ネットワークモニタ用
準リアルタイム情報システムの設計と実装

s97490-M
森下 旭
計算機科学講座
田中研究室

平成 14 年 3 月 22 日

目 次

第1章　序論

1.1　はじめに

ネットワークの管理・運用にあたり、ネットワーク間でどのような通信が行われたか、という記録は重要なものである。

そこで、本研究では、Web サーバとクライアントがデータを送受信するのに使われるプロトコルである http についてのログを、プロキシサーバーを用いる方法と、ルータを用いる方法によって収集し、そのログをデータベースに定時に格納することにより、誰が、いつ、などといった要求に対して検索を行うことができるシステムの構築を試みた。

1.2　本システムの構成

本システムを構成するものとして次のものがある。

ネットワークモニタ

本研究ではプロキシサーバとルータでログを収集することでネットワークモニタとした。ログの収集の手段としては、この二つ以外の方法として、以下のものが考えられる。

・各ホストのログを集める

・アナライザを常時稼動させログを収集する

しかし、各ホストのログを集める場合には手間がかかり、アルゴリズムも複雑になるという点、アナライザを常時稼動させる場合には一つ一つのプロセスのログを収集するとマシンに負荷がかかるし、ログの量が大きくなってしまう、といった問題点がそれぞれにある。これらのことから本研究ではプロキシサーバを用いる方法と、ルータを用いる方法の二通りの方法でログを収集することとした。

データベースへの格納

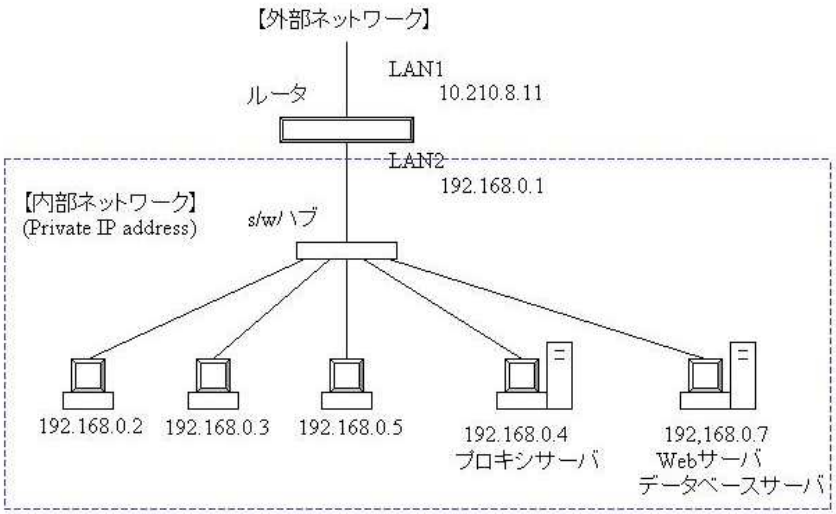
収集されたログは、プロキシサーバ側で整理し、国際標準データベース言語 SQL を用いて別ホストのデータベースサーバへ格納する。このことにより、ログ消去などの外部からの攻撃に対して、別ホストのサーバを用いること、DBMS のセキュリティ機能を用いること、の二重の保護がえられる。

定時起動ルーチン

本システムは１時間ごとに自動的に、新しく収集されたログを整理し、データベースへの格納を行う。自動化することにより手間が省け、また、１時間ごとの準リアルタイムで行うことでデータベースに格納されているデータは常に新しい情報となる。

1.2.1　ネットワーク構成

研究室のネットワークを以下に示す。



第2章　ログの収集

2.1　プロキシサーバを用いてのログの収集

プロキシサーバとは代理サーバのことで、内部のネットワークのホストから外部のネットワークのホストを呼び出す際に、外部への呼び出しを代行するサーバである [1]。

そのような代理サーバーを設置する理由として次のようなことがある。

まず、第１の理由は、セキュリティである。内部のネットワーク（イントラネット）と外部であるインターネットを接続する場合、内部のホストからインターネットを使いたいが、インターネットから内部のサーバーにアクセスされては困る、というときには、イントラネットとインターネットの接点にプロキシサーバを置き、内部から外部へのアクセスを代行させ、外部から内部へのアクセスは扱わないように設定することができる。

第２の理由は、性能である。一度アクセスした Web ページの内容はプロキシサーバが保存しておき、次回からはこれを使うようにする。すると、その都度インターネットを経由して情報を取りに行かなくてすむので、要求を出してから結果が戻るまでの時間を短縮することができる。

また、第三の理由として文字コードなどの、情報形式の変換ができるということである。

2.1.1　DeleGate について

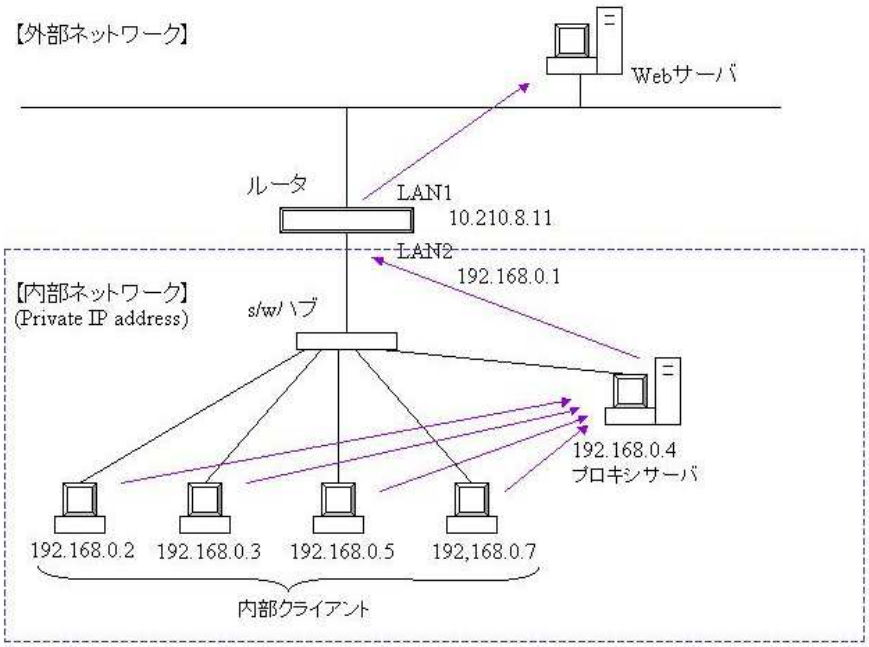
今回、WindowsNT4.0 のマシンに DeleGate のバージョン 6.1.18 をインストールしプロキシサーバとして稼動させた。

DeleGate は通産省電子技術総合研究所の佐藤豊氏が開発したプロキシサーバである。一般的にプロキシサーバという場合は，http プロキシを指すことが多い。しかし DeleGate は，http はもちろん，telnet や ftp，smtp，pop など，さまざまなプロトコルのプロキシサーバとして稼働することができる [2]。

DeleGate を用いることで内部ネットワークと外部ネットワークの分離をしたり、内部ネットワークに用意した Web サーバをプロキシサーバー経由で外部へ公開したりすることができる。

2.1.2　http における内部から外部へのアクセスに対する設定

内部クライアントから外部 Web サーバへのアクセスは、プロキシサーバを通して行うようにする。プロキシサーバでは DeleGate で内部クライアントからのアクセスを外部 Web サーバに中継する。このとき、プロキシサーバはクライアントからのアクセスに関するログを取得し、ファイルに保存する。



DeleGate の設定

DeleGate を起動させるにはインストール先のフォルダ（C:\delegate）に以下のような内容のバッチファイル（httpin.bat）と設定ファイル（httpin.cfg）を作り、そのバッチファイルを実行する。これにより設定ファイルで指定した設定で DeleGate が起動する。

—httpin.bat の内容———————————
C:\delegate\delegate.exe += httpin.cfg
———————————————————

http における内部から外部へのアクセスに対する DeleGate の設定は次のように設定ファイルで指定する。

—httpin.cfg の内容———————————
-P8080
SERVER=http
ADMIN=s97490@cis.shimane-u.ac.jp

RELIABLE=”192.168.0.[0-255]”
DGROOT=/delegate
PROTOLOG=/perl/login/[date+%Y/%m/%d/%H]

・-P8080
SERVER=http
で DeleGate を http 用プロキシとして 8080 ポートで使用するように設定する。
・ADMIN では DeleGate の管理者の e-mail address を指定。
・RELIABLE で指定されたホストからの要求のみ DeleGate で受け入れるようにする。
・DGROOT では全 DeleGate ファイルの元となるフォルダを指定している。
・PROTOLOG ではログファイルの出力先のパスを指定している。
パターン”[data+*format*]”がファイル (フォルダ) 名に含まれる場合、*format* 文字列は現在時刻で評価された値に置き換えられ、ファイル (フォルダ) は指定した時限で細分化される。この設定では 1 時間毎に別のファイルを作成しログを出力するようになる。

クライアント側の設定

クライアント側では Web ブラウザの設定でプロキシの使うようにしなければならない。設定で、DeleGate を実行しているプロキシサーバの IP アドレス 192.168.0.4 と、使用するポート番号 8080 を指定する。今回の研究で用いた Web ブラウザは Microsoft 社の Internet Explorer である。具体的な設定の順序を次に示す。

1. ブラウザの「ツール」の「インターネットオプション」をクリックする。
2. 「接続」を選択し、「LAN 設定」をクリックする。
3. 「プロキシサーバを使用する」にマークをし、「詳細」をクリックする。
4. 「使用するプロキシのアドレス」に DeleGate を実行しているプロキシサーバの IP アドレス 192.168.0.4 を指定し、「ポート番号」に DeleGate のサービスを使用するのに用いるポート番号 8080 を指定する。

2.2　ルータを用いてのログの収集

2.2.1　パケットフィルタリング

パケットフィルタリングとは、ルータやファイヤーウォールが持っている機能の一つで、送られてきたIPパケットを検査して通過させるかどうか判断する機能である。パケットのヘッダにはプロトコルや送信元アドレス、送信先アドレスやポート番号などの情報が含まれており、これを参照して通過するかどうかが決定される。通過できなかったパケットは送信元に通知されたり、破棄されたりする。どのような方針に基づいて判断するかは、そのネットワークの管理者が任意に設定することができる。

今回用いたルータ「YAMAHA RT140e」でも、この機能を使うことができ、このログを収集する。

2.2.2　ログを収集するための設定

パケット・フィルタリングのログ機能を有効にするには、事前に`syslog notice`コマンドの設定を有効しておく必要があり、ログ機能を有効にする場合は「**syslog notice on**」と設定する。

そして「**sysylog host** *host*」(*host*にはsyslogを受け取るホストのIPアドレス。今回はプロキシサーバのIPアドレス) コマンドを実行する。

また、windowsマシンでsyslogを受け取るには、別途ソフトが必要であり、今回は「3CSyslog」を用いた[3]。

第3章　準リアルタイムデータベース化

3.1　はじめに

本研究では、windowsNT4.0 のマシンに、日立で開発されたリレーショナルデータベースである HiRDB/Single Server Version6.0 をインストールし、データベースサーバとした。また、データベースクライアントとなるプロキシサーバには HiRDB クライアント (HiRDB/RunTime) をインストールした。

データベースの操作には SQL を用いる。SQL(Structured Query Language) とは、もともとは IBM 社が開発したデータベース操作用言語であり、現在では、ISO(国際標準化機構) で規格化され、リレーショナルデータベースを扱う国際標準言語となっている。

3.2　ログの整理

得られたログをデータベースに格納することを考えると、データベースへ格納するデータだけを抜き出してログを整理し、中間ファイルに書き出す必要がある。この作業をプロキシサーバ側で Perl を用いて行う。

Perl とは Larry Wall 氏が開発したプログラミング言語である。表記法は C 言語に似ており、テキストの検索や抽出に向いている。

このログを整理するプログラムを、プロキシサーバで得られたログ用と、ルータで得られたログ用の２つ作成する。

3.2.1　プロキシサーバのログ

プロキシサーバによって収集された http についてのログから得られるデータとしては、クライアントのホスト名、アクセスした日付と時間、アクセス先の URL がある。　ログを整理するプログラムでは、元のログを１行ずつ文字列として読み込み、それを、ホスト名、日時、アクセス先に分解し、そこからさらに、日時を西暦、月、日、時、分、秒に分解して中間ファイルに書き出す。データベースに格納した後、検索することを考えると、西暦、月、日、時、分、秒といったデータは数値として格納することが望ましい。そこで、月については英字表記であったのを数値に直し、日、時、分、秒が１桁の値になるときは 10 の位の 0 を取り除く。

整理前と整理後のログの例を次に示す。

プロキシで収集したログの例

●整理前

hanami - - [25/Jan/2002:17:12:32 +0900] "GET http://www.yahoo.co.jp/ HTTP/1.0" 200 22057 0*0.2

hanami - - [25/Jan/2002:17:13:17 +0900] "GET http://www.goo.ne.jp/ HTTP/1.0" 200 27341 0*0.160

hanami - - [25/Jan/2002:17:19:55 +0900] "GET http://www.delegate.org/delegate/ HTTP/1.0" 200 7

hanami - - [25/Jan/2002:17:21:13 +0900] "GET http://www.vector.co.jp/ HTTP/1.0" 200 24119 0*0.

●整理後

hanami 2002 1 25 17 12 32 http://www.yahoo.co.jp/

hanami 2002 1 25 17 13 17 http://www.goo.ne.jp/

hanami 2002 1 25 17 19 55 http://www.delegate.org/delegate/

hanami 2002 1 25 17 21 13 http://www.vector.co.jp/

3.2.2 ルータのログ

ルータで得られたログに対しても同様に、整理を行う。ルータによって収集された http についてのログから得られるデータとしては、はじめのアクセス時間、最後のアクセス時間、クライアントの IP アドレス・ポート番号、アクセス先の IP アドレス・ポート番号、アクセス回数がある。

ルータのログではアクセス先が同じで何行も続いているものがあり、このようなログは整理をするに際し、一行にまとめ、最初のアクセス時間、最後のアクセス時間、アクセス回数を記録するものとした。

ルータで収集したログの例

●整理前

Nov 22 15:57:58 192.168.0.1 LAN1 Passed at OUT(1) filter: TCP 192.168.0.4:4758 > 210.81.153.70:80

Nov 22 15:57:58 192.168.0.1 LAN1 Passed at OUT(1) filter: TCP 192.168.0.4:4758 > 210.81.153.70:80

Nov 22 15:57:58 192.168.0.1 LAN1 Passed at OUT(1) filter: TCP 192.168.0.4:4758 > 210.81.153.70:80

Nov 22 15:57:58 192.168.0.1 LAN1 Passed at OUT(1) filter: TCP 192.168.0.4:4758 > 210.81.153.70:80

Nov 22 15:57:58 192.168.0.1 LAN1 Passed at OUT(1) filter: TCP 192.168.0.4:4758 > 210.81.153.70:80

Nov 22 15:57:59 192.168.0.1 LAN1 Passed at OUT(1) filter: TCP 192.168.0.4:4758 > 210.81.153.70:80

Nov 22 15:57:59 192.168.0.1 LAN1 Passed at OUT(1) filter: TCP 192.168.0.4:4758 > 210.81.153.70:80

Nov 22 15:57:59 192.168.0.1 LAN1 Passed at OUT(1) filter: TCP 192.168.0.4:4758 > 210.81.153.70:80

Nov 22 15:58:00 192.168.0.1 LAN1 Passed at OUT(1) filter: TCP 192.168.0.4:4760 > 211.14.14.15:80

Nov 22 15:58:00 192.168.0.1 LAN1 Passed at OUT(1) filter: TCP 192.168.0.4:4765 > 61.200.81.166:80

Nov 22 15:58:00 192.168.0.1 LAN1 Passed at OUT(1) filter: TCP 192.168.0.4:4765 > 61.200.81.166:80

Nov 22 15:58:00 192.168.0.1 LAN1 Passed at OUT(1) filter: TCP 192.168.0.4:4765 > 61.200.81.166:80

●整理後

11 22 15 57 58 11 22 15 57 59 192.168.0.4:4758 210.81.153.70:80 8

11 22 15 57 0 11 22 15 57 0 192.168.0.4:4760 211.14.14.15:80 1

11 22 15 58 0 11 22 15 58 0 192.168.0.4:4765 61.200.81.166:80 3

3.3 表の定義

整理されたログの各データをデータベースへ格納するために、あらかじめ表を定義しておかなけらばならない。プロキシサーバで収集したログのデータを格納する表の名前は login, ルータで収集したログのデータを格納する表の名前は routerlog として定義する。

表 login を定義するために実行した SQL 文を以下に示す。

```
create table login(
  ホストメイ      char(50),
  セイレキ        integer,
  ツキ            integer,
  ヒ              integer,
  ジ              integer,
  フン            integer,
  ビョウ          integer,
  アクセスサキ    char(100)
)
```

ホストメイとアクセスサキは文字型、セイレキ、ツキ、ヒ、ジ、フン、ビョウは整数型として格納するようにする。

実際に作成される表は次のようになる。

ホストメイ	セイレキ	ツキ	ヒ	ジ	フン	ビョウ	アクセスサキ

表 routerlog を定義するために実行した SQL 文を以下に示す。

create table *login*(
firstmonth integer,
firstdate integer,
firsthour integer,
firstminute integer,
firstsecond integer,
lastmonth integer,
lastdate integer,
lasthour integer,
lastminute integer,
lastsecond integer,
client char(22),
host char(22),
times integer)

実際に作成される表は次のようになる。

fristmonth	firstdate	firsthour	firstminute	firstsecond	lastmonth	lastdate	lasthour	lastminute	lastsecond	client	host	times

3.4 JDBC を用いての格納

3.4.1 JDBC とは

JDBC とは、SQL ステートメントを実行するための Java API であり、JDK1.1 以降をベースとした Java のランタイム環境では標準で実装されている。

JDBC は ODBC をベースに開発されており、ODBC を Java 向けにアレンジしたインターフェースであると考えることができる。

ODBC

ODBC(Open Database Connectivity) とは、国際標準規格 SQL/CLI に準拠した、データベースに接続するためのソフトウェアの標準仕様である。以前はA社のデータベース用に開発されたクライアントのアプリケーションはB社のデータベースには使えないという問題があったが、各データベースの違いは ODBC ドライバによって吸収されるため、ユーザは ODBC に定められた手順に従ってプログラムを書けば、接続先のデータベースがどのようなデータベース管理システムに管理されているか意識することなくアクセスできる [4]。

3.4.2 HiRDB Driver for JDBC の概要

HiRDB Driver for JDBC は，Java からデータベースアクセスを実現するためのプログラムであり、データベースの利用は，HiRDB Driver for JDBC の Java クラスを使用して作成したプログラムから実行する。

JDBC ドライバは、JDBC と ODBC（Open Database Connectivity）をマッピングして実行している。したがって，HiRDB Driver for JDBC が使用する JDBC の機能は，ODBC のサポート範囲となる。

本研究では、プロキシサーバ用と、ルータのログ用の２つのプログラムを作成した。

3.4.3 環境設定

HiRDB Driver for JDBC のインストール後，環境変数 PATH に HiRDB Driver for JDBC の JDBC ドライバの部分を格納しているディレクトリ

（[インストール先ディレクトリ] bin）

を登録しなければならない。

また、JDBC ドライバクを複写して使用する場合は，環境変数 CLASSPATH に JDBC ドライバの複写先ディレクトリを登録しておかなければならない。複写先ディレクトリ以下は，次に示すような構成にする。

［複写先ディレクトリ］￥JP￥co￥Hitachi￥soft￥HiRDB_Driver_for_JDBC ￥*.class

「*」は，ファイルを識別するための任意の文字列。

HiRDB Driver for JDBC でのパッケージ名は,「JP￥co￥Hitachi￥soft ￥HiRDB_Driver_for_JDBC」となり、パッケージ名とディレクトリ名を対応させている Java の仕様のため,「JP￥co ￥Hitachi ￥soft￥HiRDB_Driver_for_JDBC」の部分は，変更してはならない。

JDBC ドライバから接続するデータベースの指定は，ODBC のデータソース名を使用するので、あらかじめデータソース名を登録しておかなければならない。なお， HiRDB Driver for JDBC で使用するデータソース名は，システムデータソースとして登録しておく必要がある。

3.4.4 JDBC を用いて作成するプログラム

JBuilder

今回、Java のプログラムを作成するにあたり、JBuilder を用いた。

JBuilder は、簡単な作業で JDBC ドライバを開発環境内にセットアップして利用することができるようになっており、面倒な操作なしにデータベースにアクセス可能である。また、データベースのデータセットへのアクセスを簡単に行うためのツールも用意され、非常にビジュアルな開発環境の中でデータベースを扱うことができる。

HiRDB Driver for JDBC を使用するための指定

HiRDB Driver for JDBC をロードするときに必要な指定項目にパッケージ名称及びドライバ名称がある。それぞれの名称を次に示す。

パッケージ名称：JP.co.Hitachi.soft.HiRDB_Driver_for_JDBC
ドライバ名称：JdbcHirdbDriver

パッケージ名称及びドライバ名称はプログラム内の Class.forName メソッドで次のように記述しロード指定する。

```
Class.forName("JP.co.Hitachi.soft.HiRDB_Driver_for_JDBC.JdbcHirdbDriver")
```

接続するデータベースの指定

データベースは，java.sql.DriverManager.getConnection メソッドで指定する。

```
DriverManager.getConnection(String url,String user,String password);
```

URL の記述規則

引数で指定する URL は次の形式で指定する。

```
jdbc:hitachi:hirdb://接続先ホスト名［:ポート番号］/:DSN=データソース名
```

SQL

プログラム中において、ログデータをデータベースへ格納するために実行される SQL 文を次に示す。

```
INSERT INTO *1 VALUE *2;
```

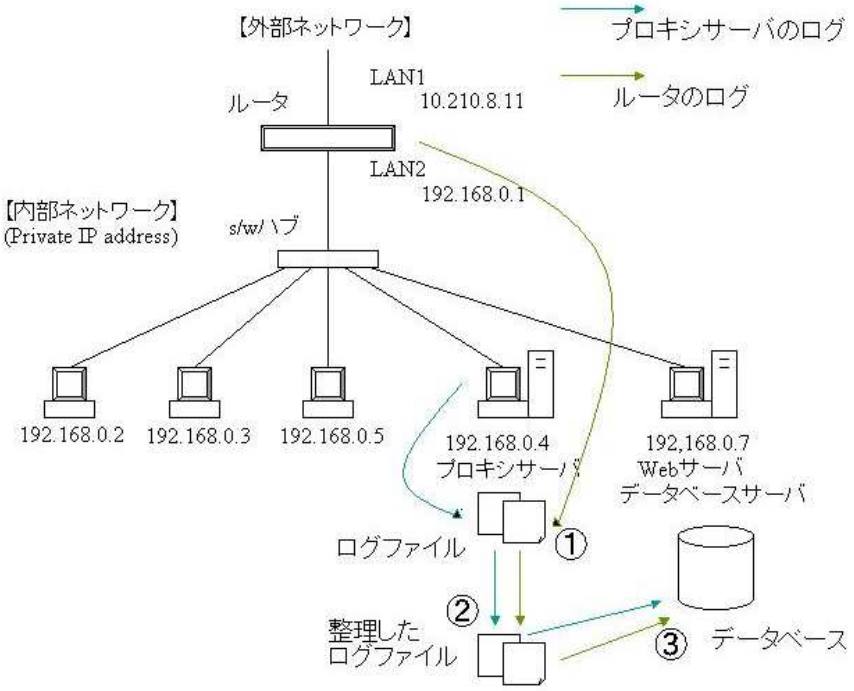
*1 テーブル名　　*2 ログデータの値の並び

3.5　準リアルタイム化

指定したプログラムを定時に実行するには、windows のタスクのスケジュール機能を用いる。スケジューラの設定で、実行するプログラムと、そのプログラムを実行する時間を指定し、プロキシサーバ、ルータ、それぞれについてのログを整理するプログラムと、データベースへ格納するプログラムを 1 時間毎に実行するようにする。

データベースに格納するプログラムでは、整理したログを書き出した中間ファイルからデータを読み込み、そのデータをデータベースへ格納する。

ログデータの流れをネットワーク図を用いて説明する。



1　プロキシサーバのログはそのままプロキシサーバでファイルに記録し、ルータのログはプロキシサーバで受け取り、ファイルに記録する。

2　プロキシサーバにおいて、それぞれのログ用の、ログを整理するプログラム実行し、新たなファイルに整理後のログデータを書き出す。

3　プロキシサーバ側から、それぞれの整理後のログ用の、データベースへ格納するプログラムを実行し、データベースサーバの定義された表へデータを格納する。

第4章　Ｗｅｂからのデータベースアクセス

4.1　アプレット

4.1.1　アプレットについて

アプレットとは、ネットワークを通じてWebブラウザにダウンロードされ、ブラウザのウィンドウに埋め込まれて実行されるJavaプログラムのこと。これにより、静的なHTMLページだけではできないような高度な機能を持たせたり、柔軟性のある機能やユーザーインターフェイスを実現することができる。

ただし、インターネットやイントラネットなどを介してプログラムをサーバからクライアントに送るため、セキュリティ的にJavaアプレットから実行できる処理は制限されており、ファイル入出力などはJavaアプレットからは行えないし、アプレットがあるWebサーバ以外へネットワーク接続できない。

なお、JDBC Driver for HiRDBでは、アプレットを実行させるWebブラウザとして、Netscape Communicator 4.01以降についてのみ、動作の保証をしている。

4.1.2　アプレットのサーバへの登録

今回は、JDBC Driver for HiRDBに付属していたサンプルのアプレットプログラムを起動してみた。

アプレットの場合は，作成したクラスをサーバに格納しておく必要がある。アプレットのクラスファイルの位置は，HTMLのAPPLETタグのCODE，及びCODEBASE属性で指定する。

HTMLファイル，アプレットクラスファイル，及びJDBCドライバ（JPディレクトリ）を同一ディレクトリに格納した場合HTMLタグの指定は次のようになる。

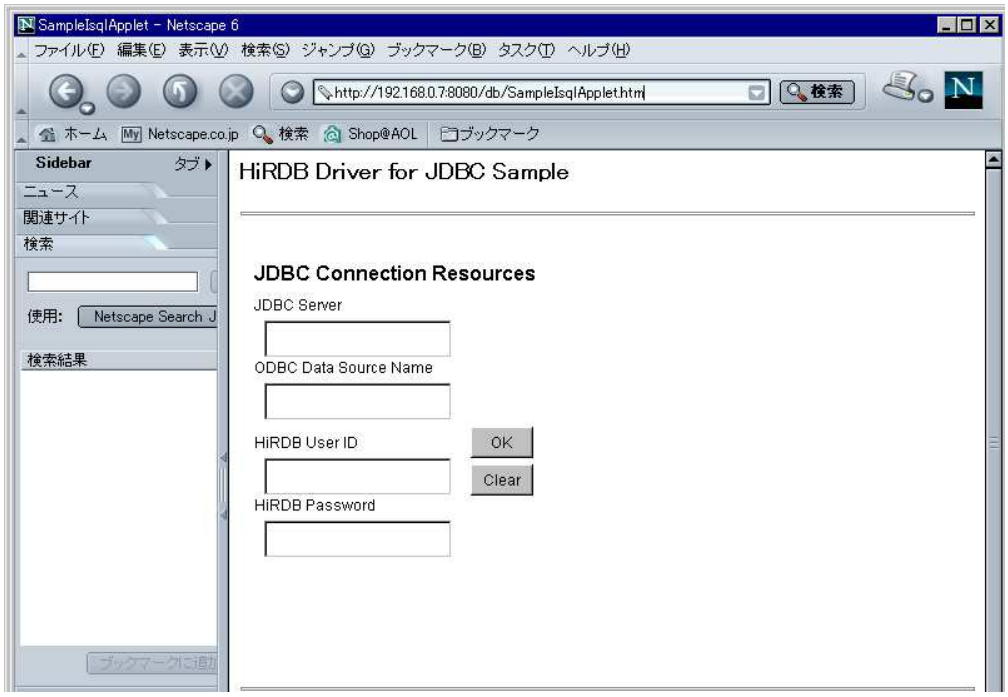
```
<APPLET CODE=”testApplet.class” WIDTH=500 HEIGHT=500 >
</APPLET >
```

4.1.3　HTMLでのアプレットの実行例

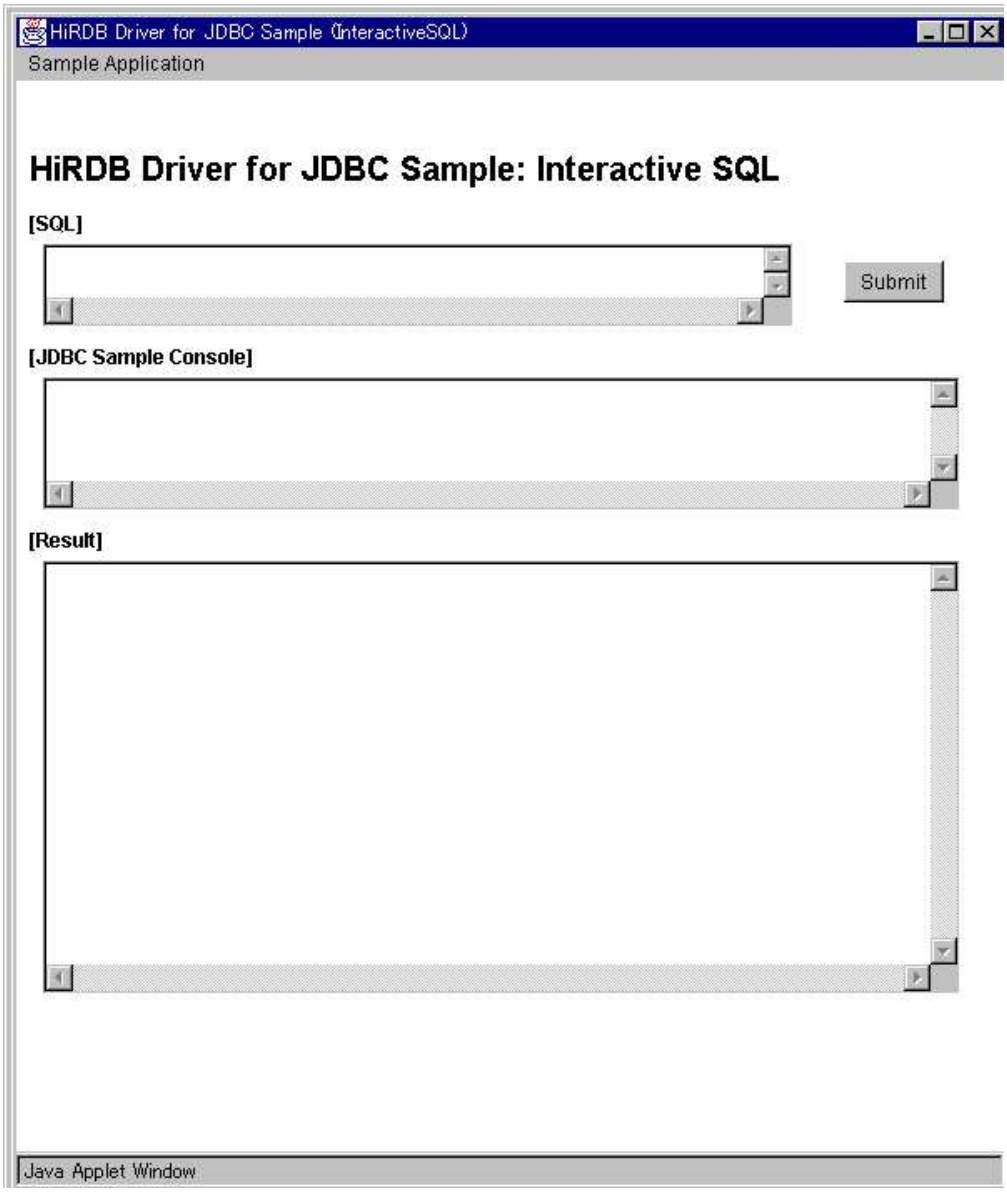
実際の実行結果を次に示す。

なお、アプレットの実行に用いたブラウザは Netscape6.2.1である。

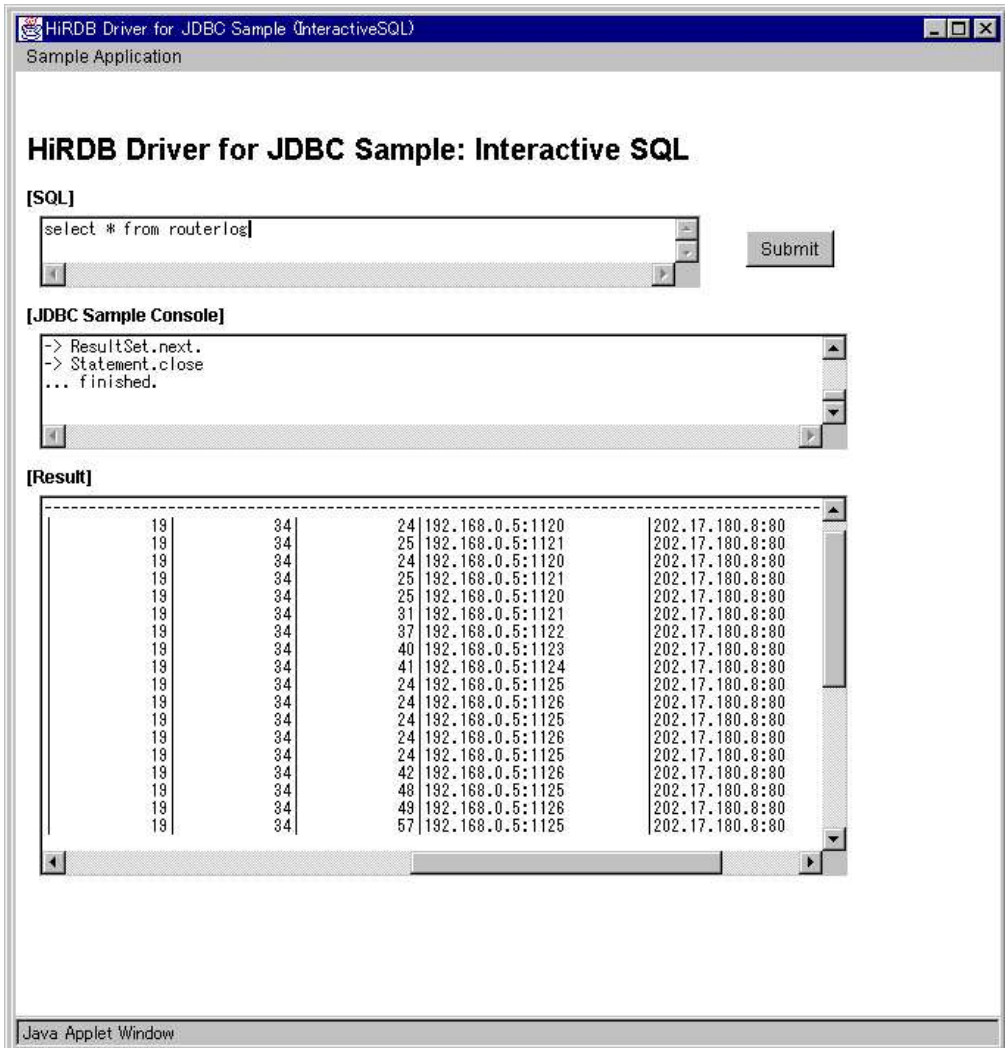
まず、Netscape を立ち上げ URL を指定すると、アプレットをダウンロードし、下のような画面になる。



前のページで、「JDBC server」,「ODBC Data Source Name」,「HiRDB User ID」,「HiRDB password」の欄に正確に記入し、OK ボタンを押すとプログラムが実行され下のようなパネルが生成される。



SQL の欄に SQL 文を書き込み、Submit ボタンを押すとデータベースとの接続が行われ、SQL 文の実行結果が表示される。



4.2　サーブレット

4.2.1　サーブレットについて

サーブレットは、J2EE(Java 2 Platform,Enterprise Edition) という、サン・マイクロシステムズが規定したサーバーサイドシステム構築の使用に含まれている機能の一つである。

Java を使って web サーバーからクライアントに対して HTML などのコンテンツを返す役割を果たす。従来の CGI(Common Gateway Interface) とは異なり、java の持つさまざまな機能を活用できるサーブレットは Java の特性を生かした web サーバーの拡張が可能になり、非常に洗練された web システムの構築を行うことができる。

サーブレットは、web サーバー上で実行されるモジュール形式の Java プログラムであり、サーブレットを動作させるには、サーブレットコンテナと呼ばれる実行環境が必要である。サーブレットコンテナは、サーブレットを作成してメモリにロードし、クライアントからのリクエストを受け付ける状態にする。また、サーブレットが作成したコンテンツをクライアントに提供したり、ロードされたサーブレットをメモリから破棄するといった制御を行う。

サーブレットコンテナは、Java 仮想マシン (Java Virtual Machine： JVM) によって実行される。そして、サーブレットはサーブレットコンテナ内のオブジェクトとして扱われる。

サーブレットコンテナは複数のサーブレットを管理するとともに、サーブレットの制御を行ってクライアントにサービスを提供する。サーブレットをプログラミングするをいうことは、サーブレットコンテナが制御するクラス（オブジェクト）を作成することを意味する。

4.2.2　JBuilder を用いてのプログラムの作成

JBuilder は、サーブレットコンテナと連携して、作成したサーブレットをメニューから実行することができる。これは、作成したものをすぐに試せることができるというメリットにつながる。

JBuilder は、サーブレットコンテナとして「Tomcat」を搭載しており、開発環境のメニューから簡単に起動して、サーブレットを実行することができる。また、ソースコードレベルでのデバッグが可能になっているため、プログラムのバグ（不具合）を追跡・発見できまる。

Tomcat は、Apache の The Jakara Project が提供するサーブレットコンテナである。オープンソースライセンスで自由に使用できるとともに、常に最新の J2EE(Java 2 Platform,Enterprise Edition) の使用をサポートしている。JBuilder は、Tomcat を搭載しており、面倒なセットアップをすることなく、開発環境からすぐに使用することができる。

JBuilder で作成したサーブレットは、メニューから実行することができる。一度実行されると Tomcat が起動し、webブラウザを利用して、サーブレットからコンテンツを受け取ることができるようになる。

4.2.3　HTML でのサーレットの実行例

まず Web ブラウザを呼び出し URL を指定しトップページを表示させる。
時間を指定し、検索ボタンを押すと指定した値が、サーブレットのプログラムへ渡される。
なお、用いた Web ブラウザは Internet Explorer 5.5 である。

サーブレットプログラムではデータベースに接続し、受け取った値を使って検索を行い、帰ってきた値を使って、実行結果を表示する HTML ファイルを作成する

今回作ったプログラムでは、データベースに接続時に必要な、ユーザ ID やパスワードなどは、プログラム中に書いておいた。

4.3 アプレットとサーブレットの違い

アプレットはサーバーからダウンロードして実行するので、複雑で大きなアプレットをダウンロードするには時間がかかってしまう。また、アプレットがデータベースサーバーと直接アクセスするので Webブラウザを実行しているクライアントとデータベースサーバーとがネットワーク接続できている必要がある。

サーブレットは、サーブレットコンテナが Java 仮想マシンの環境下で動作しているプロセスとなる。そして、そのプロセスが作成したオブジェクトが、リクエストに対する HTML コンテンツなどを生成して、クライアントに返す処理を行う。つまり、該当するクラスをインスタンス化して、オブジェクトを作成する処理が行われる。これ以降、作成されたインスタンスは、サーブレットコンテナが責任を持って管理することになり、サーブレットはコンテンツの作成が終了しても消滅することはない。作成されたインスタンスはサーブレットコンテナに管理されることになり、サーブレットコンテナが終了したり、任意の破棄要求が無い限りはメモリに常駐する [5]。したがって、リクエストの度にデータベース接続処理を行う必要が無くなる為、処理が軽くなる。

4.4 暗号化

Web ページとデータベース間の通信において、個人情報などの漏洩してはならない重要な情報は、盗聴されることを防がなければならない。そこで、Web サーバとクライアント間で、やりとりされる情報に、SSL を使用し、暗号化を行う。

今回、Web サーバとして Windows 2000 Internet Information Service (IIS) 5.0 を用い SSL を設定する方法を、例を挙げて説明する。

4.4.1 SSL

SSL とは、Webブラウザと Web サーバ間で安全な通信を行なうために Netscape Communications が開発したセキュリティ機能である。IETF による標準化も行なわれており、TLS(Transport Layer Security) として RFC2246 になっている。認証局の署名の入った証明書を使ったサーバの認証と Webブラウザと Web サーバ間での通信内容の暗号化という 2 つの機能を持つ。

SSL は TCP 層とアプリケーション層間に位置するプロトコル層として位置しており、HTTP に限らず Telnet や FTP、SMTP などのさまざまなアプリケーションプロトコルを暗号化できる点が特徴である。ただし、UDP パケットの暗号化には対応していない。

Netscape Navigator や Internet Explorer といった Webブラウザがすでに SSL への対応を行なっているため、ユーザーが特に意識せずに使うことができるというメリットもある。

現在では、Apache、Netscape Web Server、IIS など主要な Web サーバで利用することができ、E コマースやオンラインバンキングなどのサイトで一般的に用いられている。

4.4.2 独自 CA の構築と設定

Microsoft 証明書サービスを利用すると、公的 CA とは別に独自の CA(認証機関) を設けることができる。独自 CA は、機能的には公的 CA と変わらず、手元に CA が構築できるので、 証明書の発行等を手軽に行える。 今回、この独自 CA により、研究室内でのプライベートなネットワー

クでの SSL 暗号化通信のテストを行った。
全体的な作業の流れは、以下のようになる。

1. Microsoft 証明書サービスのインストール (独自 CA 側)
2. CA 証明書のインストール (Web サーバ側)
3. CA 証明書のパスのインストール (Web クライアント側)
4. サーバー証明書要求の作成 (Web サーバ側)
5. 独自 CA へのサーバー証明書要求の送信 (Web サーバ側)
6. サーバー証明書の発行 (独自 CA 側)
7. サーバー証明書の取得 (Web サーバ側)
8. サーバー証明書のインストール (Web サーバ側)
9. Web サイトの SSL 暗号化通信の設定 (Web サーバ側)

Microsoft 証明書サービスのインストール (独自 CA 側)

Microsoft 証明書サービスは、 通常の Windows 2000 インストールには含まれていないため、インストールされていない場合は、追加インストールを行う。 [アプリケーションの追加と削除] コントロールパネルから [Windows コンポーネントの追加と削除] を起動し、[証明書サービス] をチェックする。[アプリケーションの追加と削除] コントロールパネルから [Windows コンポーネントの追加と削除] を起動し、[証明書サービス] をチェックする。

CA の識別情報を入力する。今回は、例として、CA「IPA-sec CA」を作成する。

CA 証明書のインストール (Web サーバ側)

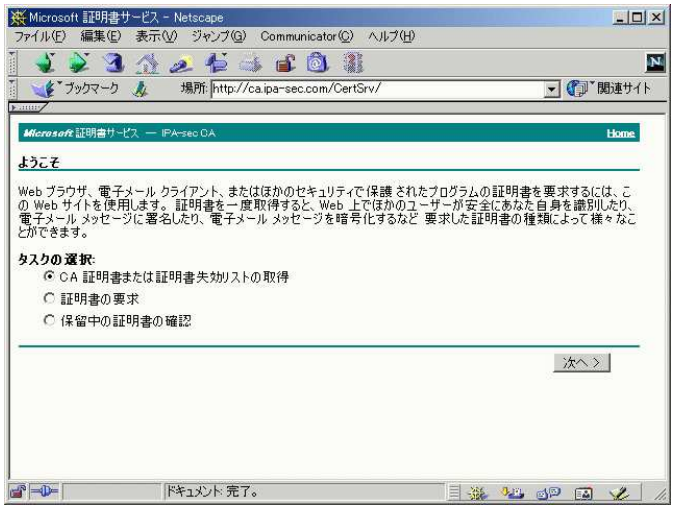
独自 CA を利用する SSL 暗号化通信を行う場合、独自 CA の CA 証明書を Web サーバ側に予めインストールする必要がある。

独自 CA を利用する Web サーバーは、独自 CA の CA 証明書をインストールすることで、独自 CA から発行されるサーバー証明書を信頼することができる。

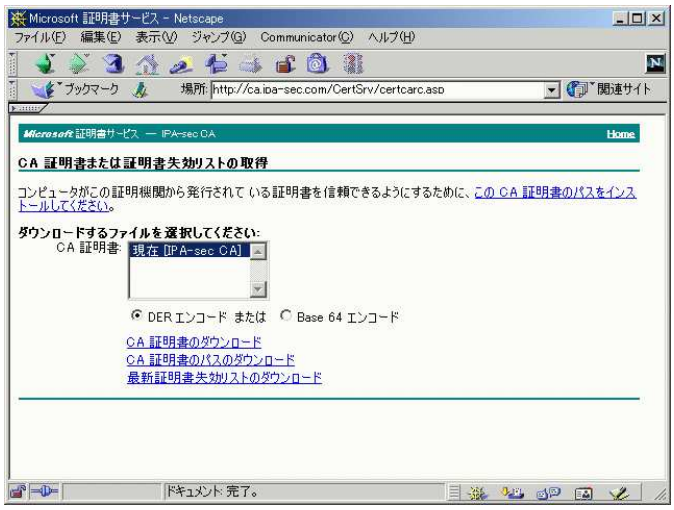
CA 証明書は、独自 CA の [証明書サービス Web ページ] からダウンロードする。独自 CA の [証明書サービス Web ページ] は、以下の URL にあるので、Web ページからアクセスする。

http://[独自 CA のドメイン名もしくはコンピュータ名]/CertSrv/

[CA 証明書または証明書失効リストの取得] をチェックして、[次へ] ボタンをクリックする。



[CA 証明書のダウンロード] をクリックして、CA 証明書をダウンロードする。



ダウンロードした CA 証明書をダブルクリックする。CA 証明書の内容が表示される。 ここで、[証明書のインストール] ボタンをクリックする。

証明書のインポート ウィザードが起動する。

以降、ウィザードのデフォルト設定のまま［次へ］ボタンをクリックしていけば、CA 証明書のインストールは完了。

CA 証明書のパスのインストール (Web クライアント側)

独自 CA を利用する SSL 暗号化通信を行う場合、独自 CA の CA 証明書のパスを Web クライアント側に予めインストールする必要がある。独自 CA を利用する全ての Web クライアントは、独自 CA の CA 証明書のパスをインストールすることで、独自 CA から発行されるサーバー証明書をインストールした Web サイトを信頼することができる。

CA 証明書のパスは、独自 CA の [証明書サービス Web ページ] からインストールする。[CA 証明書または証明書失効リストの取得] をチェックし、 [次へ] ボタンをクリックする。

[この CA 証明書のパスをインストールして下さい] をクリックする。なお、ここから先は、Web クライアントによって動作が異なる。ここでは、Internet Explorer における CA 証明書のパスのインストールについて説明する。

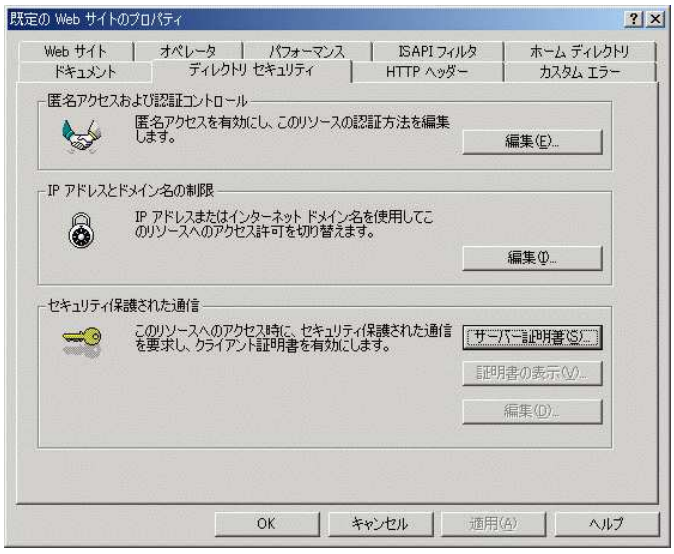
Internet Explorer の場合は、ルート証明書ストアが起動する。[はい (Y)] ボタンをクリックする。

証明書がインストールされたことを確認する。[ツール (T)] メニューの [インターネットオプション (O)...] を選択する。[インターネットオプション] プロパティが表示されるので、[コンテンツ] タブをクリックする。[証明書 (C)..] ボタンをクリックすると、[証明書] ダイアログが表示されるので、[信頼されたルート証明機関] をクリックする。このリストに”IPA-sec CA”があることを確認する。

サーバー証明書要求の作成 (Web サーバ側)

サーバー証明書要求は、[インターネットインフォメーションサービス] から作成する。タスクバーの [スタート] ボタンをクリックし、[プログラム (P)]-[管理ツール]-[インターネットインフォメーションサービス] を選択すると、[インターネットインフォメーションサービス] が起動する。[インターネットインフォメーションサービス] ウインドウの左ペインで、SSL 暗号化通信を設定したい Web サイトを右クリックして、[プロパティ(R)] を選択する。ここでは、[既定の Web サイト] に SSL 暗号化通信を設定するものとし、[既定の Web サイト] のプロパティを開きます。

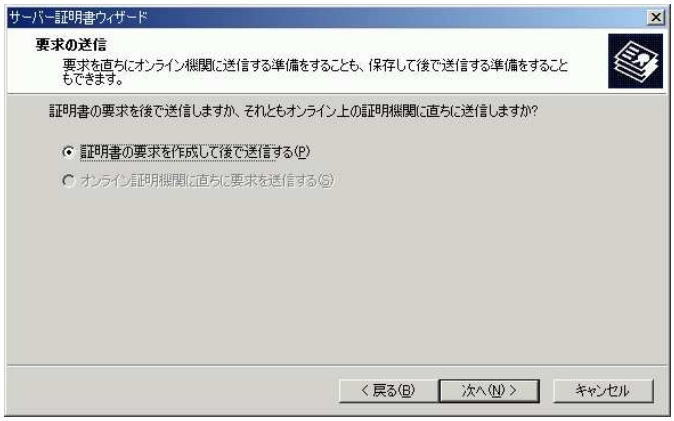
次に、Web サイトのプロパティの [ディレクトリセキュリティ] タブをクリックし、[サーバー証明書 (S)...] ボタンをクリックする。



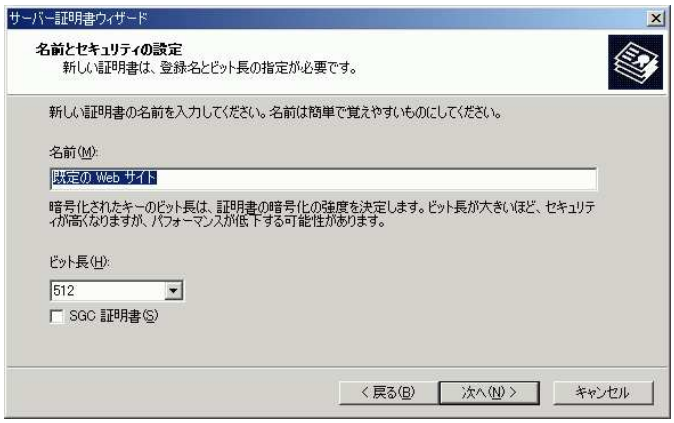
サーバー証明書ウィザードが起動する。 [次へ] ボタンをクリックする。

証明書の割り当て方法を選択する。ここでは、新規にサーバー証明書を作成するので、[証明書の新規作成 (C)] を選択する。

要求の送信方法を選択する。ここでは、後で要求を送信するので、[証明書の要求を作成して後で送信する (P)] をチェックする。



証明書の名前と暗号化キーのビット長を設定する。ここでは、名前を「既定の Web サイト」、ビット長を 512 ビットとする。



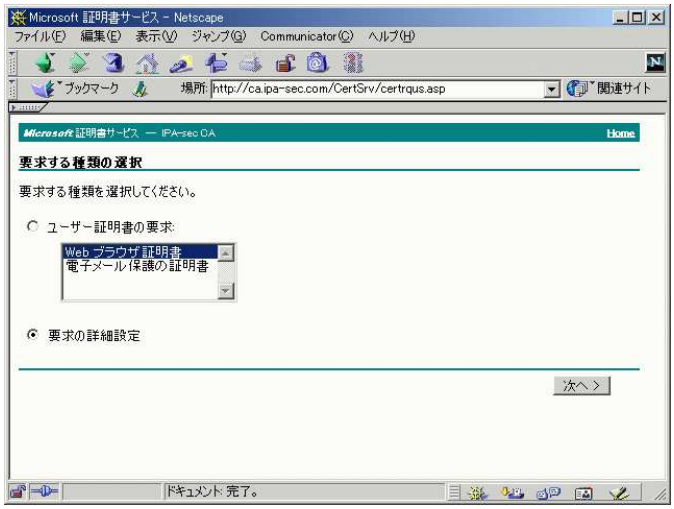
証明書要求の内容を出力するファイルを設定する。

ここまでの設定内容が表示される。[完了] ボタンをクリックすれば、サーバー証明書要求の作成は完了。

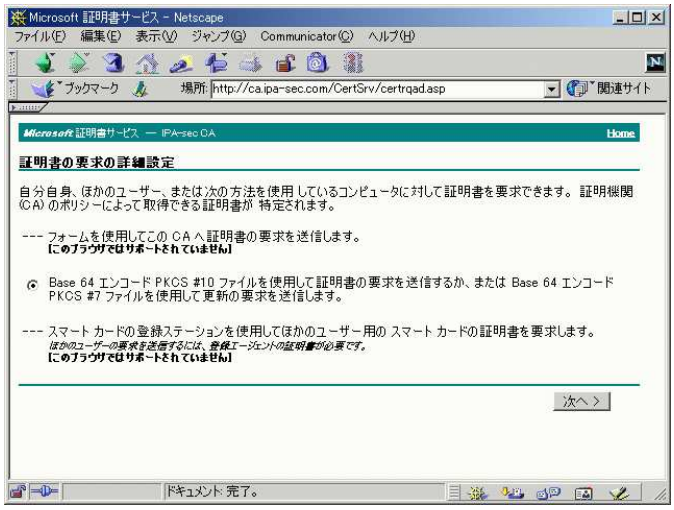
独自 CA へのサーバー証明書要求の送信 (Web サーバ側)

独自 CA へのサーバー証明書要求の送信は、独自 CA の [証明書サービス Web ページ] から送信できる。[証明書の要求] をチェックして、[次へ] ボタンをクリックする。

[要求の詳細設定] を選択して、[次へ] ボタンをクリックする。



[次へ] ボタンをクリックします。

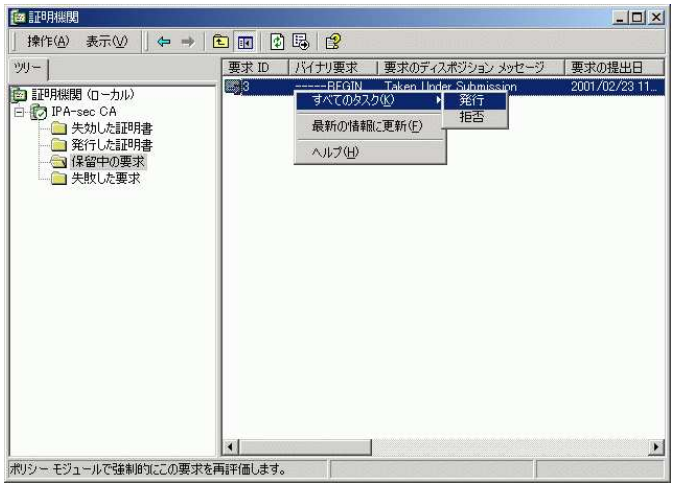


[保存された要求:] テキストボックスに、上記で作成したサーバー証明書要求ファイルの内容をコピーして貼り付け、[送信] ボタンをクリックします。

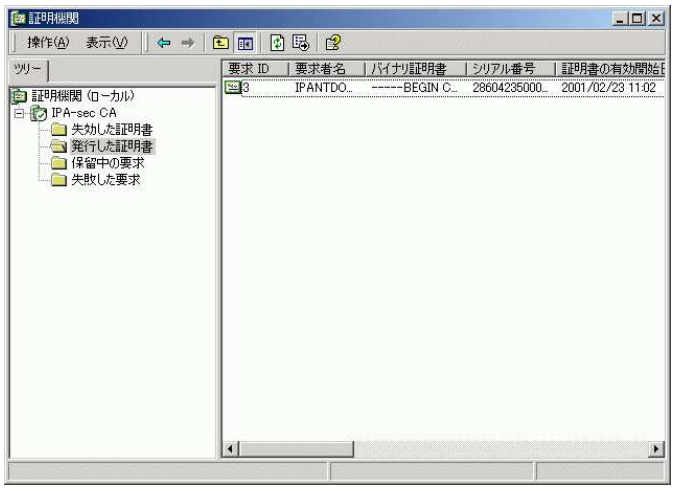
サーバー証明書の発行 (独自 CA 側)

サーバー証明書の発行は、[証明機関] から行う。タスクバーの [スタート] ボタンをクリックし、[プログラム (P)]-[管理ツール]-[証明機関] を選択すると、[証明機関] が起動する。

[証明機関]] ウインドウの左ペインにて、[証明機関 (ローカル)]-[[証明機関名]]-[保留中の要求] をクリックする。ここでは、[証明機関 (ローカル)]-[IPA-sec CA]-[保留中の要求] をクリックする。すると、右ペインに上記で送信したサーバー証明書要求が表示される。このサーバー証明書要求を発行する。証明書要求を右クリックして、[すべてのタスク (K)]-[発行] を選択する。



サーバー証明書が発行されたことを確認するには、 [発行した証明書] をクリックする。
右ペインに発行したサーバー証明書が表示される。



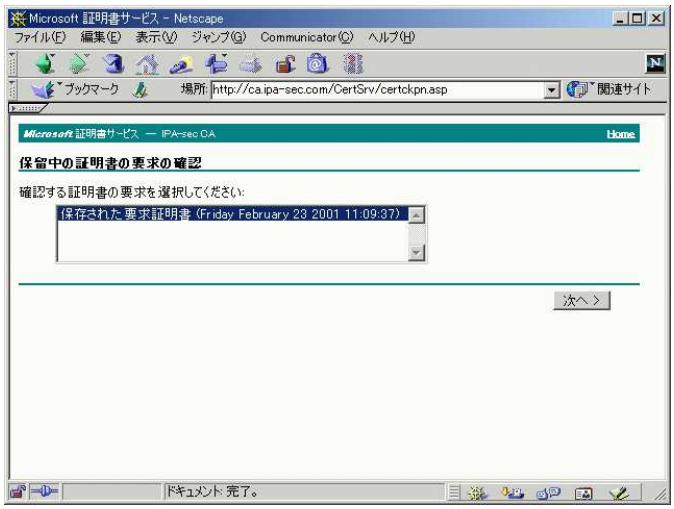
サーバー証明書の取得 (Web サーバ側)

発行されたサーバー証明書は、独自 CA の場合は、Web サーバ側から取得することが可能である。

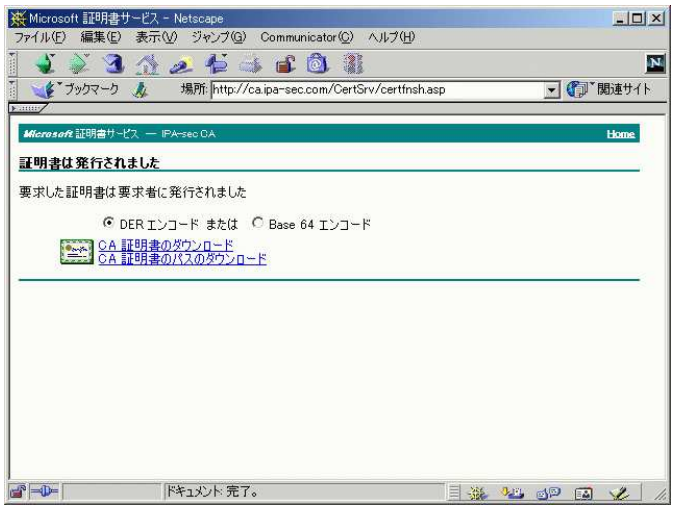
独自 CA で発行したサーバー証明書は、 独自 CA の [証明書サービス Web ページ] から取得できる。

[保留中の証明書の確認] をチェックして、[次へ] ボタンをクリックする。

発行したサーバー証明書が、[保留中の証明書の要求の確認] に表示される。取得するサーバー証明書を選択して、[次へ] ボタンをクリックする。

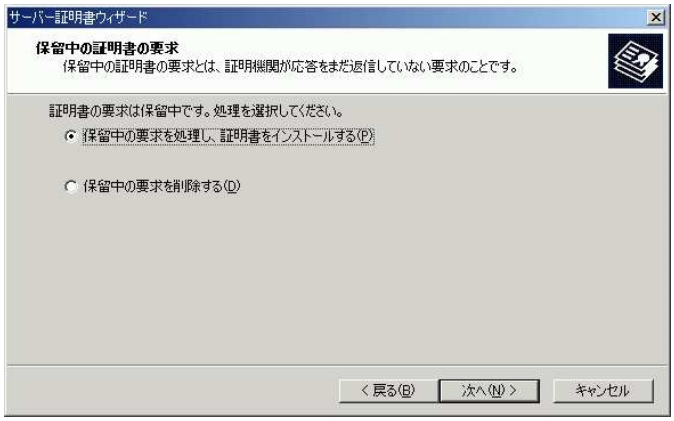


[CA 証明書のダウンロード] をクリックして、サーバー証明書をダウンロードする。



サーバー証明書のインストール (Web サーバ側)

サーバー証明書要求を作成した Web サイトのプロパティの [ディレクトリセキュリティ] ページから、[サーバー証明書 (S)...] ボタンをクリックする。
サーバー証明書要求の作成時と同じように、サーバー証明書ウィザードが起動する。[次へ] ボタンをクリックする。
サーバー証明書要求の作成時には、証明書の割り当て方法の選択だったページが、保留中の証明書の要求の選択のページに切り替わっている。ここで、[保留中の要求を処理し、証明書をインストールします (P)] を選択する。

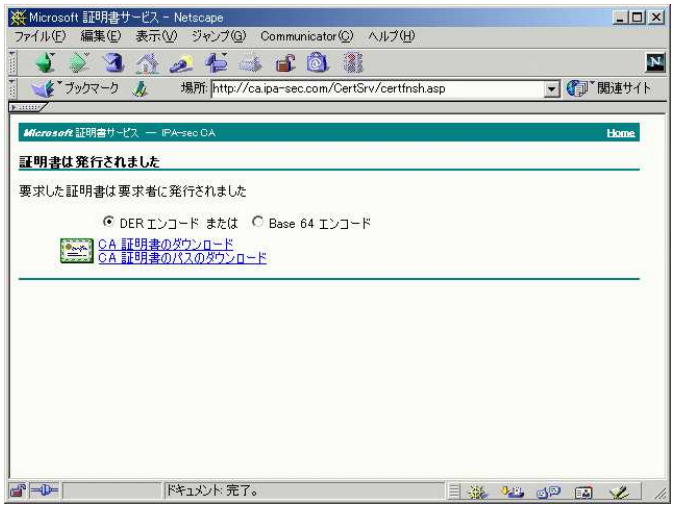


証明機関の応答ファイルのパスとファイル名を入力する。証明機関の応答ファイルとは、サーバー証明書のことである。ここでは、前述でファイルとして保存したサーバー証明書へのパスとファイル名を指定する。

サーバー証明書の設定内容が表示される。[次へ] ボタンをクリックする。

[完了] ボタンをクリックすれば、サーバー証明書のインストールは完了。

Web サイトのプロパティの [ディレクトリセキュリティ] ページから、[証明書の表示 (V)...] ボタンをクリックし、サーバー証明書が正しくインストールされていることを確認する。



Web サイトの SSL 暗号化通信の設定 (Web サーバ側)

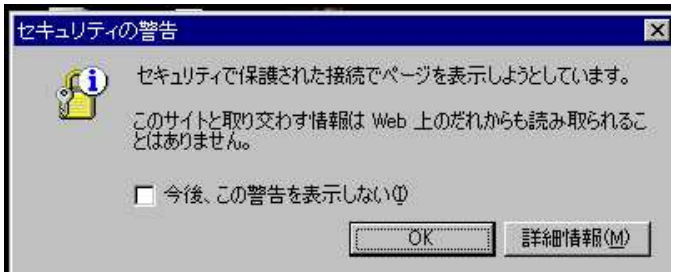
サーバー証明書をインストールした Web サイトのプロパティの [ディレクトリセキュリティ] ページから、[編集 (D)...] ボタンをクリックする。

[セキュリティ保護された通信] ダイアログが表示されます。SSL 暗号化通信を有効にするには、[保護されたチャンネル (SSL) を要求する (R)] をチェックした後、[OK] ボタンをクリックする。

以上で、独的 CA を利用した Web サイトの SSL 暗号化通信の設定は完了。

4.4.3 動作確認

Web ページを表示して、証明書が機能していることを確認する。



ブラウザで、HTTP を経由してサイトにアクセスすると、以下のようなエラーメッセージが表示される。

HTTP 403.4 - アクセスは許可されていません: SSL が必要です。

セキュリティ保護された接続 (https:// と URL を指定する) を使い、同じ Web サイトを表示すると、表示される。

第5章　終論

これまでに述べてきた実装方法により、準リアルタイムでログデータをデータベースへの格納し、web から検索を行うことができた。

今回の研究では、収集し処理したログは http についてのみであったが、smtp や pop など他のプロトコルにおいても同様のことができれば、ネットワークモニタとして、よりよいものとなるだろう。また、ログデータは単に格納されるまでであったが、格納したデータをデータマイニングすることにより、クライアントの傾向を探ったり、不正アクセス時の特徴的なパターンを見つけ出し、あやしいアクセスを検出するなどといった方面への発展が考えられる。

なお、システムの構築の面から考えると、より一層のセキュリティの強化を図らなければならない。

参考文献・URL

[1] D.Brent Capman 、Elizabeth D.Zwicky 共著
「ファイアウォール構築」
株式会社 オライリー・ジャパン 197 頁 (2000)

[2] 「DeleGate Home Page (www.delegate.org)」
http://www.delegate.org/delegate/

[3] 「FAQ for YAMAHA RT Series」
http://www.rtpro.yamaha.co.jp/RT/FAQ

[4] Kyle Geiger 著
「INSIDE ODBC」
株式会社 アスキー 23 頁 (1996)

[5] 古川正寿 著
「Jbuilder サーブレット開発」
ソフトバンク パブリッシング株式会社 7 頁 (2001)

謝辞

本研究において、最後まで熱心な御指導をいただきました田中章司郎教授には心より御礼を申しあげます。また、田中研究室生の藤井宏行さん、森本誠人さん、江野本隆行君には、本研究に関して様々な助言をいただきました。ここに感謝の意を表します。

なお、本論文と本研究で作成したプログラムなどのすべての著作権は田中章司郎教授に譲渡いたします。