

## **Please take Note!**

Before using this SupportPac, please be sure to read the paragraphs on “disclaimers”, “warranty and liability exclusion”, “errors and omissions”, and the other general information paragraphs in the "Notices" section below.

### **First Edition, November 2006.**

This edition applies to WebSphere MQ V5.3 for AIX, HP-UX, Linux, Solaris and Windows (and to all subsequent releases and modifications until otherwise indicated in new editions). Some sections are applicable to MQIPT version 1.3.3 (and to all subsequent releases and modifications until otherwise indicated in new editions).

© Copyright International Business Machines Corporation 2006. All rights reserved.

#### Note to U.S. Government Users

Documentation related to restricted rights.

Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule contract with IBM Corp.

## **Notices**

### **DISCLAIMERS**

You should not assume that the information or samples supplied with this SupportPac have been submitted to any formal testing by IBM. Samples are provided for demonstration purposes only.

Any use of this information and use of any of the techniques are the responsibility of the licensed user. Much depends on the ability of the licensed user to evaluate the information and to deploy the techniques in their own operational environment.

### **WARRANTY AND LIABILITY EXCLUSION**

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 SUPPORTPAC “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.

In Germany and Austria, notwithstanding the above exclusions, IBM's warranty and liability are governed only by the respective terms applicable for Germany and Austria in the corresponding IBM program license agreement(s).

### **ERRORS AND OMISSIONS**

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

### **INTENDED AUDIENCE**

This report is intended for architects, systems programmers, analysts and programmers who want to understand the use of LDAP CRLs with either MQIPT or WebSphere MQ on AIX, HP-UX, Linux, Solaris and Windows platforms. It is assumed that the reader is familiar with the concepts and operation of WebSphere MQ. For the sections on MQIPT, it is assumed that the reader is familiar with MQIPT, although this SupportPac is also intended for readers who do not use MQIPT. The information is not intended as the specification of any programming interface that is provided by WebSphere MQ or MQIPT.

#### **LOCAL AVAILABILITY**

References in this report to IBM products or programs do not imply that IBM intends to make these available in all countries in which IBM operates. Consult your local IBM representative for information on the products and services currently available in your area.

#### **ALTERNATIVE PRODUCTS AND SERVICES**

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.

#### **USE OF INFORMATION PROVIDED BY YOU**

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

#### **TRADEMARKS AND SERVICE MARKS**

The following terms used in this publication are trademarks of International Business Machines Corporation in the United States, other countries or both:

- IBM
- AIX
- Tivoli
- WebSphere

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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.

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

Other product and service names might be trademarks of IBM or other companies.

#### **EXPORT REGULATIONS**

You agree to comply with all applicable export and import laws and regulations.

## Contents

<a href="#">Using LDAP and SSL with WebSphere MQ and MQIPT.....</a>	<a href="#">4</a>
<a href="#">Introduction.....</a>	<a href="#">4</a>
<a href="#">Style Conventions.....</a>	<a href="#">6</a>
<a href="#">SSL and LDAP overview.....</a>	<a href="#">8</a>
<a href="#">Basic Concepts.....</a>	<a href="#">8</a>
<a href="#">How CRLs work with SSL.....</a>	<a href="#">10</a>
<a href="#">Using SSL Certificates.....</a>	<a href="#">12</a>
<a href="#">Configuring the LDAP server for WebSphere MQ.....</a>	<a href="#">13</a>
<a href="#">General Considerations.....</a>	<a href="#">13</a>
<a href="#">LDAP Server Software Setup.....</a>	<a href="#">13</a>
<a href="#">Configuring the LDAP Directory.....</a>	<a href="#">14</a>
<a href="#">Considerations for MQIPT 1.3.3.....</a>	<a href="#">16</a>
<a href="#">Considerations for WebSphere MQ V5.3.....</a>	<a href="#">17</a>
<a href="#">Considerations for WebSphere MQ V6.....</a>	<a href="#">17</a>
<a href="#">Configuring LDAP for WebSphere MQ servers.....</a>	<a href="#">19</a>
<a href="#">Configuring LDAP CRLs for WebSphere MQ clients.....</a>	<a href="#">20</a>
<a href="#">C and C++ Clients.....</a>	<a href="#">20</a>
<a href="#">Java Clients.....</a>	<a href="#">22</a>
<a href="#">Other Programming Languages.....</a>	<a href="#">22</a>
<a href="#">Sample Scenarios.....</a>	<a href="#">23</a>
<a href="#">Scenario 1: Queue Managers.....</a>	<a href="#">23</a>
<a href="#">Scenario 2: Queue Managers.....</a>	<a href="#">24</a>
<a href="#">Scenario 3: Client Application.....</a>	<a href="#">25</a>
<a href="#">Scenario 4: Client Application and MQIPT.....</a>	<a href="#">27</a>
<a href="#">Scenario 5: Queue Managers and MQIPT.....</a>	<a href="#">29</a>

## Using LDAP and SSL with WebSphere MQ and MQIPT

### Introduction

The Lightweight Directory Access Protocol (LDAP) has many potential benefits to WebSphere MQ users. Conceptually, an LDAP server is a directory which provides a way of looking up structured information. The data in the directory could include user identity data, cryptographic keys or certificates, queue names, channel definitions or indeed anything else.

There are other directory services which can also be used, such as the Microsoft® Active Directory™. This document will focus on LDAP directories and their use with WebSphere MQ; from here on the term “directory” will be used to mean an LDAP directory unless otherwise stated.

Broadly, the uses of LDAP can be divided into two categories:

1. Authentication and authorization. Authentication is checking that a person is who they claim to be. Authorization is checking that an identified individual is permitted to perform some operation.
2. Separating configuration data from the application.

In general, the advantages of using an LDAP directory with WebSphere MQ are:

1. A central LDAP directory can be used to provide consistent enforcement of security policies across an organization. One example is the use of Certificate Revocation Lists to validate SSL certificates and prevent compromised keys from being used for authentication.
2. A central directory can also enable consistent authority checking across an organization. For example, a machine's operating system can be configured to take its list of user IDs and group memberships from a central server. Multiple machines using the same user IDs and groups make it easier to implement a consistent set of WebSphere MQ authority checks across all systems.

This can avoid the need to create matching user IDs on one MQ system simply to allow another system to connect to a queue manager. The effect is to reduce the number of separate user accounts which the network administrator needs to maintain.

In addition to a common set of user IDs, a central directory can also be used as the basis of an additional (or substitute) Authorization Service component for WebSphere MQ authority checking. For example, WebSphere MQ Extended Security Edition provides its own Authorization Service component based upon a central LDAP directory to provide this capability. Some customers also choose to write their own Authorization Service component using an LDAP directory.

3. Configuration data can be made independent of the application and therefore the MQ network can be reconfigured without the need to recompile any applications. There is a sample application `amqslipc.c` shipped with MQ for Windows which illustrates how to do this.
4. The fact that the LDAP directory is shared by multiple systems means that all applications in an organization can be reconfigured in one place without the need to roll out changes to many machines. This considerably reduces the administrative overhead of common MQ tasks, such as moving an application to use a different queue manager in order to reduce workload on a particular system.

This SupportPac will focus on MQ policy enforcement: specifically, the use of LDAP to help administer SSL connectivity.

Since version 5.3, WebSphere MQ has provided the facility to encrypt data flowing across its channels by using the Secure Socket Layer (SSL) protocol.

In large organizations, managing the certificates used for SSL authentication can be difficult, especially if the private key corresponding to a certificate becomes compromised. Many people therefore choose to use SSL Certificate Revocation Lists (CRLs) to revoke certificates which are compromised or no longer required. A CRL is simply a list of certificates issued by a Certificate Authority (CA), plus a digital signature which proves that the CRL itself is genuine. Any certificate listed on a CRL is regarded as invalid by WebSphere MQ and, given the proper queue manager configuration, will not be accepted.

The advantages of using a centrally held CRL are:

1. The organization can easily control which certificates are trusted and used throughout all of its systems. There is no need to manually deploy lists of trusted certificates across all SSL-enabled systems.
2. It allows a rapid response to a private key security compromise. All that is required is to add the revoked certificate to the list and publish the new CRL on an LDAP server. This can immediately stop the certificate from being accepted in SSL authentication, preventing attackers from using the compromised key.
3. It can greatly simplify handling of certificates which are no longer required. When an organization changes structure, for example two departments being merged into one, it may be necessary to merge IT infrastructure. The ability to revoke old certificates for the old organization structure can help in building a new security infrastructure.

The purpose of this SupportPac is to show how to configure LDAP CRLs for use with WebSphere MQ to provide greater control over SSL authentication.

For customers who use MQIPT, there is also some information about use of CRLs with the MQIPT (MS81) SupportPac. References to MQIPT refer to MQIPT versions 1.3 through 1.3.3 unless otherwise stated.




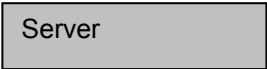
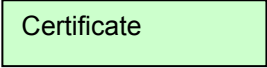
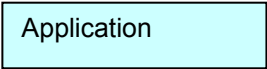
It is assumed that the reader is familiar with basic WebSphere MQ administration and the WebSphere MQ Security manual.

For the MQIPT sections, familiarity with the MS81 documentation is required, although this SupportPac is also aimed at readers who do not use MQIPT.

This SupportPac is primarily aimed at users of Windows and UNIX systems. However, the WebSphere MQ Security and System Administration manuals contain information which can assist z/OS and OS/400 users.

## Style Conventions

In this document, the following conventions and symbols are used:

Text	Text to be typed at a command line or into a file
	Arrow to represent a digital certificate signing relationship. The arrow points from the signed certificate towards the signer.
	Arrow to represent an LDAP lookup, a comparison or some other processing operation.
	Line to represent ownership, such as the relationship between an application and the SSL certificate it uses.
	Box to represent a single computer, such as a machine hosting an LDAP server.
	Box to represent an SSL certificate.
	Box to represent an application or queue manager.

## SSL and LDAP overview

### Basic Concepts

Readers who are already familiar with basic SSL concepts can skip to the next section. This section provides some background information on SSL to supplement the material in the WebSphere MQ Security manual.

SSL cryptography is based on what is known as Public Key Cryptography. In Public Key Cryptography, each entity has its own pair of keys which are mathematically related. Here, the term "entity" could mean a user, application, or any other entity which needs to participate in a secure exchange of data. The term Public Key Infrastructure (PKI) is often used to refer to the mechanism used to create and issue key pairs.

One key of the pair is known as the Public Key and is shared with the rest of the world. The other is the Private Key and is a secret known only to its owner. The keys are related in such a way that anything encrypted with one key of the pair can be decrypted using the other. However, the keys are different and it is almost impossible to derive the private key from the public key because they are related by an operation which is mathematically difficult to reverse.

A commonly used public key algorithm is RSA encryption. Here, the private portion of the key includes two prime numbers, whereas the public key is based on the product of those primes. Since finding prime factors of very large values is computationally difficult, the private key cannot be obtained from the public key. At the same time, people can encrypt data using the well-known public key safe in the knowledge that it can only be decrypted by the holder of the private key.

Conversely, anything encrypted with a private key can be decrypted using the corresponding public key. This might seem like a problem for anyone wanting to send data securely to another party. However, in a secure exchange, each party will have their own public/private key pair. Therefore if Alice wishes to send a confidential message to Bob, she would encrypt the message using her own private key and then encrypt the result a second time using Bob's public key. Thus only Bob would be able to decrypt the outer layer of protection, and if the message decrypted successfully using Alice's public key then this additionally proves that the sender of the message was Alice.

This type of mechanism can therefore provide confidentiality (by protecting the data in transit) and also authentication to prove who originated the communication.

However, there is a problem. How does Bob know that the public key he has for Alice really belongs to Alice? This is where certificates come in. The purpose of a PKI certificate is to identify the owner of a particular public key. A certificate would be issued by someone whom both Alice and Bob trust, and signed by that person as proof of Alice's identity.

A certificate therefore consists of a public key, the name of the key owner and the name and signature of a third party who asserts that the key belongs to that owner. In

PKI terms, this signature is a digital hash of the certificate which has been encrypted using the certificate issuer's private key. Anyone can use the issuer's public key to decrypt the hash and compare it with the value they compute, thus ensuring that a certificate cannot be modified by a forger. This type of certificate is known as a Personal Certificate.

An entity who signs certificates on behalf of others is known as a Certificate Authority (CA). Each CA will have its own certificate which will contain its public key and identity, plus the details of who issued that certificate. The CA's own certificate is known as a CA certificate.

This creates a chain of trust linked by issuers. Clearly the chain has to start somewhere, so the top-most certificate in a chain would be signed using its own private key. Such a certificate is called the Root Certificate or Root CA Certificate. A root certificate is always issued and signed by the same entity.

Although it can be difficult to verify the correctness of a CA certificate, modifications to a CA certificate could still be detected using the public key contained therein. Also, nobody else could sign certificates on behalf of the CA without knowing the CA's private key.

The security of an SSL exchange depends on the fact that the private key of each entity remains private to that entity. As soon as a private key is known to anyone else, there is no longer any guarantee about the identity of the person using that key pair.

Obviously it is desirable to ensure that keys which have become compromised are not used in trusted secure conversations. A Certificate Revocation List is the way to achieve this. If a certificate's private key is compromised, the CA which issued the certificate publishes a CRL to state that the certificate is “revoked”.

A CRL is a list of certificates which the original issuer of those certificates regards as no longer trustworthy. Each certificate contains a serial number which is unique to each certificate issued by that CA. A CRL contains:

- the Distinguished Name of the issuer CA
- the serial numbers of all certificates which are revoked
- a digital signature generated using the CA's private key so that the CRL can be verified as genuine
- the next update date: the date when the CRL is no longer valid and an updated CRL will be required

A certificate can only be revoked by the CA which issued it.

In general, a CRL is a list of any certificates which have been revoked by their issuer. However, there is a special type of CRL known as an ARL: Authority Revocation List. An ARL is a CRL in which the certificates revoked are CA certificates. Sometimes a top-level (root) CA will issue certificates which are to be used by an intermediate-level CA. When these intermediate-level CA certificates are revoked, the CRL which revokes them is known as an ARL. In practice, not every CA makes this distinction and you can often regard all revocation lists as being CRLs.



The standard for certificates is known as X.509. This specifies the fields and data formats which are used in certificates.

When dealing with large organizations, simple names like "Alice" and "Bob" are not unique enough. The type of name used in X.509 certificates is known as a Distinguished Name (DN) because they separate out elements of the name. Elements of a typical DN include:

- the Common Name (CN) of the entity, e.g. "Alice"
- the Organizational Unit (OU) of the entity, often a department e.g. "Accounts"
- the Organization (O) name, e.g. "IBM"
- the Locality (L) name, e.g. "Armonk"
- the State (ST) name, e.g. "New York"
- the Country (C) name, e.g. "US"

A typical Distinguished Name contains the CN, O and C fields as a minimum. In this example, the full DN for Alice might be:

CN=Alice, OU=Accounts, O=IBM, L=Armonk, ST=New York, C=US

This uniquely identifies Alice across the whole organization.

## How CRLs work with SSL

When a program initiates an SSL connection with a remote system, it takes part in a process called the SSL handshake. The handshake is a process which includes the steps documented in Chapter 4 of the MQ Security manual.

Now the use of CRLs fits into steps 3 and 5 of the handshake. CRLs are normally loaded into a central LDAP directory so that they can be accessed by any application in the organization. This enables any SSL application to see the CRL and to reject certificates quickly as they become revoked.

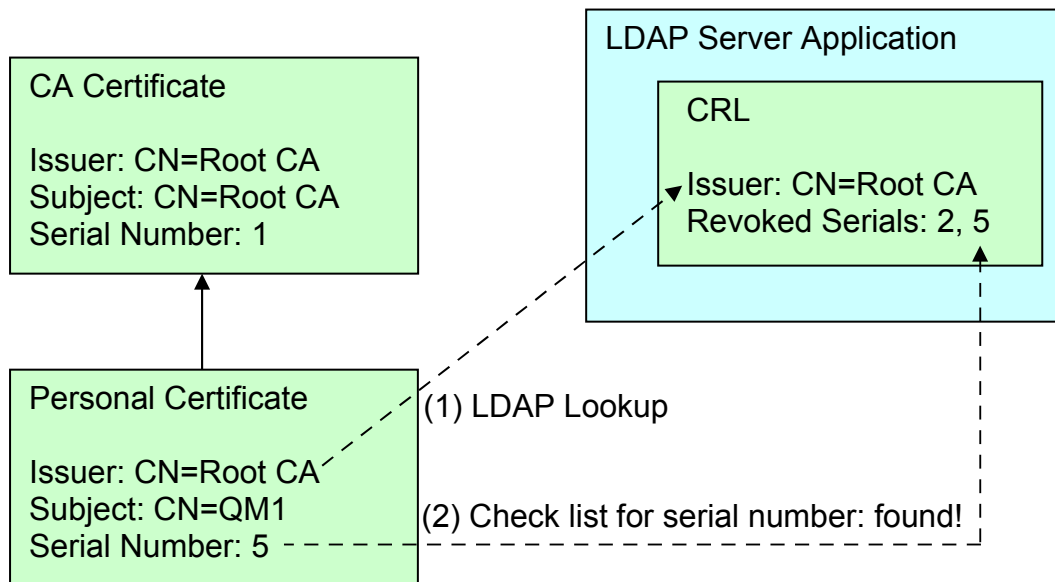
Note that an SSL client is just the entity which initiates the SSL connection. A WebSphere MQ server can act as a client for SSL purposes if it starts the connection. Therefore Requester, Sender, Server and Cluster Sender channels can all function as SSL clients because they are all capable of initiating a connection.

It is the SSL server's decision whether it requires the SSL client to identify itself.

Authentication of the SSL client is enabled by the following settings:

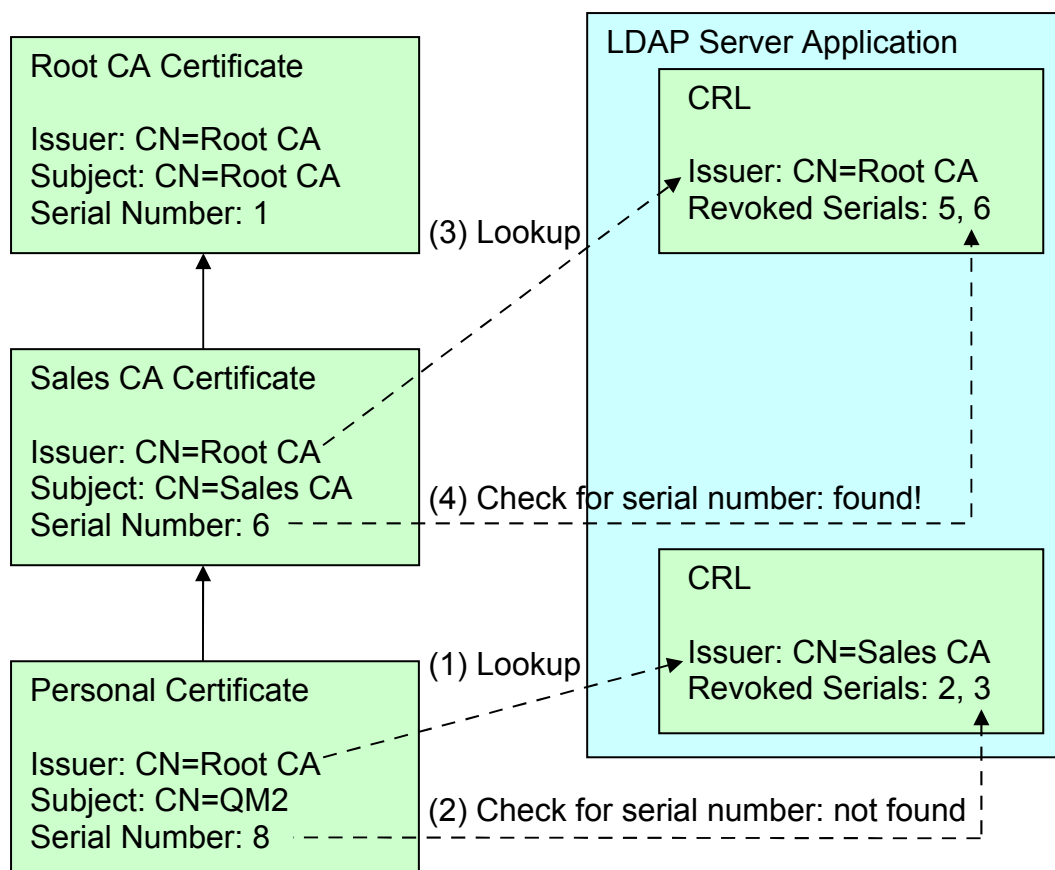
- On a WebSphere MQ Server Connection, Receiver or Cluster Receiver channel, by setting the SSLCAUTH attribute to REQUIRED. Requester channels also have a SSLCAUTH attribute because they can also act as the responder end of a channel. SSLCAUTH(REQUIRED) is the default setting.
- On an MQIPT route, by setting the SSLServerAskClientAuth property to true. The default setting is false.

Each CRL is located by finding cRLDistributionPoint objects whose DN attribute matches the Subject field of each CA certificate in the SSL certificate chain. The certificate is then checked to see if its serial number is listed in the CRL. This process is illustrated in Figure 1. In the example shown, the certificate for QM1 is revoked and will be rejected during the SSL handshake. The full DNs have been omitted for the sake of clarity. The bracketed numbers show the sequence of operations.



**Figure 1: LDAP CRL checking**

Figure 2 shows an example with a certificate chain involving multiple CAs. The root CA issued the Sales CA certificate, which in turn issued the certificate for QM2. In this example, QM2's certificate is not revoked, but the Sales CA certificate is revoked. Therefore the certificate for QM2 will not be trusted for use in an SSL handshake because every certificate in the signer chain must be valid and not revoked.



**Figure 2: LDAP CRL checking with longer certificate chains**

More information on the structure of the LDAP directory and objectClasses is provided in the next chapter.

## Using SSL Certificates

SSL certificates can either be self-signed or signed by a separate CA (which will in turn have its own certificate).

In most production systems, CA-signed certificates are preferred because they reduce the number of certificates which need to be deployed across a network and they also allow for the use of CRLs.

There is little benefit in using CRLs with self-signed certificates because the only certificate that could be revoked would be itself. The benefits of CRLs become much more apparent when managing chains of certificates.

Note also that revoking a certificate does not change the contents of the SSL certificate itself. There is nothing within a certificate which specifies that it is revoked; it is the fact that the certificate serial number is listed in a CRL published by its issuer which means that the certificate is revoked.

## Configuring the LDAP server for WebSphere MQ

### General Considerations

There are a number of issues to consider when planning for LDAP server usage. These include:

1. **Availability.** What happens if your LDAP server crashes or the network connection between the LDAP server and the MQ application is lost?

WebSphere MQ provides the capability to specify a list of LDAP servers which provide the same information. In the event of one server being unavailable, another is used instead.

WebSphere MQ version 5.3 does not cache LDAP CRLs, so if no LDAP server is available and a CRL check is required then the SSL handshake will fail and therefore the SSL connection will not succeed.

WebSphere MQ version 6 caches LDAP CRLs for up to 12 hours after the original lookup. Therefore most short-lived LDAP server failures will not affect it. Each CRL is only cached until its next update time; if this time is less than 12 hours away then the CRL will only be cached until that time.

MQIPT allows two separate LDAP servers to be configured for each route defined. The first is the primary server; the other is the backup for when the primary LDAP server is unavailable.

2. **Scalability.** Can your LDAP infrastructure handle the volume of CRL queries which will occur at peak MQ connection time?

WebSphere MQ chooses the first available LDAP server it finds in the SSL CRL name list to look up CRLs. If load balancing of queries is required across multiple LDAP servers then it will be necessary to define the name list in a different sequence for different MQ SSL systems.

3. **Updates.** Certificate Authorities issue new CRLs periodically. An organization wishing to take advantage of CRLs needs to define a process for updating the CRLs in their LDAP server as new ones are issued. The next update date of a CRL determines the latest time when a CRL is considered valid.

More information on how to configure WebSphere MQ and MQIPT to address these issues can be found in the following chapters.

### LDAP Server Software Setup

First of all, it is necessary to install an LDAP server software package. There are many such server packages available, such as IBM Tivoli Directory Server, Netscape Directory Server and OpenLDAP.

An LDAP server requires a configuration file which specifies data such as:

- the location of the LDAP database files
- the Distinguished Name of the LDAP administrator's user ID
- whether anonymous queries of the directory are permitted
- the root DN suffix of the database

An example OpenLDAP configuration file named `openldap.conf` is shipped in the `samples/config` directory of the SupportPac. In order to use this configuration file, it must be placed in the correct location before the server is started. Usually this will reside in the `/etc/openldap` directory.

OpenLDAP is usually started by running one of the following commands:

```
slapd
/sbin/service ldap start
/etc/init.d/ldap start
```

## Configuring the LDAP Directory

Once the LDAP server software is installed and configured, the next task is to populate the LDAP directory structure with data. The LDAP directory structure is defined in an LDIF file. LDIF files can be imported into the LDAP server using the tools supplied with the LDAP server software.

There are also LDAP client applications which can modify an LDAP directory. An example script named `openimport.sh` is supplied with the SupportPac in the `samples/scripts` directory. This shows how to import data using the OpenLDAP client.

Here is a typical example of an LDIF file showing the internal Certificate Authority for a fictitious company "Example":

```
# Lines beginning with a hash are comments

# Organization
dn: o=Example, c=UK
objectClass: top
objectClass: organization
o: Example

# Organizational unit
dn: ou=Sales, o=Example, c=UK
objectClass: organizationalUnit
ou: Sales

# Root CA
dn: cn=Example Sales Root CA, ou=Sales, o=Example, c=UK
objectClass: certificationAuthority
objectClass: cRLDistributionPoint
cn: Example Sales Root CA
authorityRevocationList;binary:: <CRL data>
certificateRevocationList;binary:: <CRL data>
cACertificate;binary:: <CA certificate data>
```

Here, the file defines three LDAP directory entries. A blank line marks the end of each entry.

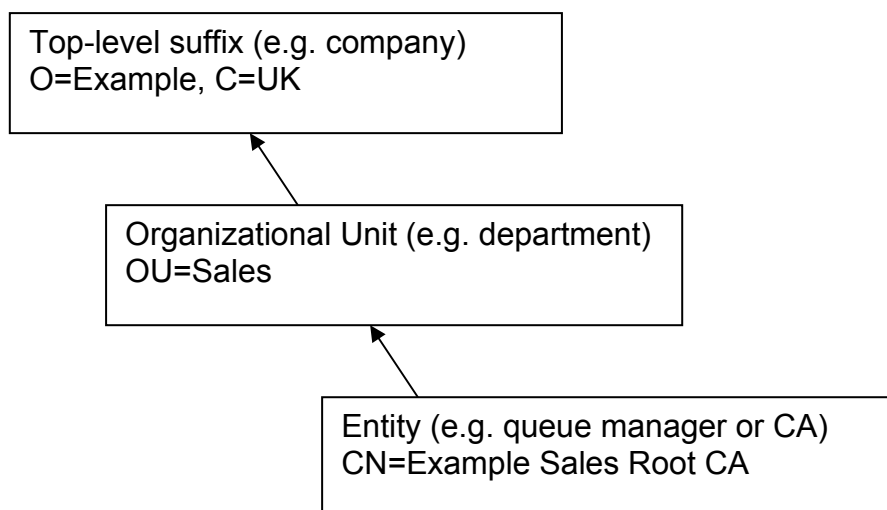
The structure of the file defines a hierarchy corresponding to the hierarchical Distinguished Name of the certificate Subject fields. Each entry in the LDAP server has at least one objectClass. It is the objectClasses that define which attributes the entry must have and which other attributes it is permitted to have. The definition of LDAP object structures is known as the LDAP Schema. On the majority of systems, the LDAP server software will be supplied with a complete schema which will not require any changes.

In this example, the CA has the Distinguished Name:

```
CN=Example Sales Root CA, OU=Sales, O=Example, C=UK
```

In order to represent this entity within the LDAP server, start with the top-level elements and build a tree down from there. An LDAP directory tree is rooted at a *suffix*. This provides a top-level starting point for the tree structure.

Once a suffix is defined, it is then possible to create elements underneath it. The links in the hierarchy are formed by the DN values in each element. Graphically, this example could be represented as:



Each element has further attributes associated with it. For LDAP CRL purposes, the most important entries are those which have objectClass `cRLDistributionPoint`. These are the entries which are queried in order to find CRLs to check certificates.

In the LDIF file, `authorityRevocationList`, `certificateRevocationList`, and `cACertificate` attributes contain binary data encoded in the same base64 encoding used for e-mail attachments and textual encodings of X.509 certificates. This format is also known as Privacy Encoded Mail, or PEM.

You should set the values for these attributes as follows:

- authorityRevocationList is a CRL containing a list of CA certificates which were issued and subsequently revoked by this CA. This will be supplied by the CA. This is where the CA's ARL goes.
- certificateRevocationList is a CRL containing a list of personal certificates which were issued and subsequently revoked by this CA. This will be supplied by the CA. This is where the CA's CRL goes.
- cACertificate is the PEM data from the certificate file for this CA.

Often a CA will only supply a single CRL. In this situation, set both the authorityRevocationList and certificateRevocationList to the same value. In any case, when setting the authorityRevocationList, certificateRevocationList and cACertificate attributes, do not include the ----BEGIN----- and -----END----- lines from certificate or CRL data: only the data in between is required.

The PEM attributes are separated from their value with a double colon (::). The double colon signifies that the attribute contains binary data or non-printing characters.

Some LDAP import utilities prefer the PEM data to be all on one line in the LDIF file; others will expect it to have 64 characters per line, with all lines other than the first line indented with white space. This depends on the LDAP software in use. Note also that a space is required between the ":" or "::" and the start of each attribute value and that trailing spaces are not trimmed from the supplied value.

The distinguished names specified in the LDIF file must match the names in the Subject field of the corresponding certificates. Note that the hierarchy of Distinguished Names need not be related to the hierarchy of certificate issuer/subjects.

Line separator handling varies among LDIF import implementations: on UNIX systems, use the dos2unix tool to ensure there are no carriage return (ASCII 13) characters in the file. If an LDAP import fails for an unexpected reason, it is likely that the LDIF file contains some unwanted whitespace characters which need to be removed.

### **Considerations for MQIPT 1.3.3**

Customers who use MQIPT with the LDAP route property set to true should note the following:

1. MQIPT supports version 1 or 2 CRLs with signature hashes using MD2, MD5 or SHA-1. There is no support for SHA-2 at present.
2. MQIPT caches LDAP CRLs for a period of time specified by the LDAPCacheTimeout route property. The default period is 24 hours. This allows resilience to LDAP server failure, but it also means that a new CRL will not be detected by an MQIPT route for up to 24 hours after the CRL is published (unless the LDAPCacheTimeout is set to a value less than 24).

If you need a route to recognize a new CRL faster than this, it is necessary to

stop and restart the route.

The way the MQIPT LDAP CRL cache works is that there is a timer which empties the LDAP cache once per every LDAPCacheTimeout interval. Cache entries do not have an individual time-to-live; when the timer expires, all cached CRLs are removed from the cache regardless of how long they have been there.

3. When MQIPT is used in SSLProxy mode, it will not validate the certificates used, nor will it check the LDAP server for any CRLs. Customers who need MQIPT to check CRLs must configure at least one of the SSLClient or SSLServer route properties to true.

### **Considerations for WebSphere MQ V5.3**

Customers using WebSphere MQ V5.3 for distributed platforms should note the following:

1. There must be an entry defined in the LDAP server for every CA certificate in the certificate chain used for an SSL connection. The entry must have an objectClass of cRLDistributionPoint and a DN attribute matching the Subject name of the CA certificate. If CRL checking is enabled and any LDAP entry is missing for a certificate chain then the SSL connection will fail and an AMQ9633 message will be recorded in the error log.
2. WebSphere MQ V5.3 does not support version 3 of the LDAP protocol. Many LDAP servers do not allow earlier LDAP protocol versions by default. For example, on OpenLDAP-based servers it is necessary to add the following line to the global section of the LDAP server configuration file:

```
allow bind_v2
```

3. Digital signatures based upon SHA-2 are not supported with GSKit 6 and therefore certificates or CRLs signed using SHA-2 cannot be used with WebSphere MQ version 5.3. Customers who require these hashes must upgrade to WebSphere MQ version 6 and GSKit 7.0.3.25 or later.
4. If the SSL CRL configuration is changed, the change might not be picked up immediately. This is because WebSphere MQ caches some aspects of SSL configuration. If any SSL channels have been run since the queue manager started then the only guaranteed way to ensure that the LDAP servers are used is to quiesce and restart the queue manager.

### **Considerations for WebSphere MQ V6**

Customers using WebSphere MQ V6 for distributed platforms should note the following:

1. There must be an entry defined in the LDAP server for every CA certificate in the certificate chain used for an SSL connection. The entry must have an



## MQ01 – Using LDAP and SSL with WebSphere MQ and MQIPT

objectClass of cRLDistributionPoint and a DN attribute matching the Subject name of the CA certificate. If CRL checking is enabled and any LDAP entry is missing for a certificate chain then the SSL connection will fail and an AMQ9633 message will be recorded in the error log.

2. WebSphere MQ V6 caches LDAP CRLs for up to 12 hours. This allows resilience to LDAP server failure.

When a REFRESH SECURITY TYPE(SSL) MQSC command is issued on a queue manager, the cache is emptied.

3. Customers using certificates with SHA-2 digital signatures must install GSKit 7.0.3.25 or later.
4. If the SSL CRL configuration is changed, the change might not be picked up immediately. This is because WebSphere MQ caches some aspects of SSL configuration, such as the host names of the LDAP servers to use. If any SSL channels have been run since the queue manager started then it is necessary to run the following MQSC command:

```
REFRESH SECURITY TYPE(SSL)
```

In addition to clearing the cached SSL configuration, this will also close and restart all SSL channels.

## Configuring LDAP for WebSphere MQ servers

The main steps to configure a WebSphere MQ server for LDAP CRL use are:

1. Define the AUTHINFO objects, one per LDAP server. Each server must contain the same CRL data.
2. Define a NAMELIST object which lists all of the AUTHINFO objects by name.
3. Alter the queue manager's SSLCRLNL attribute to specify the name list object.

For example, suppose your LDAP server is ldap.my.org and the server runs on port 389. The MQSC definition commands would be:

```
DEFINE AUTHINFO (AUTHINFO.LDAP) AUTHTYPE (CRLLDAP)
CONNAME ('ldap.my.org (389) ')

DEFINE NAMELIST (NAMELIST.LDAP) NAMES (AUTHINFO.LDAP)

ALTER QMGR SSLCRLNL (NAMELIST.LDAP)
```

If you have two LDAP servers, ldap1.my.org using port 1389 and ldap2.my.org using port 5389, the definition commands would be:

```
DEFINE AUTHINFO (AUTHINFO.LDAP1) AUTHTYPE (CRLLDAP)
CONNAME ('ldap1.my.org (1389) ')

DEFINE AUTHINFO (AUTHINFO.LDAP2) AUTHTYPE (CRLLDAP)
CONNAME ('ldap2.my.org (5389) ')

DEFINE NAMELIST (NAMELIST.LDAP) NAMES (AUTHINFO.LDAP1,
AUTHINFO.LDAP2)

ALTER QMGR SSLCRLNL (NAMELIST.LDAP)
```

If any SSL channels have previously been run on the queue manager then it is necessary to ensure that the queue manager picks up the new configuration. Refer to the sections Considerations for WebSphere MQ V5.3 and Considerations for WebSphere MQ V6 to see how to do this for your WebSphere MQ version.

The above examples assume that your LDAP servers allow anonymous queries to the directory structure. In many organizations, access to the LDAP server is restricted to authenticated access only. For this reason, you may specify optional LDAPUSER and LDAPPWD attributes for each AUTHINFO object. These will be used by WebSphere MQ to authenticate it to the LDAP server.

## Configuring LDAP CRLs for WebSphere MQ clients

### C and C++ Clients

There are three alternative ways to use LDAP CRLs in these client environments:

#### 1. Using a Client Channel Definition Table.

In order for this to work, set the following environment variables on the client system:

- MQCHLLIB - the directory containing the channel definition table file
- MQCHLTAB - the name of the file, usually AMQCLCHL.TAB
- MQSSLKEYR - the stem name of the client's SSL key repository. This is the full path and file name but without the .kdb suffix.

When the client makes its MQCONN call, the LDAP server information will be read from the client channel definition table. Any SSL certificates required will be read from the client key repository.

If the remote SVRCONN channel specifies SSLCAUTH(REQUIRED) then the file referenced by MQSSLKEYR must contain a personal certificate which can be used to identify the client for the SSL connection.

For the client to do LDAP CRL checking itself, the CCDT file must have been created on a queue manager which has an SSLCRLNL set up. When a queue manager is configured for SSL CRL checking, the LDAP server details are copied into the CCDT file automatically.

#### 2. Using Microsoft Active Directory on Windows.

The setmqcsp command is used to publish WebSphere MQ channel definitions to Active Directory. Also, the setmqcrl command is used to configure LDAP CRLs in the Active Directory. For more information on these commands, refer to the WebSphere MQ System Administration Guide.

#### 3. Using the MQCONNX call.

In order for this to work, initialize the MQCNO structure as follows:

- a. Set the ClientConnOffset or ClientConnPtr to point to a piece of memory where the MQCD channel definition is stored. In the C++ language bindings the MQCD is represented by an instance of the ImqChannel class.
- b. Within the MQCD structure, set the SSLCipherSpec to the required cipherspec. If a peer name filter is required, set the SSLPeerNamePtr value.

## MQ01 – Using LDAP and SSL with WebSphere MQ and MQIPT

- c. Within the MQCNO structure, set the SSLConfigOffset or SSLConfigPtr to point to a piece of memory where the MQSCO SSL Configuration Options structure is located.
- d. In the MQSCO structure, set the KeyRepository field to contain the stem name of the key repository database.
- e. In the MQSCO structure, set the AuthInfoRecOffset or AuthInfoRecPtr to point to a piece of memory containing at least one MQAIR Authentication Information Record. In the C++ language bindings, the MQAIR is represented by an instance of the ImqAuthenticationInformation class.
- f. In each MQAIR record, specify the details of the LDAP server.
- g. Ensure that the AuthInfoRecCount field in the MQSCO correctly reflects the number of MQAIR records referenced by the MQSCO.

A sample sslput.c has been provided to illustrate how to do this. This is based on the standard amqsput0.c sample program supplied with WebSphere MQ. The sample can be compiled as a client application for use on either Windows or UNIX using the appropriate compiler command as shown in Part 3 of the WebSphere MQ Application Programming Guide.

Note that the MQSERVER environment variable cannot be used to establish SSL channels because it provides no means of specifying an SSL cipherspec.

When using a WebSphere MQ client application, the server side of the channel might require SSL authentication as explained earlier. If authentication is required, the MQ client code must select a certificate from its key repository to send. This is done automatically on behalf of the application by the MQ client libraries.

There are two ways in which the MQ client determines which certificate to send:

1. **Certificate label.** For a WebSphere MQ client, the certificate usually must have a name which depends upon the name of the user who invokes the application. Therefore, if a user logged on as Fred runs the client then the certificate has the label `ibmwebspheremqfred` in order to be sent.
2. **Default certificates.** The alternative is to use the GSKit tools to set the default certificate in the key repository. The default certificate is then sent regardless of which user ID the application is running under. Contact your IBM Service representative to determine if your version of WebSphere MQ supports default certificates.

On versions of WebSphere MQ which support default certificates, the default certificate is only used if no `ibmwebspheremq`-type certificate is found matching the application's user ID. Therefore a labeled certificate will take precedence over the default.

For more details on the C++ language bindings and how the objects relate to the C structures, refer to Appendix B of the WebSphere MQ C++ manual.

## Java Clients

A Java client application in bindings mode will use the native MQI client library to connect to the queue manager. The recommended approach for a bindings client application is to use a client channel definition table and set the MQSSLKEYR, MQCHLLIB and MQCHLTAB environment variables.

For a non-bindings Java application, the SSL CRL can be configured in two different ways:

1. For a non-JMS application, the MQEnvironment sslCertStores value can be set to an ArrayList of CertStore objects. One of the objects in the list should be a CertStore of type LDAP created using an LDAPCertStoreParameters object containing the LDAP server name and port.
2. For a JMS application, the SSLCRL transport property can be set to a space separated list of LDAP servers in the following form:

```
ldap://hostname[:port]
```

For example:

```
ldap://myldap1.ibm.com ldap://myldap2.ibm.com:5389
```

This property is only used if the connection factory's TRANSPORT property is set to CLIENT (or DIRECT).

Customers using Java applications where CRL checking is required on the client side should refer to the WebSphere MQ Java manual for further details.

## Other Programming Languages

The WebSphere MQ manuals contain details about how to use LDAP CRLs with other programming languages such as .NET and Visual Basic.

## Sample Scenarios

This SupportPac is supplied with a number of sample SSL certificates and CRLs, plus the LDAP server configuration data files to support them. This chapter gives a brief overview of each sample scenario to assist customers with understanding and testing the sample materials.

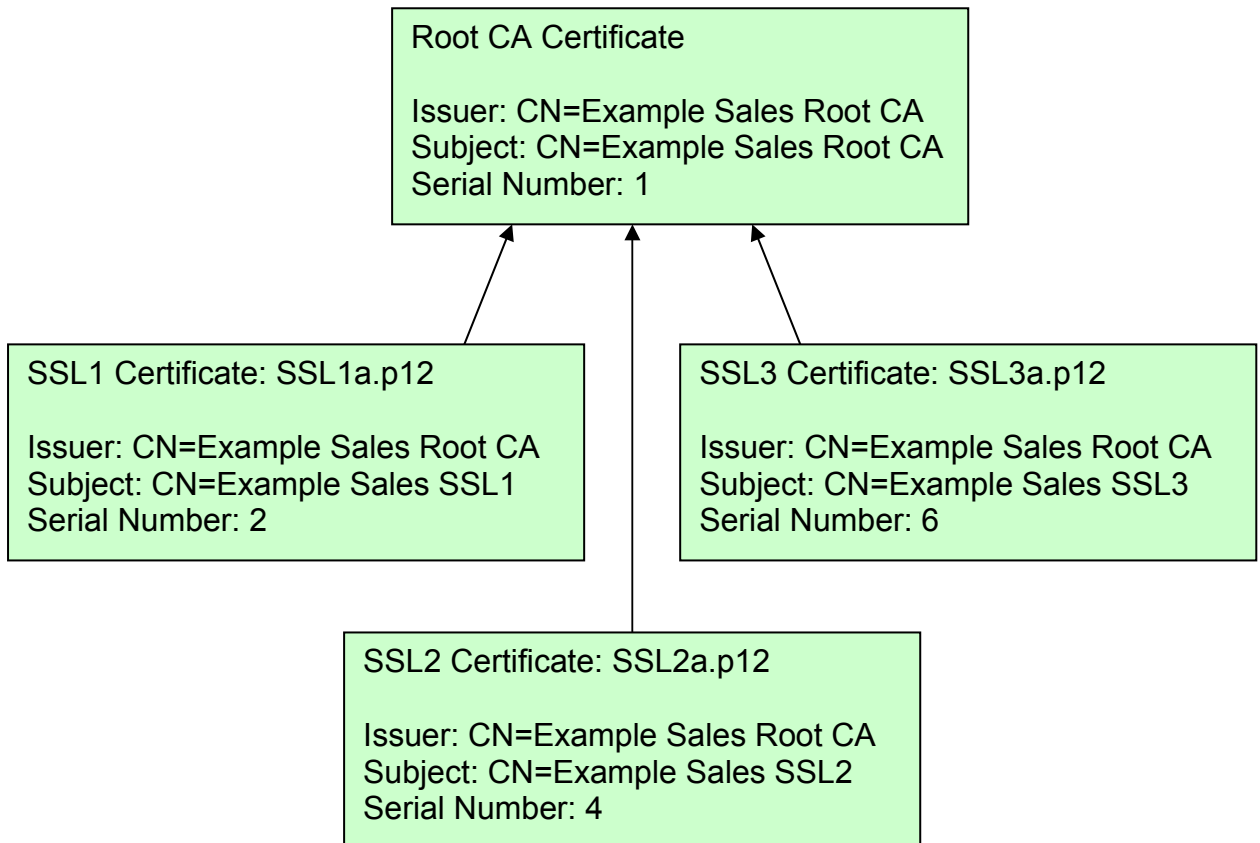
Each scenario is independent and contains different certificates. Where a password is required to access the sample data files, this password has been set to `ldaptest`. Some samples include automated scripts to show the commands to set up each scenario. Before running the scripts provided on Unix platforms, please set `JAVA_HOME` correctly for GSKit as per the WebSphere MQ Security manual.

### Scenario 1: Queue Managers

In this scenario, there are three queue managers which need to communicate within the Sales team of an organization. There is a single root Certificate Authority which has issued all of the SSL certificates for these queue managers.

For each queue manager, two different SSL certificates have been provided. One certificate has been revoked by the root CA, whereas the other is valid. This allows you to test the use of a revoked versus a valid certificate for each queue manager.

Figure 3 shows an outline of the revoked certificates for this configuration. Full DNs have been omitted for clarity. For each revoked certificate, there is a corresponding `SSLnb.p12` file containing a non-revoked certificate for the same queue manager.



**Figure 3: Scenario 1 sample certificates**

In order to set up this scenario, the steps are:

1. Create three queue managers: SSL1, SSL2 and SSL3. Start each queue manager along with a listener for each.
2. Set up the SSL key repositories by adding the root CA certificate to each and then importing the PKCS12 certificate and private key for the corresponding queue manager.
3. Start an LDAP server and import the supplied config.ldif into it. Notice how the CA data values in config.ldif correspond to the CA's certificate (root.crt) and CRL (root.crl).
4. Configure each queue manager to use the LDAP server as described in Configuring LDAP for WebSphere MQ servers earlier in this document.
5. Define some channels between the queue managers using the normal WebSphere MQ administration commands. Start the channels.

Channel pairs will refuse to start when a revoked SSL certificate is used, subject to the considerations stated in the Considerations for WebSphere MQ V5.3 and Considerations for WebSphere MQ V6 sections earlier. Note also that if the LDAP server is unavailable or has not been correctly configured then this can also prevent channels from starting.

Some example scripts, setupv53.sh, setupv6.sh and setupv6.bat have been provided to automate the majority of this setup and create three queue managers on the same machine. The .sh files are for use on Unix systems, whereas the .bat files are for Windows setup. These scripts install a revoked certificate for SSL1, so that channels to that queue manager should fail to start.

All that you must do is set up the LDAP server and provide the hostname and port for the queue manager to use when querying the LDAP server. If the LDAP server requires a user name and password to query it, then you must alter the AUTHINFO objects to set the LDAPUSER and LDAPPWD values.

Once all channels are started, the channels to SSL1 will fail to start and enter the RETRYING state. If you delete the revoked certificate from the SSL1 repository and replace each certificate with its non-revoked SSL*nb*.p12 version, then SSL channels will be able to start.

## Scenario 2: Queue Managers

This scenario is almost identical to Scenario 1, except that there is an intermediate CA in between the root and the personal certificates. Each personal certificate has been signed by the Sales CA which has in turn been signed by the Root CA.

The setup procedure for this scenario is very similar to the first, except there is an extra step to add the Sales CA certificates to each SSL key database. Also, note the

contents of the LDIF file contain extra entries to define both CAs. This illustrates how to structure the LDAP directory entries for longer certificate chains.

### Scenario 3: Client Application

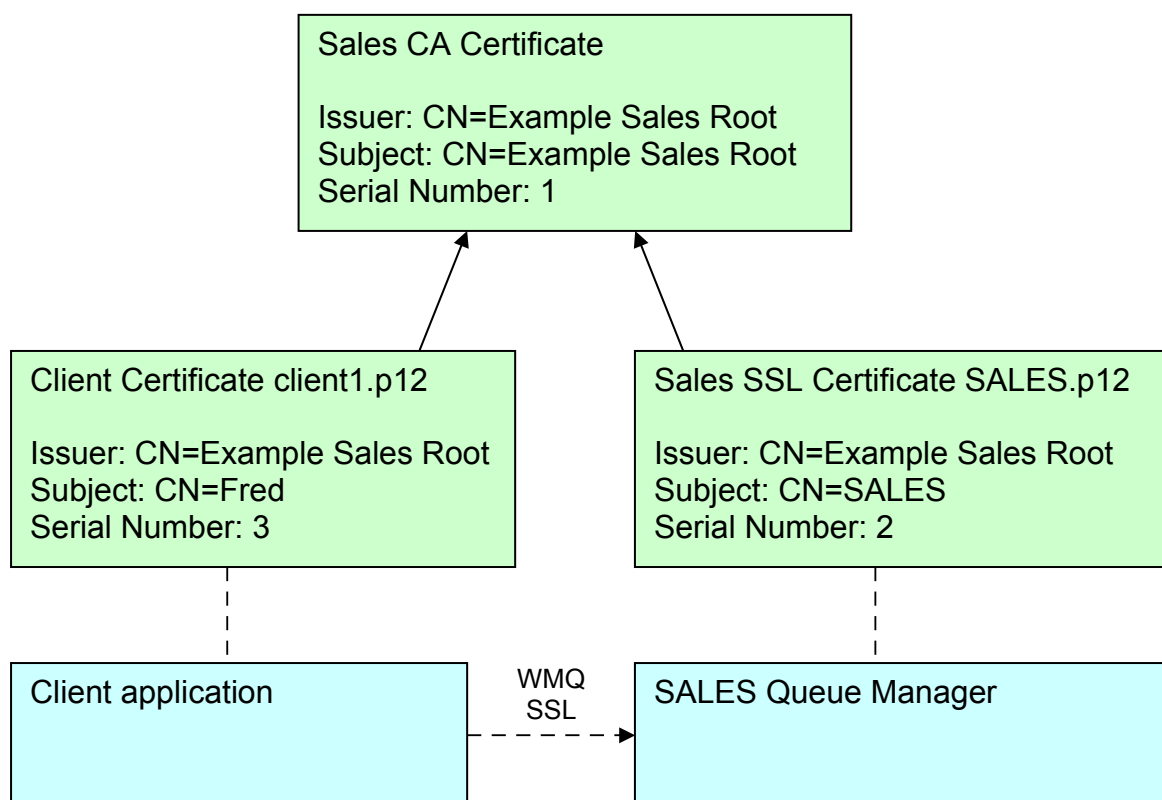
This scenario shows a client application which needs to connect to a queue manager named SALES. In this situation, traveling sales team members log in remotely to the company network in order to upload their daily results onto the queue manager.

The Sales organization has issued each team member with their own SSL certificate to identify them to the central system. Since certificates are kept on company laptops which are at risk of being stolen, the company wants to be able to revoke a certificate immediately in the event of a machine being reported stolen. This is to stop a stolen laptop from being used to access the company sales queue manager.

In this example, the queue manager has its own certificate, and each client machine has its own personal certificate also. All certificates have been signed by the same CA.

Since the Sales department does not trust external connections, the administrator has set up the server-connection channel to require SSL client authentication. This means that the client must send a certificate to identify itself.

Figure 4 shows how the client application connects to the queue manager.



**Figure 4: Client to server scenario**



## MQ01 – Using LDAP and SSL with WebSphere MQ and MQIPT

The certificate in client1.p12 is listed as revoked in the Sales CRL. There is also a client2.p12 which provides a certificate which is not revoked.

The steps to set up this scenario are:

1. Create the SALES queue manager, start it and run a listener for it.
2. Create the key repository for the SALES queue manager, add the CA certificate and import the SALES.p12 personal certificate and private key.
3. Import the supplied config.ldif file into the LDAP server.
4. Configure the queue manager to use the LDAP server according to the earlier instructions in Configuring LDAP for WebSphere MQ servers.
5. Create a key repository for the client to use. Add the CA certificate to it and import the client1.p12 file for the client's personal certificate and private key. Copy the key repository files to the client machine, taking care to transfer the files in binary mode if FTP is used.

Set the certificate label to correspond to the name of the user ID under which the client application will run. One way to ensure the correct label is to specify the `-new_label` parameter on the GSKit command line.

In the supplied client1.p12 and client2.p12 the certificates are labeled `ibmwebspheremqfred`, so unless `-new_label` is used to relabel them at import time (or the certificate is imported as the default and your WebSphere MQ version supports default certificates) then the client application should be run as user "fred".

6. Ensure that the user ID under which the client application will connect has appropriate MQI permissions to connect to the queue manager and access any queues required. For example:

```
setmqaut -m SALES -t qmgr -p fred +connect
setmqaut -m SALES -t q -n QUEUE -p fred +put
```

Example scripts named `setupv53.sh`, `setupv53.bat`, `setupv6.sh` and `setupv6.bat` are supplied to do the server-side of this configuration. They will also create a client key repository files in their current working directory. You can then configure the LDAP server using the supplied `config.ldif` file and transfer the client key repository files to the client system.

The supplied `sslput.c` sample can be used as the client application to test whether you can connect or not. Since the client1.p12 certificate is revoked, the `MQCONN` call will fail for that certificate. If the client2.p12 certificate is used instead, the connection will succeed. Some example `runclient` scripts are provided to show how the `sslput` program could be invoked to test this.

You can use the WebSphere MQ certificate management tools to delete one certificate and import another in its place into the client key repository. Refer to the WebSphere MQ Security manual for more information on deleting and importing certificates.

### Scenario 4: Client Application and MQIPT

This scenario is similar to Scenario 3, in that it involves MQ clients connecting into a SALES queue manager. However, in this setup the organization has imposed a firewall setup which prohibits the presence of an externally visible queue manager.

The purpose of this scenario is to demonstrate how MQIPT can be configured to allow such SSL connections to pass through port 443 to the target queue manager. This configuration is shown in Figure 5.

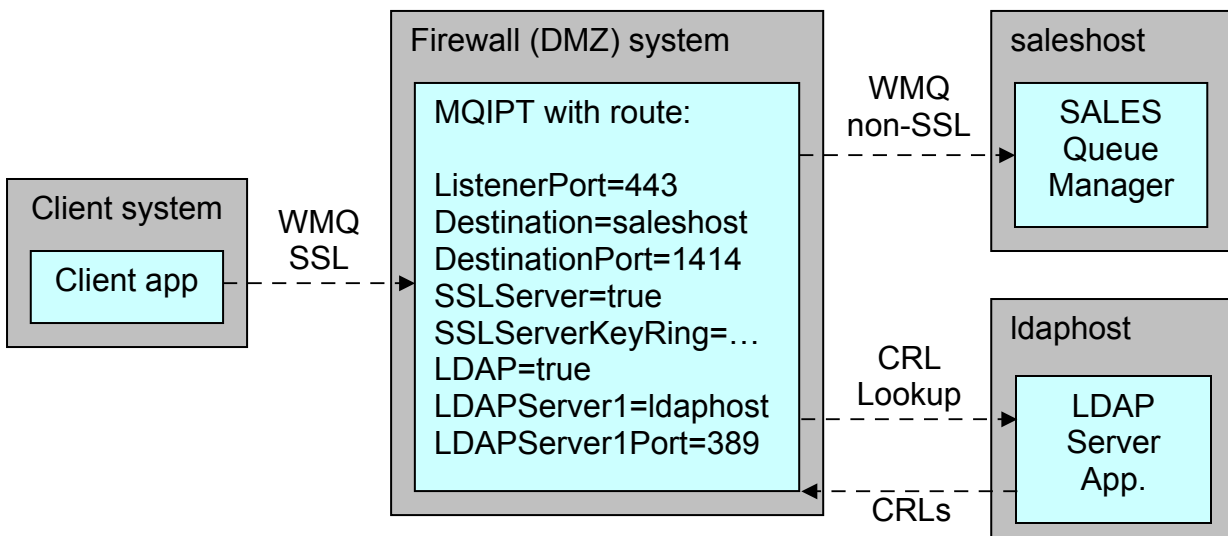


Figure 5: MQIPT LDAP CRL scenario

In this example, the clients use SSL over the untrusted internet to reach the MQIPT system in the Demilitarized Zone (DMZ) of the corporate network. Once the connection has reached MQIPT, a plain WebSphere MQ client connection is made to the listener of the back-end SALES queue manager.

The SSL certificate chain has the same structure as for Scenario 3: a common CA has issued certificates for the client and for MQIPT. In this case, it is MQIPT which is the SSL server instead of the queue manager.

In order to complete the SSL handshake, the MQIPT route must have its own SSL certificate to identify itself to the client. This certificate and its private key must be contained in a PKCS12 file referenced by the SSLServerKeyRing route property.

If the SSLServerAskClientAuth route property is set, then the client application is required to send an SSL certificate to identify itself. This means that the route also requires a copy of the common CA certificate so that it can validate the client's SSL certificate. This certificate should be in a PKCS12 file named by the SSLServerCAKeyRing route property.

## MQ01 – Using LDAP and SSL with WebSphere MQ and MQIPT

One way to generate such a file is to use the KM utility supplied with MQIPT to “import” the CA certificate and save a PKCS12 token file.

Alternatively, other tools could generate such a file. Here is an example command to generate a PKCS12 file with just a CA certificate using OpenSSL:

```
openssl pkcs12 -export -in root.crt -nokeys  
-cacerts -out root.p12 -noiter -nomaciter
```

The steps to set up this scenario are:

1. Create the SALES queue manager, start it and run a listener for it.
2. Create the key repository for the SALES queue manager, add the CA certificate and import the SALES.p12 personal certificate and private key.
3. Import the supplied config.ldif file into the LDAP server.
4. Create the MQIPT route configuration file. An example mqipt.conf is supplied for this scenario, but it will need editing to match your configuration before it can be used.
  - a. Set the LDAPServer1 and LDAPPort1 route properties to the correct TCP/IP host name and port for your LDAP server.
  - b. Set the Destination and DestinationPort route properties to the correct TCP/IP host name and port for the SALES queue manager.
  - c. Create a PKCS12 key database file to store the SSL certificate and private key belonging to the MQIPT server. In this case, a suitable MQIPT.p12 file has been supplied.
  - d. Use the mqiptPW utility to save the password for the PKCS12 into a stash file for MQIPT to use. In this case, a suitable MQIPT.pwd file has been supplied.
  - e. Set the SSLServerKeyRing route property to the path and filename of the MQIPT.p12 file. Also set the SSLServerKeyRingPW route property to the path and filename of the MQIPT.pwd file.
  - f. If you wish to require a client to send its own personal certificate then set the SSLServerAskClientAuth route property to `true`. You should also create a PKCS12 file containing the CA certificates which can be used to verify the certificate sent by the client and then set the SSLServerCAKeyRing and SSLServerCAKeyRingPW route properties accordingly.
5. Create a key repository for the client to use. Add the CA certificate (root.crt) to it and import the client1.p12 file for the client’s personal certificate and private key. Copy the key repository files to the client machine, taking care to transfer the files in binary mode if FTP is used.

Refer to the Configuring LDAP CRLs for WebSphere MQ clients section for more information on client certificate usage. The WebSphere MQ Security manual explains the command syntax for these certificate operations.

6. Ensure that the user ID under which the client application will connect has appropriate MQI permissions to connect to the queue manager and access any queues required. For example if the client is run as user “fred”:

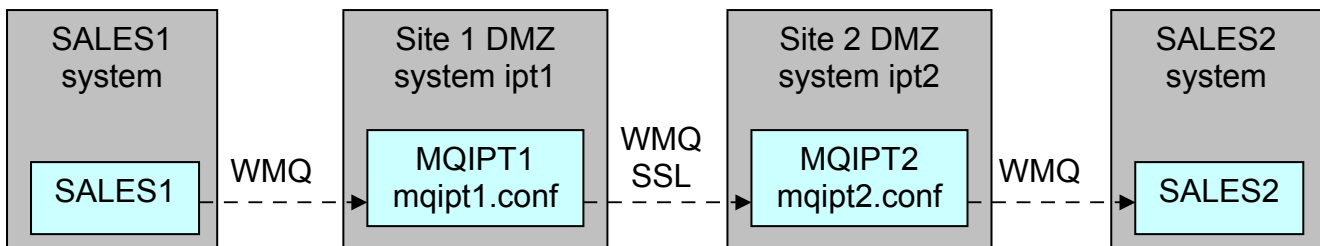
```
setmqaut -m SALES -t qmgr -p fred +connect
setmqaut -m SALES -t q -n QUEUE -p fred +put
```

### Scenario 5: Queue Managers and MQIPT

In this setup, the organization has a sales team split across two sites. Each site has a queue manager: SALES1 at one site and SALES2 at the other. The two sites are geographically remote and are only linked by the internet. The company wishes to encrypt data flowing between the two queue managers to ensure that it remains secret.

The IT department has insisted that SSL CRLs are used in case one site becomes compromised by an attacker.

One way to implement this scenario is shown in Figure 6. For simplicity, this diagram only shows the MQ channel and MQIPT route definitions necessary to send messages from queue manager SALES1 to SALES2.



**Figure 6: MQIPT scenario linking two queue managers**

In this example, a single CA has signed some certificates for the MQIPT1 and MQIPT2 systems. These are stored in SALES1.p12 and SALES2.p12 respectively.

Here, the queue managers do not use SSL directly and are unaware that MQIPT is encrypting their data across the internet. The SALES1 certificates are used by MQIPT1 and the SALES2 certificates are used by MQIPT2. One advantage of this approach is that the processor-intensive SSL operations are offloaded from the queue manager system onto a separate DMZ machine.

Sample MQIPT route configurations have been supplied in mqipt1.conf and mqipt2.conf. These illustrate the main settings which need to be configured and include some example port numbers and file names to correspond with the supplied SSL certificate files.

The steps required to set up this scenario are:

## MQ01 – Using LDAP and SSL with WebSphere MQ and MQIPT

1. Create queue managers SALES1 and SALES2 on their respective systems.
2. On SALES1, define a sender channel to the MQIPT1 system. Ensure that the CONNAME attribute contains the correct host name and port number. For example:

```
DEFINE CHANNEL(TO.SALES2) CHLTYPE(SDR)
TRPTYPE(TCP) CONNAME('ipt1.example.org(1515)')
XMITQ(SALES2)

DEFINE QLOCAL(SALES2) USAGE(XMITQ)
```

There is no need to specify the SSLCIPH channel attribute because MQIPT will encrypt the data across the internet. This operation is transparent to the queue manager.

3. Copy the supplied mqipt1.conf file to the Site 1 DMZ system along with the SALES1.p12, SALES1.pwd, SALESCA.p12 and SALESCA.pwd files. Then copy the supplied mqipt2.conf file to the Site 2 DMZ system along with the SALES2 and SALESCA files. If FTP is used, ensure that the files are transferred in binary mode.
4. Configure the MQIPT1 route properties:
  - a. Ensure that the ListenerPort property matches the port number in the SALES1 sender channel definition.
  - b. Ensure that the Destination and DestinationPort properties refer to the host name and port number of the MQIPT2 system in Site 2.
  - c. Ensure that the SSLClient property is set to true because this route makes outgoing SSL connections.
  - d. Ensure that the SSLClientKeyRing property refers to a PKCS12 file containing the personal certificate to be used to identify the MQIPT1 system to the MQIPT2 system. For this scenario, the supplied SALES1.p12 is sufficient.
  - e. Set the SSLClientCAKeyRing property to refer to a PKCS12 file containing a CA certificate which can be used to verify the MQIPT2 certificate. For this scenario, this certificate is supplied in root.crt and a SALESCA.p12 file containing this certificate has been provided.
  - f. Set the LDAP property to true and set the LDAPServer1 and LDAPPort1 properties to refer to the LDAP server.
5. Configure the MQIPT2 route properties:
  - a. Ensure that the ListenerPort property matches the port number in MQIPT1's Destination property.

- b. Ensure that the Destination and DestinationPort properties refer to the host name and port of the SALES2 queue manager's listener.
- c. Ensure that the SSLServer property is set to true because this route accepts incoming SSL connections.
- d. Ensure that the SSLServerKeyRing property refers to a PKCS12 file containing the personal certificate to be used to identify the MQIPT2 system to the MQIPT1 system. For this scenario, the supplied SALES2.p12 is sufficient.
- e. Set the SSLServerCAKeyRing property to refer to a PKCS12 file containing a CA certificate which can be used to verify the MQIPT1 certificate. For this scenario, this certificate is supplied in root.crt and a SALESCA.p12 file containing this certificate has been provided.
- f. Set the SSLServerAskClientAuth property to true. This will ensure that MQIPT2 checks MQIPT1's certificate validity.
- g. Set the LDAP property to true and set the LDAPServer1 and LDAPPort1 properties to refer to the LDAP server.

6. On SALES2, define a receiver channel to match the definition on SALES1.

An example MQSC command is:

```
DEFINE CHANNEL (TO.SALES) CHLTYPE (RCVR) TRPTYPE (TCP)
```

As with the definition on SALES1, this channel definition does not need the SSLCIPH property set. This is because MQIPT2 will decrypt the data before it arrives at SALES2.

7. On SALES2, start a TCP listener on the port number specified in the MQIPT2 DestinationPort route property. For example:

```
runmqtsr -m SALES2 -t tcp -p 1717
```

8. Import the supplied config1.ldif into your LDAP server. This configuration contains an empty CRL for the Sales root CA which will allow the connection to succeed.
9. Start the MQIPT1 and MQIPT2 routes. You can now start the sender channel on SALES1. If all is correctly configured, the channel will enter the RUNNING state.

There is also a supplied config2.ldif file which includes a different CRL which revokes the certificate for MQIPT1. If this LDIF file is imported into the LDAP server instead of config1.ldif then the sender/receiver channel pair will fail to start.