Altibase® Tools & Utilities

Altibase SSL/TLS User's Guide

Altibase Tools & Utilities Altibase SSL/TLS User's Guide

Release 7.1

㈜알티베이스

08378 서울시 구로구 디지털로 306 대륭포스트타워II 10층

전화: 02-2082-1114 팩스: 02-2082-1099

고객서비스포털: http://support.altibase.com

homepage: http://www.altibase.com

서문

이 매뉴얼에 대하여

이 매뉴얼은 Altibase에 SSL/TLS 통신을 설정하고 사용하는 방법에 대해 설명한다.

대상 사용자

이 매뉴얼은 다음과 같은 Altibase 사용자를 대상으로 작성되었다.

- 데이터베이스 관리자 및 사용자
- 프로그래밍 개발자
- 기술지원부
- 보안 담당자

다음과 같은 배경 지식을 가지고 이 매뉴얼을 읽는 것이 좋다.

- 컴퓨터, 운영 체제 및 운영 체제 유틸리티 운용에 필요한 기본 지식
- 관계형 데이터베이스 사용 경험 또는 데이터베이스 개념에 대한 이해
- 컴퓨터 프로그래밍 경험
- 데이터베이스 서버 관리, 운영 체제 관리 또는 네트워크 관리 경험
- 분산 환경에서의 데이터의 저장, 관리 및 처리와 관련된 지식
- SSL/TLS 프로토콜 및 보안 통신에 대한 기본 지식

소프트웨어 환경

이 매뉴얼은 데이터베이스 서버로 Altibase 버전 7.1을 사용한다는 가정 하에 작성되었다.

이 매뉴얼의 구성

이 매뉴얼은 다음과 같이 구성되어 있다.

- 제 1장 Altibase SSL/TLS 소개
  이 장에서는 AltibaseSSL/TLS의 개념 및 특징에 대해 설명한다.
- 제 2장 SSL 설치 및 시작
  이 장에서는 Altibase에서 SSL 통신을 사용하기 위한 소프트웨어 요구사항 및 설치하는 방법에 대하여 설명한다.
- 제 3장 SSL 연결 관리
  이 장에서는 SSL 기능을 사용할 때 아래의 방법으로 모니터링을 하고, 사용자에 대한 통신 보안을 강화하는 등의 관리 방법을 설명한다.
- 부록 A. Altibase SSL 샘플

문서화 규칙

이 절에서는 이 매뉴얼에서 사용하는 규칙에 대해 설명한다. 이 규칙을 이해하면 이 매뉴얼과 설명서 세트의 다른 매뉴얼에서 정보를 쉽게 찾을 수 있다.

여기서 설명하는 규칙은 다음과 같다.

- 샘플 코드 규칙

샘플 코드 규칙

코드 예제는 SQL, Stored Procedure, iSQL 또는 다른 명령 라인 구문들을 예를 들어 설명한다.

아래 테이블은 코드 예제에서 사용된 인쇄 규칙에 대해 설명한다.

규칙	의미	예제
[ ]	선택 항목을 표시	VARCHAR [(*size*)][[FIXED |] VARIABLE]

규칙	의미	예제
{ }	필수 항목 표시. 반드시 하나 이상을 선택해야 되는 표시	{ ENABLE | DISABLE | COMPILE }
|	선택 또는 필수 항목 표시의 인자 구분 표시	{ ENABLE | DISABLE | COMPILE } [ENABLE | DISABLE | COMPILE]
. . .	그 이전 인자의 반복 표시 예제 코드들의 생략되는 것을 표시	SQL> SELECT ename FROM employee; ENAME ---------- SWNO HJNO HSCHOI . . . 20 rows selected.
그 밖에 기호	위에서 보여진 기호 이 외에 기호들	EXEC :p1 := 1; acc NUMBER(11,2);
기울임 꼴	구문 요소에서 사용자가 지정해야 하는 변수, 특수한 값을 제공해야만 하는 위치	SELECT * FROM *table_name*; CONNECT *userID*/*password*;
소문자	사용자가 제공하는 프로그램의 요소들, 예를 들어 테이블 이름, 칼럼 이름, 파일 이름 등	SELECT ename FROM employee;
대문자	시스템에서 제공하는 요소들 또는 구문에 나타나는 키워드	DESC SYSTEM_.SYS_INDICES_;

관련 자료

자세한 정보를 위하여 다음 문서 목록을 참조하기 바란다.

- Administrator’s Manual
- General Reference
- SQL Reference
- iSQL User’s Manual
- Error Message Reference

알티베이스는 여러분의 의견을 환영합니다.

이 매뉴얼에 대한 여러분의 의견을 보내주시기 바랍니다. 사용자의 의견은 다음 버전의 매뉴얼을 작성하는데 많은 도움이 됩니다. 보내실 때에는 아래 내용과 함께 고객서비스포털(http://support.altibase.com/kr/)로 보내주시기 바랍니다.

- 사용 중인 매뉴얼의 이름과 버전
- 매뉴얼에 대한 의견
- 사용자의 성함, 주소, 전화번호

이 외에도 알티베이스 기술지원 설명서의 오류와 누락된 부분 및 기타 기술적인 문제들에 대해서 이 주소로 보내주시면 정성껏 처리하겠습니다. 또한, 기술적인 부분과 관련하여 즉각적인 도움이 필요한 경우에도 고객서비스포털을 통해 서비스를 요청하시기 바랍니다.

여러분의 의견에 항상 감사드립니다.

1.Altibase SSL/TLS 소개

이 장은 Altibase SSL/TLS의 개념 및 특징에 대해 설명한다.

SSL/TLS란

이 절은 Altibase SSL/TLS의 기본 개념을 설명한다.

SSL/TLS 프로토콜 소개

보안 소켓 계층(Secure Socket Layer, 이하 SSL)은 넷스케이프사에 의해 개발된 표준 네트워크 보안 프로토콜이다.

SSL1.0은 공개된 적이 없고, 2.0버전이 릴리즈 되었지만 많은 보안 결함이 있어서 1996년에 완전히 새로운 3.0 버전으로 릴리즈하였다. 그 후 IETF(Internet Engineering Task Force)는 RFC 2246의 SSL 3.0에 기반한 보안 프로토콜인 TLS 1.0을 발표하였다. TLS 프로토콜에 대한 자세한 사항은 'RFC-2246 The TLS Protocol(ftp://ftp.rfc-editor.org/in-notes/rfc2246.txt)'을 참조하기 바란다.

암호화와 인증서

대칭키 알고리즘은 송신측과 수신측간의 동일한 비밀키를 공유함으로써 통신 보안을 가능하게 한다. 이 비밀키는 효과적인 알고리즘으로 평문을 암호화하고, 암호문을 복호화하는데 사용된다. 그러나 송신자와 수신자간의 통신시 비밀키를 안전하게 공유하는데에는 한계가 있다.

이러한 문제를 해결하기 위해 비대칭키(또는 공개키) 암호화 알고리즘이 등장한다. 비대칭키 암호화 알고리즘은 공개키와 개인키를 한 쌍으로 하여 암호화와 복호화를 할 수 있다. 누구나 자유롭게 얻을 수 있는 공개키를 사용해 암호화된 메세지는 공개키와 매칭되는 개인키를 가진 사람만 복호화를 할 수 있도록 하며, 개인키로 암호화된 메세지 역시 공개키로 복호화할 수 있다.

즉 대칭키 알고리즘에서 사용하던 비밀키 대신 비대칭키 알고리즘을 사용함으로써, 비밀키를 상대방의 공개키로 암호화하여 개인키를 가지고 있는 상대방에게 보낸다. 그러면 개인키를 갖고 있는 상대방은 암호화된 비밀키를 자신의 개인키를 이용하여 비밀키를 복호화할 수 있다. 그 후에는 비밀키를 이용하여 서로 메세지를 암호화 및 복호화하여 통신을 할 수 있다.

그럼에도 불구하고, 중간공격자(Man-In-The-Middle attacker, MITM attacker)가 서버와 클라이언트를 가장하여, 통신 보안을 위협할 수 있다. 즉 클라이언트의 공개키 요청을 중간에서 가로채서 자신의 공개키를 대신 보내는 경우가 발생할 수 있다.

따라서 제 3의 신뢰할 수 있는 공식 인증 기관(Certificate Authority, 이하 CA)은 공개키 인증서(이하 인증서)를 발급하여 공개키의 소유권을 확인한다. 예를 들어, WWW(World Wide Web)과 같은 개방된 시스템 환경에서는 통신 보안을 위해 공개 인증서를 사용하는 것은 중요하다. 반면에 개인 인증서는 비공식 인증 기관(Non-official CA)에서 서명하고 발급한 인증서를 말한다. 개인 인증서는 자유롭게 사용하고, 필요할 때 마다 생성할 수 있기 때문에 폐쇄된 시스템 환경에서 유용하다.

Altibase의 통신 보안

Altibase는 데이터를 암호화 및 복호화하기 위하여 대칭키 알고리즘과 인증을 위한 공개키(public key)/개인 키(private key)의 한 쌍을 안전하게 교환하기 위한 비대칭 알고리즘을 이용하는 SSL/TLS(이하 SSL)를 채택하고 있다.

따라서 클라이언트가 SSL 통신으로 Altibase에 접속할 때에는 아래의 순서로 SSL 핸드쉐이크를 수행한다.

1. 클라이언트와 서버는 보안이 된 연결을 위해 SSL 프로토콜 버전, 데이터를 주고 받을 때 사용할 암호화 알고리즘 등과 같은 정보를 교환한다.
2. 만약 클라이언트에게 서버 인증이 필요한 경우, 서버는 클라이언트에게 자신의 인증서를 전송하고 클라이언트는 전송받은 인증서로 서버를 확인한다.
3. 서버가 클라이언트의 인증을 요청하면 클라이언트는 서버에게 자신의 인증서를 보낸다. 그리고 서버는 클라이언트에게서 받은 인증서로 클라이언트를 확인한다.
4. 클라이언트와 서버는 SSL 통신에 필요한 정보를 비대칭키 알고리즘으로 교환하고, 데이터의 암복호화와 무결성 검증을 위해 사용될 세션키를 생성한다.
5. 클라이언트와 서버는 교환될 메시지가 세션키로 암호화되고 핸드쉐이크가 완료되었는지 여부를 서로 통신한다.

Altibase의SSL 특징

SSL 통신으로 Altibase를 사용할 때 다음과 같은 특징이 있다.

- Altibase 는 OpenSSL 라이브러리에서 지원하는 TLS 1.0 프로토콜을 사용한다.
- Altibase는 서버 전용 인증과 상호 인증을 지원한다.
  -서버 전용 인증 : 서버만 인증서를 가지고 있고, 클라이언트에게 인증서를 제공한다.
  -상호 인증 : 클라이언트와 서버 둘 다 인증서가 필요하며, 서로 상대방의 인증서를 확인한다.
- Altibase 서버는 SSL 통신을 위해 기존에 사용하던 통신 포트 외에 별도의 포트를 사용하여 서비스한다(예를 들어, HTTPS를 위한 443 port). Altibase는 TCP 접속을 할 때 비보안에서 보안으로 접속 전환이 허용되지 않기 때문이다.
- 보안 연결을 사용하려면 모든 클라이언트 애플리케이션이 Java 1.5 이상에 포함된 Java Secure Socket Extension(JSSE) API가 필요하다. JSSE는 데이터 암호화, 서버 인증, 메시지 무결성과 선택적 클라이언트 인증 뿐만 아니라 SSL2.0, 3.0과 TLS 1.0 프로토콜 구현을 제공한다.
- Altibase는 SSL 통신을 위해 JDBC와 ODBC를 통한 인터페이스를 제공하며, 현재는 인텔 계열의 리눅스에서만 사용할 수 있다.

2.SSL 설치 및 시작

이 장은 Altibase에서 SSL 통신을 사용하기 위한 소프트웨어 요구사항 및 설치하는 방법에 대하여 설명한다.

소프트웨어 요구사항

이 절은 서버와 클라이언트에서 SSL 통신을 사용하기 위한 요구사항을 설명한다.

서버(Server)

- OpenSSL 툴킷 0.9.4~1.0.2
- Altibase 6.5.1 이상 (인텔 계열 리눅스만 지원)

OpenSSL 툴킷은 Altibase에서 SSL/TLS 프로토콜을 이용한 SSL 통신을 사용하기 위해 필요하다. OpenSSL 툴킷은 OpenSSL 프로젝트로 개발되었으며 OpenSSL 홈페이지(http://www.openssl.org/source)에서 다운로드할 수 있다. 단 설치된 OpenSSL 버전이 허트블리드 버그(Heartbleed Bug)에 감염되지 않았는지 확인하고 조치해야 한다.

따라서 OpenSSL 툴킷을 사용하기 전에 감염된 여부를 확인해야 한다.
OPENSSL_NO_HEARTBEATS 옵션을 사용해 재컴파일하면 된다.

클라이언트(Client)

ODBC

ODBC에서 SSL 통신을 사용하기 위해서 반드시 OpenSSL 툴킷이 설치되어야 한다.

JDBC

SSL을 통하여 클라이언트 자바 애플리케이션을 자유롭게 실행하기 위해서는JRE 1.6 이상을 쓸 것을 권장한다. JRE 1.6 이상이 권장되는 이유는 다음과 같다(JRE 1.5도 사용은 가능하다).

- JRE 1.6 이상만 클라이언트 인증서를 keystore로 내보내는 것을 지원한다 (그러나 인증서를 보내는 것이 항상 요구되는 것은 아니다).

SSL 사용을 위한 환경 설정

이 절은 Altibase에 SSL 통신을 사용하기 위한 환경 설정에 대해 설명한다.

- 서버에서 SSL 사용을 위한 환경 설정
- JDBC에서 SSL 사용을 위한 환경 설정
- ODBC에서 SSL 사용을 위한 환경 설정

서버 환경 설정

- Step 1: OpenSSL 설치 및 라이브러리 확인
- Step 2: 서버 프로퍼티 설정
- Step 3: SSL 클라이언트 인증 지정

- Step 4: 서버 인증서, 개인키, 인증 기관 설정
- Step 5: 서버 시작

Step 1:OpenSSL 설치 및 라이브러리 확인

SSL이 활성화된 Altibase를 설치하기 전에 OpenSSL 툴킷을 설치하는 것이 좋다. 만약 OpenSSL 툴킷이 설치되지 않았는데 Altibase의 SSL을 사용하면,Altibase는 OpenSSL 라이브러리를 찾을 수 없다는 경고 메세지가 나타난다.

서버에 OpenSSL이 설치되었다는 것을 확인하고, 허트블리드 버그에 감염되지 않았는지 확인한다. 필요할 경우 OS에서 제공하는 패키지 매니저(e.g., RPM, Red Hat Linux)로 설치하거나 http://www.openssl.org/source 에서 직접 다운로드받아서 컴파일한다.

설치가 끝나면, 아래와 같이 설치된 OpenSSL 버전을 확인한다.

```
$ openssl version
OpenSSL 0.9.7e-fips-rhel5 01 Jul 2008
```

Step 2: 서버 프로퍼티 설정

Altibase에서 SSL로 접속하여 사용하려면 다음의 프로퍼티를 설정하여야 한다. SSL 접속을 위한 관련 프로퍼티들은 $ALTIBASE_HOME/conf에 있다. 프로퍼티에 대한 자세한 설명은 *General Reference*를 참조한다.

- ***SSL_ENABLE***

  SSL_ENABLE은 Altibase에서 SSL을 활성화/비활성화 한다. SSL을 사용하려면, 이 프로퍼티의 값은 1이어야 한다.

- ***SSL_PORT_NO***

  SSL_PORT_NO는 SSL 접속이 사용할 포트 번호를 지정한다. 이 값은 시스템에서 유일해야 한다.

- ***SSL_MAX_LISTEN***

  SSL 로 접속하여 동시에 Altibase에 접속할 수 있는 대기 큐의 최대 크기를 설정한다. 이 값이 클수록 Altibase에 더 많은 메모리가 필요하다.

- ***SSL_CIPHER_LIST***

  이 값은 서버와 클라이언트가 협의하고 사용할 수 있는 암호 알고리즘의 이름 목록이다. 사용자의 보안 정책에 따라 한 개 이상의 암호를 사용할 수 있으며, 암호는 콜론 (:)으로 구분한다. 사용자가 사용할 수 있는 암호 목록은 OpenSSL(http://www.openssl.org/)에서 확인하거나 아래처럼 명령어를 사용하여 확인할 수 있다.

```
$ openssl ciphers
```

Step 3: SSL 클라이언트 인증 설정

- ***SSL_CLIENT_AUTHENTICATION***

  SSL 인증 모드를 서버 전용 인증 또는 상호 인증으로 설정할 수 있다. 이 값을 0으로 하면 서버만 인증하고, 1로 하면 서버-클라이언트를 상호 인증한다.

Step 4: 인증서, 개인키, 인증 기관(CA) 설정

- SSL_CERT
- SSL_KEY
- SSL_CA
- SSL_CAPATH

SSL_CERT

SSL_CERT 값은 서버의 인증서가 위치한 경로를 지정한다.

예를 들어 서버의 인증서 이름이 cert/server-cert.pem이고, $ALTIBASE_HOME/cert에 위치한 경우, 이 프로퍼티의 값은 $ALTIBASE_HOME/cert/server-cert.pem이다.

SSL_KEY

서버의 개인키(private key)가 위치하는 경로를 지정한다.

예를 들어 서버의 개인 키 이름이 server-key.pem이고, $ALTIBASE_HOME/cert에 위치한 경우, 이 프로퍼티의 값은 $ALTIBASE_HOME/cert/server-key.pem이다.

SSL_CA, SSL_CAPATH

공인 기관에서 발급받은 CA 인증서의 소유권을 인정받기 위해 SSL_CA 또는 SSL_CAPATH 프로퍼티가 설정되어야 한다. CA 인증서 파일은 사용자 지정 경로나 X.509 형식의 디렉토리에 위치한다

만약 공인인증서가 없다면, 개인인증서를 생성하여 SSL을 사용할 수 있다.

Step 5: 서버 시작

SSL 통신이 가능한 Altibase를 시작한다. Altibase는 서버와 클라이언트에서 SSL을 사용하기 위하여 샘플 파일을 부록에 소개하고 있다.

SSL_ENABLE 프로퍼티의 값이 1(SSL사용)로 설정되었다면, SSL을 이용하기 위한 대기 포트가 아래와 같이 나타난다. 즉 환경변수로 ALTIBASE_PORT_NO와 ALTIBASE_SSL_PORT_NO 둘 다 정의되어야 한다.

```
$server start
------------------------------------------------------------------
     Altibase Client Query utility.
     Release Version 7.1.0.0.1
     Copyright 2000, Altibase Corporation or its subsidiaries.
     All Rights Reserved.
------------------------------------------------------------------
ISQL_CONNECTION = UNIX, SERVER = localhost
[ERR-910FB : Connected to idle instance]
Connecting to the DB server.... Connected.
...
TRANSITION TO PHASE : SERVICE
  [CM] Listener started : TCP on port 20300 [IPV4]
  [CM] Listener started : SSL on port 20443 [IPV4]
  ...
--- STARTUP Process SUCCESS ---
Command executed successfully.
```

JDBC 환경 설정

- Step 1: 인증서 가져오기
- Step 2: 인증을 위한 환경 설정
- Step 3: SSL을 위한 JDBC 프로퍼티 설정
- Step 4: Altibase 환경변수 설정

Step 1: 인증서 가져오기

서버의 CA 인증을 위해 인증서를 Trustore에 가져오거나, CA 인증과 암호키를 얻기 위해 Keystore에 인증서를 가져온다. JSSE는 인증을 위해 Truststore와 Keystore를 사용한다.

인증서를 가져오기 위해 수행할 작업은 공개 인증서와 개인인증서의 타입에 따라 다르며, 그 후 작업은 선택된 인증 모드에 따라 다르다.

1. *서버 전용 모드의 개인 인증서*
   Truststore에 서버의 CA 인증서를 가져온다.
2. *상호 인증 모드의 개인 인증서*
   Truststore에 서버의 CA 인증서를 가져온다. 그리고 Keysotre에 CA 인증과 비밀키를 포함하고 있는 PKCS #12 파일을 가져온다.
3. *서버 전용 모드의 공개 인증서*
   작업 내역이 없다.
4. *상호 인증 모드의 공개 인증서*
   Keysotre에 CA 인증과 비밀키를 포함하고 있는 PKCS #12 파일을 가져온다.

1번, 2번의 경우처럼 개인 인증서를 사용할 때, 아래와 같이 CA 인증서를 trustore에 가져온다.

```
$keytool -import -alias alias_name -file server_certificate_file.pem -keystore truststore -storepass password
```

2번과 4번의 경우, 상호 인증 모드를 위하여 우선 CA 인증과 비밀 키가 필요하다. 인증서와 비밀 키를 가져오기 전에 가지고 있는 인증서와 비밀 키가 PKCS #12 형식의 파일에 있는지, 자바 버전이 1.6 이상인지 확인한다.

PKCS #12 파일이 없다면, 클라이언트의 인증서와 비밀 키를 가진 PKCS #12 파일을 생성하기 위해 아래와 같이 pkcs12 옵션을 사용하여 OpenSSL을 실행한다.

```
$openssl pkcs12 -export -in client_certificate.pem -inkey client_secretkey_file.pem > pkcs_file.p12
```

PKCS #12 파일이 있다면, 아래와 같이 Keystore에 PKCS #12 파일을 가져온다. [1]

[1] Java 6 이상의 버전에서만 '-importkeysotre' 옵션을 사용하여 Keystore에 pem형태의 파일을 가져올 수 있다.

```
$keytool -importkeystore -srckeystore pkcs_file.p12 -destkeystore keystore.jks -srcstoretype pkcs12
```

Step 2: 인증을 위한 환경 설정

자바 애플리케이션에서 SSL 인증 환경을 설정하기 위해, JRE는 Truststore와 Keystore에 접근해야 한다. 사용자는 아래의 3가지 방식중 하나를 선택하여 인증

환경을 설정할 수 있다.

- 자바 애플리케이션에서 명령어 사용
- 자바 애플리케이션에서 System.set Property 설정
- 자바 애플리케이션에서 JDBC 프로퍼티 설정

자바 애플리케이션에서 명령어 사용

```
-Djavax.net.ssl.keyStore=path_to_keystore
-Djavax.net.ssl.keyStorePassword=password
-Djavax.net.ssl.trustStore=path_to_truststore
-Djavax.net.ssl.trustStorePassword=password
```

자바 애플리케이션에서 System.set Property 설정

```
System.setProperty("javax.net.ssl.keyStore", "path_to_keystore");
System.setProperty("javax.net.ssl.keyStorePassword", "password");
System.setProperty("javax.net.ssl.trustStore", "path_to_truststore");
System.setProperty("javax.net.ssl.trustStorePassword", "password");
```

자바 애플리케이션에서 JDBC 프로퍼티 설정

```
Properties sProp = new Properties();
sProps.put("keystore_url", "path_to_keystore");
sProps.put("keystore_password", "password");
sProps.put("truststore_url", "path_to_truststore");
sProps.put("truststore_password", "password");
```

Step 3: SSL을 위한 JDBC 프로퍼티 설정

Altibase는 SSL을 사용하기 위해 SSL 연결을 위한 JDBC를 제공한다. SSL을 위한 JDBC 프로퍼티는 아래와 같이 구분된다.

- SSL 연결을 위한 JDBC 프로퍼티
- 인증을 위한 JDBC 프로퍼티

SSL 연결을 위한 JDBC 프로퍼티

이름	설명	값의 범위	기본값
ssl_enable	서버에 SSL 통신을 사용해서 접속할지 여부를 설정한다. 이 값이 true일 경우 SSL 커넥션을 생성하고, false일 경우 TCP 커넥션을 생성한다.	[true | false]	false
port	접속을 시도할 대상 서버의 포트번호를 지정한다. SSL 포트 번호가 적용되는 우선순위는 아래와 같다. ssl_enable이 true일 경우 port 값이 지정되었다면 우선 적용되고, 지정되지 않은 경우 ALTIBASE_SSL_PORT_NO 환경 변수의 값을 따른다. 하지만 ALTIBASE_SSL_PORT_NO의 값도 지정되지 않은 경우, 기본값 20300이 적용된다.	0 ~ 65535	ssl_enable(false): 20300 ssl_enable(ture): 20443

이름	설명	값의 범위	기본값
ciphersuite_list	사용할 암호 알고리즘 목록이다. 암호명은 콜론(:)으로 구분된다. *SSL_RSA_WITH_RC4_128_MD5:SSL_RSA_WITH_RC4_128_SHA.* 만약 JRE가 이 알고리즘을 지원하지 않는다면, IllegalArgumentException이 나타난다.	String	JRE가 지원하는 모든 cipher suite list

인증을 위한 JDBC 프로퍼티

이름	설명	값의 범위	기본값
verify_server_certificate	대상 서버의 CA 인증서를 인증할지 여부를 설정한다. 이 값을 FALSE로 설정하면, 클라이언트의 애플리케이션은 서버의 CA 인증서를 인증하지 않고, SSL exception이 발생한다. 그러나 서버의 개인인증서를 가져올 필요는 없다.	[true | false]	true
keystore_url	KeyStore의 경로를 지정한다. KeyStore는 개인 인증서와 공개 인증서를 가지고 있다.	String	N/A
keystore_type	keystore_url의 keystore 타입을 설정한다. Java는 기본적으로 Java Key Store(JKS) 타입으로 처리된다.	[JKS, JCEKS, PKCS12 등]	JKS
keystore_password	keystore_url에 비밀번호를 지정한다.	String	N/A
truststore_url	TrustStore의 경로를 지정한다. TrustStore는 CA의 인증서를 갖고있는 KeyStore이다.	String	N/A
truststore_type	truststore_url의 TrustStore 타입을 설정한다. Java는 기본적으로 Java Key Store(JKS) 타입으로 처리된다.	[JKS, JCEKS, PKCS12 등]	JKS
truststore_password	truststore_url의 비밀번호를 지정할 수 있다.	String	N/A

Step 4: Altibase 환경변수 설정

Altibase가 SSL 통신을 하기 위한 포트(port) 번호를 설정하는 단계이며, 생략할 수 있다.

JDBC에서 port 프로퍼티를 설정하지 않았을 때, ALTIBASE_SSL_PORT_NO에 설정된 값이 포트번호로 사용된다. 만약 ALTIBASE_SSL_PORT_NO가 생략되었다면, port 프로퍼티의 기본 값이 사용된다.

Name	Description	Value	Default Value
ALTIBASE_SSL_PORT_NO	접속할 서버의 SSL 포트 번호를 설정한다.	Range: 1024 ~ 65535	N/A

JDBC에서 SSL 사용시 고려사항

JDBC에서 SSL을 사용할 때 아래의 사항을 고려해서 사용한다

KeyStore에 PKCS #12 가져오기(JRE 1.6 이상)

SSL을 통한 상호 인증을 사용하기 위해서는 우선 클라이언트의 CA 인증서와 개인 키를 KeyStore로 가져와야 한다. 이 때 지원되는 버전은 JRE1.6이상이다. 자바 6 이상에서만 importkeystore 옵션을 사용해서 pem 형태의 파일을 keystore로 보낼 수 있기 때문이다.

그러나 JRE1.6이상에서 PKCS를 한 번만 import한 후에는 JRE1.5를 사용할 수 있다. 즉 import 과정만 JRE1.6이 필요한것이지 상호인증 기능 자체는 1.5에서도 동작한다.

```
$keytool -importkeystore -srckeystore pkcs_file.p12 -destkeystore keystore.jks
-srcstoretype pkcs12
```

ODBC 환경 설정

- Step 1: OpenSSL 라이브러리 확인
- Step 2: 클라이언트 인증서 준비
- Step 3: SSL을 위한 ODBC 프로퍼티 설정
- Step 4: 클라이언트 프로그램 작성

Step 1: OpenSSL 라이브러리 확인

ODBC에서 SSL 통신을 사용하기 위해 OpenSSL 라이브러리가 필요하다. ODBC는 서버와 연결하면서, OpenSSL 라이브러리를 읽어 필요한 함수들을 호출한다.

따라서 클라이언트 애플리케이션을 작성하기 전에 OpenSSL 라이브러리가 제대로 설치되었는지 확인해야 한다. 라이브러리가 설치되는 위치는 운영시스템에 따라 달라질 수 있다.

- 라이브러리 설치 확인

```
$ ls -al /usr/lib/libssl*
$ ls -al /usr/lib/libcrypto*
```

- 유틸리티 설치 확인

```
$ openssl version
```

Step 2: 클라이언트 인증서 준비

서버와 클라이언트의 상호 인증을 위해 PEM 형태로 된 클라이언트의 인증서와 개인키를 준비한다. 해당 파일들은 ODBC를 이용하는 클라이언트가 접근할 수 있는 위치에 있어야 한다.

Step 3: SSL을 위한 ODBC/CLI 프로퍼티 설정

SSL 통신을 사용하려는 클라이언트 프로그램을 작성하기 전에 SSL 관련 프로퍼티들을 설정해야 한다. 클라이언트는 서버 접속시 다음의 프로퍼티들을 연결 문자열(connection string)로 지정할 수 있다. 사용 방법은 샘플 프로그램을 참조한다.

SSL 접속을 위한 관련 프로퍼티들은 $ALTIBASE_HOME/conf에 있다.

이름	설명	값의 범위	기본값
SSL_CA	CA 인증서의 소유권을 인정받기 위해서 CA 인증서를 저장하는 파일의 경로를 지정할 수 있다. CA 인증서 파일은 사용자 지정 경로나 X.509 형식의 디렉토리에 위치한다. 예) SSL_CA=/cert/ca-cert.pem	없음	NULL
SSL_CAPATH	CA 디렉토리 형식의 CAPATH를 지정할 수 있다. 예) SSL_CAPATH=/etc/ssl/certs	없음	NULL
SSL_CERT	클라이언트의 인증서 파일 경로를 지정한다. 예) SSL_CERT=/cert/client-cert.pem	없음	NULL
SSL_KEY	클라이언트의 개인 키(private key)가 저장된 파일 경로를 지정한다. 예) SSL_KEY=/cert/client-key.pem	없음	NULL
SSL_VERIFY	Altibase 서버의 인증서를 검증할지 여부를 설정한다. 만약 검증에 실패하면 SSL Handshake 는 실패하고, SSL 통신은 더 이상 진행되지 않는다. 0(OFF): 서버의 인증서를 검증하지 않는다. 1(ON): 서버의 인증서를 검증한다. 예) SSL_VERIFY=0	0: OFF 1: ON	0 (off)
SSL_CIPHER	이 프로퍼티는 클라이언트가 서버와 협의하여 사용할 수 있는 암호 알고리즘들이다. 암호 알고리즘은 사용자의 보안 정책에 따라 하나 또는 그 이상의 암호를 사용할 수 있다. 한 개 이상의 암호를 사용할 경우 콜론(:)으로 구분한다. 사용자가 사용할 수 있는 암호 목록은 OpenSSL(http://www.openssl.org/)에서 확인하거나 다음과 같이 명령어를 사용하여 확인할 수 있다. $ openssl ciphers 예) SSL_CIPHER=EDH-DSS-DES-CBC-SHA:DES-CBC-SHA	없음	NULL

다음은 서버의 SSL 프로퍼티와 ODBC/CLI용 프로퍼티를 비교한 표이다.

이름	서버 (altibase.properties)	ODBC/CLI
SSL_ENABLE	O	X 클라이언트는 CONNTYPE=SSL 일 경우 SSL_ENABLE=1과 동일한 의미이다.
SSL_PORT_NO	O	X 클라이언트에서는 별도의 SSL용 포트 없이 CONNTYPE=SSL;PORT=20443 등으로 접속한다.
SSL_MAX_LISTEN	O	X
SSL_CLIENT_AUTHENTICATION	O	X
SSL_CIPHER_LIST	O	X
SSL_CA	O	O
SSL_CAPATH	O	O
SSL_CERT	O	O
SSL_KEY	O	O

이름	서버 (altibase.properties)	ODBC/CLI
SSL_CIPHER	X	O
SSL_VERIFY	X	O

Step 4: 클라이언트 프로그램 작성

클라이언트 애플리케이션에서 SSL 통신을 사용하기 위해 프로그램을 작성한다.
Altibase 디렉토리에 SSL을 사용하는 샘플 프로그램을 확인할 수 있다.
$ALTIBASE_HOME/sample/SQLCLI/SSL 참고한다.

3.SSL 연결 관리

SSL 관리

이 절은 SSL 기능을 사용할 때 아래의 방법으로 모니터링을 하고, 사용자에 대한 통신 보안을 강화하는 등의 관리 방법을 설명한다.

- TCP 접속 제한
- SSL 모니텅링 및 관리

TCP 접속 제한

서버와 클라이언트간의 안전한 통신을 위해 사용자의 TCP 접속을 제한하여 보안을 강화할 수 있다.

사용자에 대하여 TCP 접속을 해제하지 않으면, 원격의 사용자는 TCP는 물론 SSL을 통해서도 데이터베이스에 접근할 수 있다. 그러나 SYS 사용자가 원격 사용자에 대해 SSL 접속을 통해서만 데이터베이스에 접근하려면, TCP 접속 기능을 해제할 수 있다.

- 사용자의 TCP 접속 제한

```
ALTER USER user_name [ENABLE | DISABLE] TCP
CREATE USER user_name IDENTIFIED BY password [ENABLE | DISABLE] TCP
```

데이터베이스 사용자들에 대한 TCP 접속이 허용되었는지 여부는 SYSTEM_.SYS_USERS_ 메타 테이블의 DISABLE_TCP 칼럼에 SELECT 쿼리를 수행하여 확인할 수 있다.

```
SELECT * FROM SYSTEM_.SYS_USERS_
```

예제

<질의> TCP 접속이 허가되지 않는 사용자를 생성한다.

```
iSQL> CREATE USER user1 IDENTIFIED BY user1 DISABLE TCP;
Create success.
```

<질의> 생성된 사용자가 TCP 접속이 제한되었는지 확인한다.

```
iSQL> SELECT user_name, disable_tcp FROM SYSTEM_.SYS_USERS_;
USER_NAME                                DISABLE_TCP
-----------------------------------------------------------
SYSTEM_                                  F
SYS                                      F
USER1                                    T
3 rows selected.
```

<질의> TCP 접속이 허가되지 않은 사용자가 TCP로 접속할 수 있도록 변경한 후, 메타 테이블을 확인하여 접속 권한이 변경되었는지 확인한다.

```
iSQL> ALTER USER user1 ENABLE TCP;
Alter success.
iSQL> SELECT user_name, disable_tcp FROM SYSTEM_.SYS_USERS_;
USER_NAME                                DISABLE_TCP
-----------------------------------------------------------
SYSTEM_                                  F
SYS                                      F
USER1                                    F
3 rows selected.
```

SSL 모니터링 및 관리

수행중인 데이터베이스를 모니터링하면 데이터베이스에 대한 불법 접근을 감지하는데 효과적이다.

V$SESSION 성능 뷰는 연결된 클라이언트와 접속 프로퍼티 등의 데이터베이스 접속에 대한 정보를 확인할 수 있다. 특히 COMM_NAME 칼럼은 프로토콜 타입, 클라이언트 주소와 연결된 포트 넘버를 보여준다. 성능 뷰에 대한 자세한 설명은 General Reference 매뉴얼을 참조하기 바란다.

```
iSQL> SELECT id, db_username, comm_name  FROM V$SESSION WHERE comm_name like 'SSL%';
ID          DB_USERNAME
-----------------------------------------------------------
COMM_NAME
-------------------------------------------------------------------
1           USER1
SSL 127.0.0.1:40328
1 row selected.
```

데이터베이스 서버에 대한 불법 접근을 감지하면, 관리자는 다음과 같이 강제 종료할 수 있다.

- SYS 사용자가 SYSDBA 모드로 로그인하여, 불법 접속한 세션을 종료한다.

```
$isql -s localhost -u user_name -p password -SYSDBA
iSQL> ALTER DATABASE database_name SESSION CLOSE session_number
```

- 세션을 종료한 후에는 모니터링하여 대상 세션이 종료되었는지 확인한다.

```
iSQL> SELECT * FROM V$SESSION WHERE ID = session_id
```

SYSDBA 모드는 SYS 사용자가 관리자 역할을 수행할 수 있는 권한이다. SYS 계정으로 SYSDBA 모드로 데이터베이스에 로그인하려면 iSQL에서 iSQL 사용시 -SYSDBA 옵션을 사용하거나 CONNECT 명령을 AS SYSDBA 옵션과 함께 사용한다.

```
$ isql -s localhost -u sys -p manager -sysdba
--------------------------------------------------------------------
     Altibase Client Query utility.
     Release Version 7.1.0.0.1
     Copyright 2000, Altibase Corporation or its subsidiaries.
     All Rights Reserved.
--------------------------------------------------------------------
ISQL_CONNECTION = UNIX, SERVER = localhost
iSQL(sysdba)> SELECT id, db_username, comm_name FROM V$SESSION WHERE comm_name like 'SSL%';
ID          DB_USERNAME
-----------------------------------------------------------
COMM_NAME
-------------------------------------------------------------------------
1           USER1
SSL 127.0.0.1:40328
1 row selected.
iSQL(sysdba)> ALTER DATABASE mydb SESSION CLOSE 1;
Alter success.
iSQL(sysdba)> SELECT id, db_username, comm_name  FROM V$SESSION WHERE comm_name like 'SSL%';
ID          DB_USERNAME
-----------------------------------------------------------
COMM_NAME
-------------------------------------------------------------------------
No rows selected.
```

A.부록: SSL Sample

Altibase는 서버와 클라이언트에서 SSL을 사용하기 위하여 샘플 파일을 제공한다.

$ALTIBASE_HOME/sample/cert 에는 서명된 CA 인증서, 두 쌍의 인증서, 개인 키, 샘플 자바 소스 코드 등이 제공되고 있다.

Altibase 디렉토리에 SSL을 사용하는 샘플 프로그램은 . $ALTIBASE_HOME/sample/ 에서 확인할 수 있다.

JDBC를 이용한 SSL 통신 샘플

CLI 샘플 프로그램은 $ALTIBASE_HOME/sample/SQLCLI/SSL을 참고한다.

```
import java.util.Properties;
import java.sql.*;

import Altibase.jdbc.driver.util.RuntimeEnvironmentVariables;

class SslSimpleSQL
{
    public static void main(String args[])
    {
        Properties        sProps   = new Properties();
        Connection        sCon     = null;
        Statement         sStmt    = null;
        PreparedStatement sPreStmt = null;
        ResultSet         sRS;

        if ( args.length == 0 )
        {
           System.err.println("Usage : java class_name port_no");
           System.exit(-1);
        }

        String     sDevhome = RuntimeEnvironmentVariables.getVariable("ALTIBASE_HOME");
        String     sTrusStore = sDevhome + "/sample/CERT/truststore";
        String     sKeyStore = sDevhome + "/sample/CERT/keystore.jks";

        // Keystore
        System.setProperty("javax.net.ssl.keyStore", sKeyStore);
        System.setProperty("javax.net.ssl.keyStorePassword", "altibase");

        // Truststore
        System.setProperty("javax.net.ssl.trustStore", sTrusStore );
        System.setProperty("javax.net.ssl.trustStorePassword", "altibase");

        String sPort     = args[0];
        String sURL      = "jdbc:Altibase://127.0.0.1/mydb";
        String sUser     = "SYS";
        String sPassword = "MANAGER";

        sProps.put( "user",     sUser);
        sProps.put( "password", sPassword);
        sProps.put( "ssl_enable" , "true");
        sProps.put( "ssl_port", sPort);

        // sProps.put( "encoding", sEncoding );

        /* Deploy Altibase's JDBC Driver  */
        try
        {
            Class.forName("Altibase.jdbc.driver.AltibaseDriver");
        }
        catch ( Exception e )
        {
            System.out.println("Can't register Altibase Driver");
            System.out.println( "ERROR MESSAGE : " + e.getMessage() );
            System.exit(-1);
        }

        /* Initialize environment */
        try
        {
```

```
    sCon = DriverManager.getConnection( sURL, sProps );
    sStmt = sCon.createStatement();
}
catch ( Exception e )
{
    System.out.println( "ERROR MESSAGE : " + e.getMessage() );
    e.printStackTrace();
}

try
{
    sStmt.execute( "DROP TABLE TEST_EMP_TBL" );
}
catch ( SQLException e )
{
}

try
{
   sStmt.execute( "CREATE TABLE TEST_EMP_TBL " +
                  "( EMP_FIRST VARCHAR(20), " +
                  "EMP_LAST VARCHAR(20), " +
                  "EMP_NO INTEGER )" );

    sPreStmt = sCon.prepareStatement( "INSERT INTO TEST_EMP_TBL " +
                                      "VALUES( ?, ?, ? )" );

    sPreStmt.setString( 1, "Susan" );
    sPreStmt.setString( 2, "Davenport" );
    sPreStmt.setInt(    3, 2 );
    sPreStmt.execute();

    sPreStmt.setString( 1, "Ken" );
    sPreStmt.setString( 2, "Kobain" );
    sPreStmt.setInt(    3, 3 );
    sPreStmt.execute();

    sPreStmt.setString( 1, "Aaron" );
    sPreStmt.setString( 2, "Foster" );
    sPreStmt.setInt(    3, 4 );
    sPreStmt.execute();

    sPreStmt.setString( 1, "Farhad" );
    sPreStmt.setString( 2, "Ghorbani" );
    sPreStmt.setInt(    3, 5 );
    sPreStmt.execute();

    sPreStmt.setString( 1, "Ryu" );
    sPreStmt.setString( 2, "Momoi" );
    sPreStmt.setInt(    3, 6 );
    sPreStmt.execute();

    sRS = sStmt.executeQuery( "SELECT EMP_FIRST, EMP_LAST," +
                              " EMP_NO FROM TEST_EMP_TBL " );

    /* Fetch all data */
    while( sRS.next() )
    {
        System.out.println( "  EmpName : " + sRS.getString(1) +
                            " " + sRS.getString(2) );
        System.out.println( "  EmpNO   : " + sRS.getInt(3) );
```

```
            }

            /* Finalize process */
            sStmt.close();
            sPreStmt.close();
            sCon.close();
        }
        catch ( SQLException e )
        {
            System.out.println( "ERROR CODE    : " + e.getErrorCode() );
            System.out.println( "ERROR MESSAGE : " + e.getMessage() );
            e.printStackTrace();
        }
    }
}
```