cause is: |†java.security.cert.CertPathValidatorException: The certificate issued by
```

```
CN=IBM-CA, OU=SWG, O=IBM, L=Krakow, ST=Poland, C=PL is not trusted; internal cause is:
|†java.security.cert.CertPathValidatorException: Certificate chaining error|    at
com.ibm.jsse2.j.a(j.java:12)|    at com.ibm.jsse2.as.a(as.java:118)|    at com.ibm.jsse2.C.a(C.java:193)|
at com.ibm.jsse2.C.a(C.java:245)|    at com.ibm.jsse2.D.a(D.java:242)|    at
com.ibm.jsse2.D.a(D.java:56)|    at com.ibm.jsse2.C.r(C.java:69)|    at com.ibm.jsse2.C.a(C.java:580)|
at com.ibm.jsse2.as.a(as.java:512)|    at com.ibm.jsse2.as.i(as.java:969)|    at
com.ibm.jsse2.as.a(as.java:680)|    at com.ibm.jsse2.as.startHandshake(as.java:859)|    at
com.ibm.net.ssl.www2.protocol.https.c.afterConnect(c.java:16)|    at
com.ibm.net.ssl.www2.protocol.https.d.connect(d.java:44)|    at
sun.net.www.protocol.http.HttpURLConnection.getOutputStream0(HttpURLConnection.java:1297)|    at
sun.net.www.protocol.http.HttpURLConnection.getOutputStream(HttpURLConnection.java:1272)|    at
com.ibm.net.ssl.www2.protocol.https.b.getOutputStream(b.java:99)|    at
io.confluent.kafka.schemaregistry.client.rest.RestService.sendHttpRequest
```

Valid certificate path could not be built. Certificate presented by Schema Registry is not signed by known Certificate Authority.

Make sure correct CA certificates are added to the truststore. For information on how to display the content of the truststore, refer to “Debugging SSL” section.

IIDR CDC certificate not trusted

```
47      2018-11-26 16:47:19.114      AA Target Data Channel{61}
        com.datamirror.ts.util.TsExceptionHandler    processUnhandledException() An uncaught
exception has occurred: †org.apache.kafka.common.errors.TimeoutException Timeout expired while
fetching topic metadata
```

This error indicates a problem setting up SSL connection between IIDR CDC agent and Kafka broker. Possible causes:

- IIDR CDC certificate is not considered trusted by Kafka broker.
- Kafka broker certificate is not considered trusted by IIDR CDC instance.
- SASL authentication failure.

Examine the content of the keystore. Check if all required CA certificates are present in the truststore. Test your SSL configuration outside of IIDR CDC. For more information on how to perform those actions, refer to “Debugging SSL” section.

Unable to access keystore / truststore

```
java.io.IOException Keystore was tampered with, or password was incorrect|    at
com.ibm.crypto.provider.JavaKeyStore.engineLoad(Unknown Source)|    at
java.security.KeyStore.load(KeyStore.java:1456)|    at
org.apache.kafka.common.security.ssl.SslFactory$SecurityStore.load(SslFactory.java:206)|    at
org.apache.kafka.common.security.ssl.SslFactory$SecurityStore.access$000(SslFactory.java:190)|    at
org.apache.kafka.common.security.ssl.SslFactory.createSSLContext(SslFactory.java:126)|    at
org.apache.kafka.common.security.ssl.SslFactory.configure(SslFactory.java:108)|    at
org.apache.kafka.common.network.SslChannelBuilder.configure(SslChannelBuilder.java:41)|    at
org.apache.kafka.common.network.ChannelBuilders.create(ChannelBuilders.java:70)|    at
org.apache.kafka.clients.ClientUtils.createChannelBuilder(ClientUtils.java:83)|    at
```

```
org.apache.kafka.clients.consumer.KafkaConsumer.<init>(KafkaConsumer.java:623)|   at
org.apache.kafka.clients.consumer.KafkaConsumer.<init>(KafkaConsumer.java:587)
```

The exception indicates problems accessing keystore or truststore. Possible reasons:

- Corrupted file.
- Incorrect password.
- Incorrect truststore type (PKCS12 instead of JKS).

Debugging SSL

Some useful commands and strategies to debug SSL problems.

Connect to Kafka broker and display the certificate with its Subject Alternative Names:

```
openssl s_client -connect brokerHostname:brokerPort | openssl x509 -text
```

Connect to Schema Registry and display the certificate with its Subject Alternative Names:

```
openssl s_client -connect registryHostname:registryPort | openssl x509 -text
```

Display the content of JKS truststore:

```
keytool -list -v -keystore kafka.client.truststore.jks -storetype JKS
```

Display the content of JKS keystore:

```
keytool -list -v -keystore kafka.client.keystore.jks -storetype JKS
```

If keytool invocation returns "Invalid keystore format" error, try using keytool shipped with IIDR CDC. The tool can be found in "IIDR_installdir/jre64/jre/bin" directory.

Add below parameter at the beginning of "dmts64.vmargs" file to enable SSL debugging:

```
-Djavax.net.debug=all
```

Debug information are written on standard output or to "nohup.out" file.

How to verify your SSL configuration outside of IIDR CDC

It is often useful to verify your SSL client configuration outside of IIDR CDC. You can do it with console producer and consumer programs that are shipped with Kafka.

If you use mutual authentication for both Kafka brokers and Schema Registry, then run below command to produce to a Kafka topic:

```
export JAVA_HOME=/cdc/install/dir/jre64/jre

export SCHEMA_REGISTRY_OPTS="-Djavax.net.ssl.trustStore=/path/to/kafka.client.truststore.jks -
Djavax.net.ssl.trustStorePassword=password -Djavax.net.ssl.keyStore=/path/to/kafka.client.keystore.jks
-Djavax.net.ssl.keyStorePassword=password"

echo 16830912 | /kafka/install/dir/bin/kafka-avro-console-producer --broker-list
```

```
brokerHostname:brokerPort --topic topicName --producer.config /path/to/kafkaproducer.properties --
property schema.registry.url=https://schemaHost:schemaPort --property value.schema='{"type":"int"}'
```

Next consume the data that you just produced:

```
/kafka/install/dir/bin/kafka-avro-console-consumer --bootstrap-server brokerHostname:brokerPort --
topic topicName --consumer.config /path/to/kafkaconsumer.properties --property
schema.registry.url=https://schemaHost:schemaPort --from-beginning
```

In case your Kafka environment does not have a Schema Registry, issue those commands instead:

```
export JAVA_HOME=/cdc/install/dir/jre64/jre

export KAFKA_OPTS=""

echo 16830912 | /kafka/install/dir/bin/kafka-console-producer --broker-list brokerHostname:brokerPort
--topic topicName --producer.config /path/to/kafkaproducer.properties

/kafka/install/dir/bin/kafka-console-consumer --bootstrap-server brokerHostname:brokerPort --topic
topicName --consumer.config /path/to/kafkaconsumer.properties --from-beginning
```

You need to adjust examples accordingly if you use the other type of configuration.

Useful resources

Confluent Kafka online documentation:
https://docs.confluent.io/current/kafka/authentication_ssl.html
https://docs.confluent.io/current/kafka/authentication_sasl/index.html

Keytool manual:
https://docs.oracle.com/javase/8/docs/technotes/tools/unix/keytool.html

Openssl manual:
https://www.openssl.org/docs/man1.0.2/apps/openssl.html

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing IBM Corporation North Castle Drive Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing Legal and Intellectual Property Law IBM Japan Ltd. 1623-14, Shimotsuruma, Yamato-shi Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

Trademarks

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation J46A/G4 555 Bailey Avenue San Jose, CA 95141-1003 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

The following terms are trademarks or registered trademarks of other companies:

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Java™ and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

The United States Postal Service owns the following trademarks: CASS, CASS Certified, DPV, LACSLink, ZIP, ZIP + 4, ZIP Code, Post Office, Postal Service, USPS and United States Postal Service. IBM Corporation is a non-exclusive DPV and LACSLink licensee of the United States Postal Service.

Other company, product, or service names may be trademarks or service marks of others.