# STARS チュートリアル

**小菅 隆**

**Ver.1.1 2013/03/27**

>List

10 STARS のとりあえず *‘とりあえず STARS を動かしてみます。*

20 STARS の詳細 *‘STARS の詳細について解説します。*

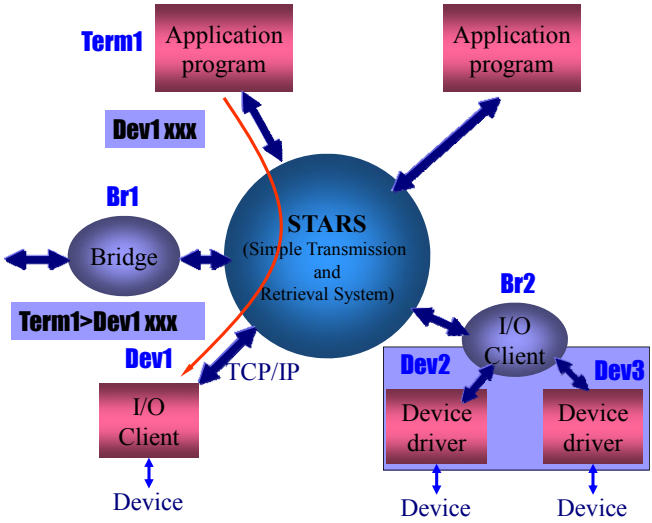
30 STARS Perl Interface モジュールと Perl Wizard *‘Perl モジュールの説明と Wizard を使ったプログラミングについて解説します。*

40 STARS .NET Interface とテンプレート *‘.NET 用のインターフェースの説明とテンプレートを使ったプログラミングを解説します。*

# 10　STARS のとりあえず

## STARS って

「http://stars.kek.jp」から STARS に関するソフトウエアと情報は入手可能で、無料で利用可能です。ただし、あくまでも自己責任に於いて使用してください。




STARS の概要：TCP/IP Socket とテキストのメッセージ交換でシステムを構築

STARS（Simple Transmission and Retrieval System）は非常にシンプルなメッセージ配信システムで、TCP/IP ソケットを使用してテキストのメッセージを送受するシステムです。STARS では一つのサーバプログラムに複数のクライアントプログラムが接続して、テキストメッセージの送受を行います。テキストメッセージの送受だけなのですが、これで結構いろんなシステムへ応用可能なのです。高エネルギー加速器研究機構 放射光研究施設ではビームラインの制御に使用したり、はたまた、ロボットを動かしたり、実験室への入退室管理に使用したりと、結構実用的に使われています。STARS のサーバ自体は Windows、Macintosh、Linux、FreeBSD など、様々な OS で動作しますので、是非試してみてください。

## STARS を使うのに必要な知識

STARS では TCP/IP Socket とテキストベースのメッセージを扱います。つまりこれらをある程度は使える必要があります。

### *TCP/IP Socket*

残念ながら、「TCP/IP Socket って何のこと？」という状態では STARS を使う事はできません。概念程度でいいので、TCP/IP Socket については知っておく必要があります。ホスト名、IP アドレスなどなど、STARS では接続認証の際に使用されます。なお、各種開発言語用の STARS ライブラリを使用すると、TCP/IP Socket を意識する必要は無くなりますが、それでもトラブル対策などには有効な知識となります。

### *プログラミング言語*

STARS にはいくつかの汎用的な STARS Client プログラムがありますが、新しく何らかの機能を実現するためには STARS Client プログラムを作成する必要があります。自分自身のシステムを構築するためには、何らかのプログラミング言語を使えなければなりません。なお、プログラム作成に使用する開発言語は TCP/IP ソケットと文字列の取り扱いが可能なプログラミング言語であれば、基本的に何でも構いません。自分の好きな開発言語を使ったり、状況に応じてプログラミング言語を選んだりすることが可能です。

### *テキストの処理*

STARS ではテキストデータのみによるメッセージの送受を行います。当然数値などのデータもテキストとして送受されます。このため、「数値<=>テキスト」の変換作業や、メッセージ中のパラメータ抽出など頻繁に行う事となります。

## 試しに STARS Server を動かしてみる

詳細な話に入る前に、とりあえず STARS Server を動かして、それに繋いでみましょう。試すには STARS Server のインストールと Telnet クライアントが必要になります。

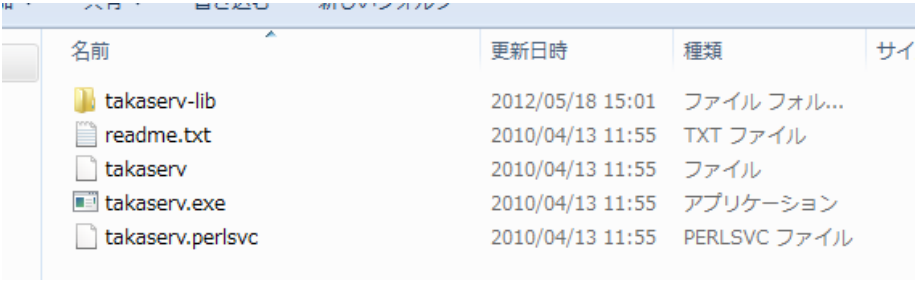
### *STARS Server のダウンロードとインストール*

Web ブラウザを開き http://stars.kek.jp にアクセスしてください。Windows を利用しているのなら、「Download Windows version」を、それ以外なら「Download Unix version」のリンクをクリックしてください。Windows 版と Unix 版の違いは Windows 専用の STARS Client が含まれているか、ソースコード等のデリミタが「CR＋LF」か「LF」となっているか程度で、ほとんどが同様のも

のです。なお、それぞれは圧縮された形式（Windows 版は zip、Unix 版は tgz）となっています。それぞれのダウンロードのページに移動したら、kernel の項目のファイルをダウンロードし、適当なディレクトリに展開してください。

### *STARS Server の前準備*

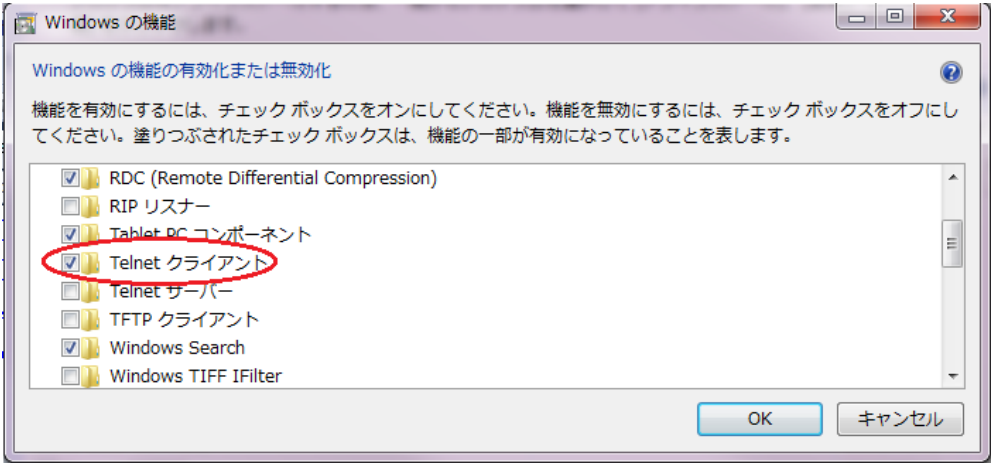
展開されたファイルのうち「takaserv」が STARS Server です。また、「takaserv.exe」は Windows 用の実行形式のファイルです。




はじめに「takaserv-lib」のディレクトリを覗いてみましょう。ここには STARS の設定ファイルや、後で説明するキーワードファイルなどが置いてあります。詳しい説明は後回しにして、とりあえず「term1.key」というファイルを「term2.key」の名前にしてコピー、更に「term3.key」という名前にしてこのディレクトリにコピーしてください。

### *もし、Telnet クライアントが無かったらインストール*

ターミナルやコマンドプロンプトなどから「telnet」と入力して、もし、「そんなソフト入っていない」的な事をパソコンが言うようであれば、Telnet クライアントのインストールが必要です。Windows のエディションによってはデフォルトで Telnet クライアントが有効になっていない場合があります。その時は、「コントロールパネル」=>「プログラムと機能」=>「Windows の機能の有効化または無効化」のようにたどって、「Telnet クライアント」を有効にします。




### *Perl はありますよねぇ・・・*

Windows ではこの「お試し」に実行形式のものを使用するので、使用する OS が Windows の場合は読み飛ばしてください。

Linux などの場合は、ターミナル画面で「perl -v」と入れてみてください。大抵の場合、perl がインストールされているのですが、もし、そうでなければ、インストールが必要です。

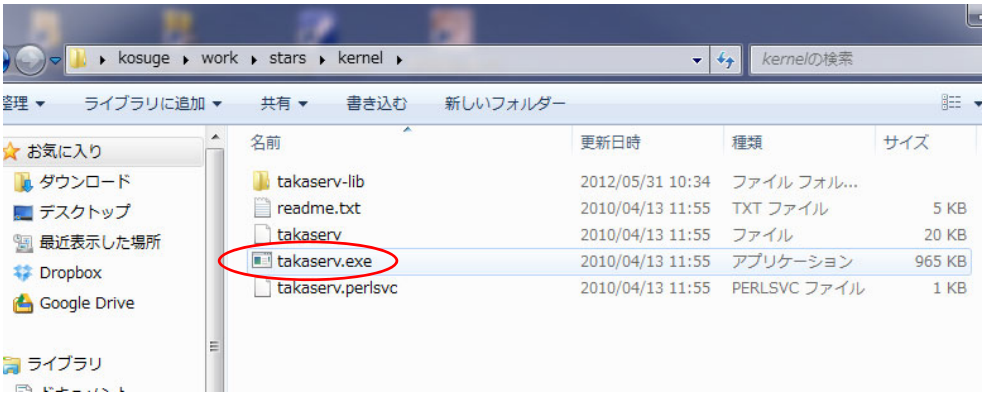
### *STARS Server の起動とクライアントの接続*

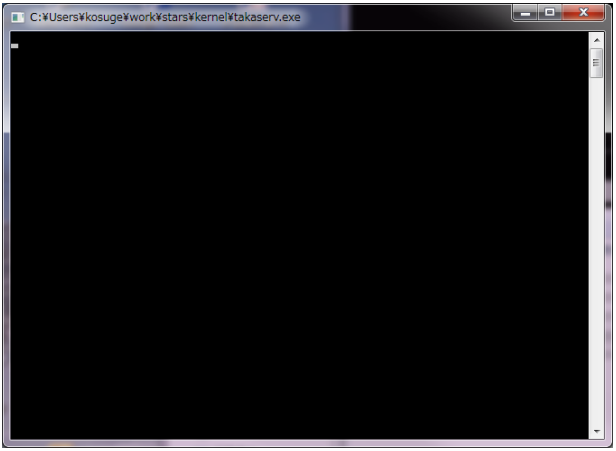
STARS Server のあるディレクトリに移動して Linux 等の場合は

```
perl takaserv
```

のようにして起動してください。

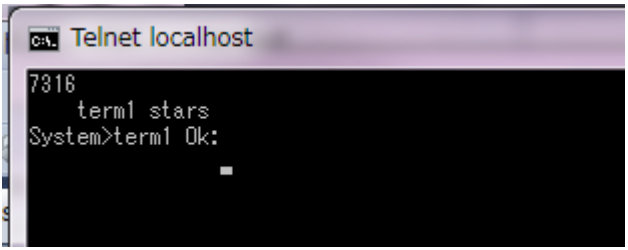
Windows の場合は「takaserv.exe」をダブルクリックで可能です。






この時、上のように何も表示されないコマンドプロンプトのようなウインドウが現れます。これは STARS Server が実行されているウインドウです。ここでは文字を入力してもないも起きません。ただし、Ctrl+C（Ctrl キーを押しながら、C キーを押す）を入力すると STARS Server が停止します。STARS Server が起動したときこのウインドウが邪魔な場合は最小化ボタンをクリックし、ウインドウを最小化してしまいましょう。

次にターミナルやコマンドプロンプトで「telnet localhost 6057」と入力してみてください。STARS Server が数字を返して来ますので、次に「term1 stars」と入力、Enter を押します。「System>term1 Ok:」のようになれば接続に成功です。




なお、Windows の Telnet クライアントでは「Back space」などのコードもそのまま送信されるので、入力を間違えたからといって「Back space」や「Delete」キーを押してしまうと、接続に失敗します。

接続がうまくいったら、ターミナルやコマンドプロンプトを複数起動し、term2、term3 と複数のクライアントから接続してみましょう。

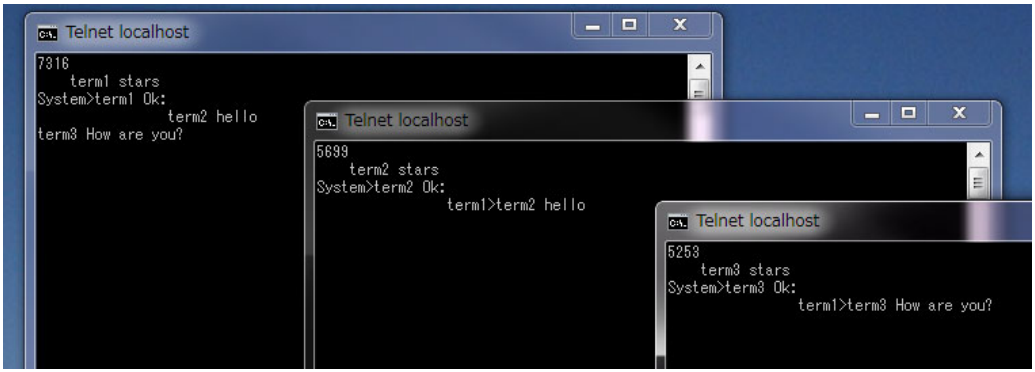
この時「Bad host. hostname」のようなメッセージが出て接続ができない場合は

「takaserv-lib」の下の「allow.cfg」ファイルをテキストエディタなどで開いて「hostname」のホスト名を一行追加してみてください。

### *ノード名、キーワード、メッセージ配信*

STARS では STARS Server に接続してくるそれぞれの STARS Client は識別のためのユニークな「ノード名」を持つ事になっています。Telnet て接続する際に使用した、term1 や term2、term3 はそれぞれノード名にあたります。また、STARS では STARS Client が接続してくる際、簡単なキーワードによる認証が行われます。ここで入力した「stars」がこのキーワードにあたります。キーワードによる認証については後程説明しますのでここでは「そんなもんかぁ」と思っていてください。

さて、それではメッセージの配信をテストしてみましょう。まず、term1 として接続したターミナルに移動して「term2 hello」と入力してみましょう。term2 に「term1>term2 hello」と表示されます。また、term1 から「term3 How are you?」などと入力すると term3 にメッセージが表示されます。つまり、STARS Server は入力された文字列の最初のワードを宛先と認識して、それぞれの STARS Client に配信します。



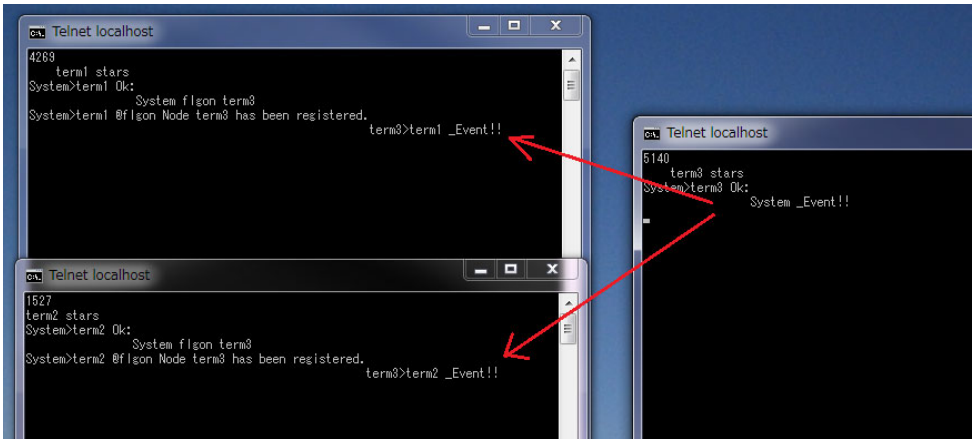

STARS Server から送られてきたメッセージには送り主に関する情報も挿入されているので、送られてきたものがコマンドだとすると、答えをどのノード宛に返せばいいかがわかります。

STARS ではこのようにまるでそれぞれの STARS 送られてきたメッセージに応じて、それぞれの STARS Client が処理を行う事で、制御システム等を構築してゆきます。つまり STARS では、それぞれの STARS Client が、まるで「チャット」でもしているかのようになります。

*メッセージ配信機能*

STARS ではそれぞれの STARS Client はユニークなノード名を持ちますが、実は STARS Server 自体も「System」という名前を持っています。なお、STARS では大文字小文字を区別するので注意が必要です。System を宛先に、たとえば help コマンドを（「System help」）送ると、利用可能なコマンドが返されて来ます。それでは、その中の「flgon」コマンドを試してみましょう。はじめに term1 から「System flgon term3」と入力してみてください。「System>term1 @flgon Node term3 has been registered.」と表示されると思います。そして同様に term2 から「System flgon term3」と入力します。

ここで送った「flgon」コマンドは、「もし、term3 に何かのイベントが発生したら、教えて」という意味です。次に term3 から「System _Event!!」のように System に対してアンダースコアから始まる文字を送ってみてください。「_Event!!」の文字は term1 と term2 にそれぞれ配信されます。STARS ではアンダースコアから始まるメッセージはイベントとして処理され、System に送られてきたイベントのメッセージは予め「flgon」コマンドを送ってきた STARS Client に配信（メッセージ配信機能）されます。



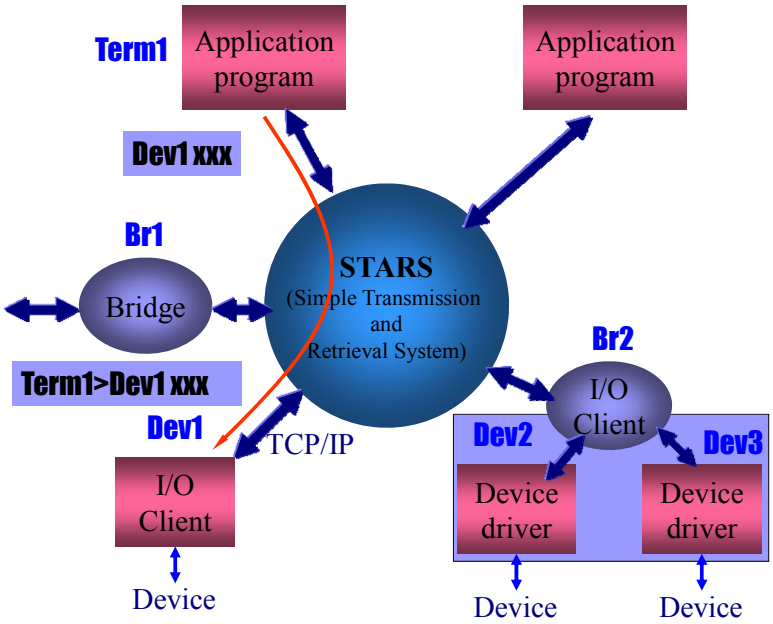

STARS ではこのメッセージ配信機能を利用する事で、非同期型、あるいはイベントドリブン型のシステム構築が可能となります。

# 20　STARS の詳細

## STARS の構成

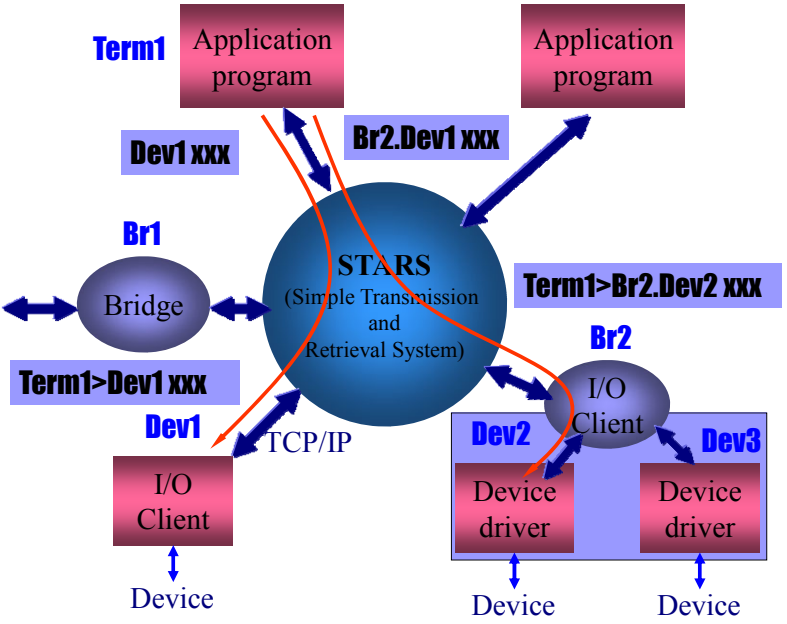
「STARS のとりあえず」では大雑把な説明でしたが、ここからは STARS についてもう少し詳しく説明してゆきます。

STARS はサーバクライアント型のソフトウエアで、1 つの STARS Server に複数の STARS Client が TCP/IP Socket を利用して接続します。各 STARS Client が宛先情報を持ったメッセージを STARS Server に送信すると、STARS Server は宛先として指定された STARS Client にそのメッセージを配信します。なお、前にも説明したように 1 つのサーバに接続された各 STARS Client はユニークなノード名（Node Name）を有する事になっていて、これらの Node Name がメッセージの宛先として利用されます。




これまでは触れませんでしたが、STARS ではノード名に階層化構造を利用する事が可能で、区切り文字としてはピリオド「．」が使用されます。たとえば、「Br2.Dev2」のような宛先は「Br2」という機器の下に「Dev2」が接続されているようにとらえられます。単純には STARS サーバはピリオド以降を無視して、単に「Br2」にメッセージを配信するだけです。その後「Br2」はこの「Br2.Dev2」

という宛先を解析して「Dev2」に指示をださなければなりません。




### *STARS が扱うメッセージ*

STARS において送受されるメッセージはすべてテキストのメッセージで、メッセージのターミネータとしては「LF」（0Ah）が使われます。STARS のメッセージは「送信元」+「>」+「宛先」+ 「スペース」+「コマンド or リプライ or イベント」のように構成されていて、例としては次のようになります。

Term1>Dev1 GetValue

なお、STARS Client からメッセージを送信する際には、メッセージを送信する STARS Client は、「送信元」及び「>」（例では「Term1>」）の部分を省略する事ができます。この時、STARS Server は送信元の STARS Client のノード名を知っているので、宛先の STARS Client に送信するメッセージに「送信元」及び「>」を自動的に付加します。

STARS では以上のようなメッセージの送受を行いますが、更に「コマンド or リプライ or イベント」の部分の開始文字に以下のような規定を設けています。

- 「@」（アットマーク）から始まるもの => コマンドに対するリプライ
- 「_」（アンダースコア）から始まるもの => イベント情報

- それ以外の文字から始まるもの => コマンド

このような規定を設ける事で、STARS では「コマンド及びリプライによる動作」、「イベントドリブン型の動作」の両方とも扱えるようにしています。

## STARS Client 接続時のセキュリティーチェック

STARS では、STARS Client が STARS Server に接続する際、簡単なセキュリティーチェックを行う事で、関係の無い者が勝手に STARS Server に接続する事や、誤って別のサーバに接続してしまう事が無いようにしています。実際のチェックは 3 段階にわかれていて、「1. 予め登録された PC からの接続であるかのチェック」、「2. キーワードによるチェック」、「3. ノード名+ホスト名（IP アドレス）のチェック」となります。また、これらのチェックを行う為の設定ファイル、キーワードファイルは「takserv-lib」のディレクトリに配置されています。

### *予め登録された PC からの接続であるかのチェック*

STARS Server はまず初めに、ホスト名あるいは IP アドレスをチェックして、接続を要求してきた PC が接続可能 PC のリストにあるかどうかをチェックします。もし、その PC が登録されていなければ、STARS Server は「Bad host.」のメッセージをクライアントに送り、TCP/IP Socket の接続を切ります。接続可能 PC のリストは「allow.cfg」になります。以下はデフォルトで用意されている「allow.cfg」の内容です。

```
# Example of allow.cfg
127.0.0.1
localhost
# Allow IP address between 192.168.11.204 - 192.168.11.206 #now
commented
#192.168.11.20[4-6]
# Allow IP address matches 192.168.11. #now commented
#192.168.11.*
```

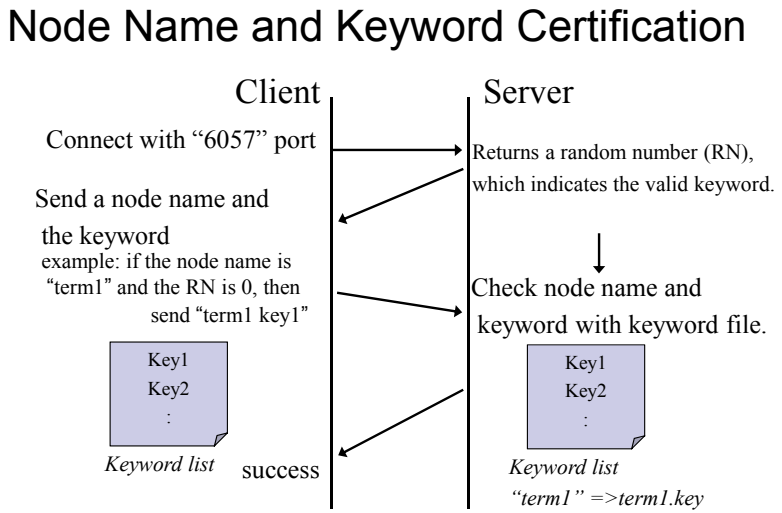
「#」で始まる行はコメントして扱われます。記入例がコメントとして記されているので参考にしてください。「127.0.0.1」と「localhost」は STARS Server が動作する PC 自身に対する記述です。もし、この記述が無いと同一 PC 内からの接続が拒否されます。なお、IP アドレスは「＊」（アスタリスク）などを用いて範囲で指定する事も可能です。

*キーワードによるチェック*

次に STARS Server はキーワードによるセキュリティーチェックを行います。各々の STARS Client は独自のキーワードリストを持つ事になっていて、接続しようとする STARS Client については、そのキーワードリストのコピーを予め STARS Server に登録しておく必要があります。STARS Server に登録するには「ノード名.key」のように STARS Client のノード名をプリフィックスに持ち、拡張子が「.key」のような名前のキーワードリストファイルを、「takaerv-lib」のディレクトリ内に置きます。以下はキーワードリストの例です。

```
abcdefg
hijklmn
1234567
   :
   :
```

このようにキーワードリストファイルは単なるテキストファイルで、一つのキーワード毎に改行されます。




上の図のように STARS Client が STARS サーバに接続してくると、STARS Server は初めにランダムな番号を送信してきます。この番号はキーワードリスト中の「何番目のキーワードを使用するか」を示しているので、STARS Client は自

分のノード名と指定された番号に合致するキーワードを付けて STARS Server に送信します。なお、STARS Server が 0 を返してきた場合は 1 番目のキーワードが使用されます。STARS サーバはこのノード名とキーワードをチェックし、正しければ接続を許可します。

キーワードは 10,000 個まで登録可能ですが、もしキーワードが 10,000 個より少ない場合、たとえば 100 個だとすると 100 の数字が返された場合は 1 番目のキーワードを、101 の場合は 2 番目のキーワードを使用する事になります。更にキーワードを 1 つだけにしてしまうと、常に同じキーワードが使われるようになります。このようにキーワードを 1 つにして使うと、キーワードチェックのプロセスをかなり省略できるので、組込み用の機器などで STARS Client を実現する場合など、プログラミンが容易になります。

実は、「STARS のとりあえず」で Telnet クライアントに入力した「term1 stars」はノード名の後にキーワードを付けて送ったものです。term1.key の内容を確認すると「stars」と 1 行だけ書かれているのがわかります。つまり、常にこの「stars」というキーワードが使用されるようになっています。ただし、この方法は通常のネットワーク上で利用する場合は、セキュリティーホールとなりますので、必ず第 3 段階のチェックである「ノード名＋ホスト名（IP アドレス)」のチェックを組み合わせるようにしましょう。

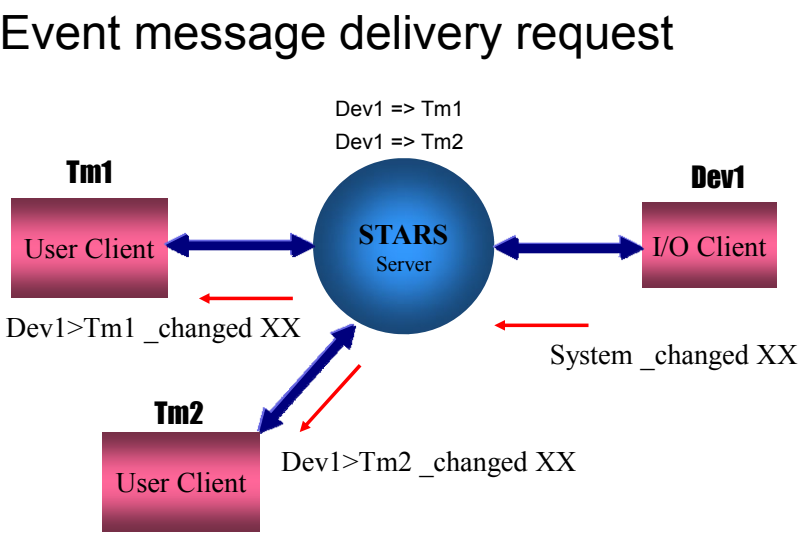
### *ノード名+ホスト名（IP アドレス）のチェック*

STARS の接続に関する第 3 段階のセキュリティーチェックはノード名とクライアントの動作するコンピュータのホスト名(場合によっては IP アドレス) のマッチングによるチェックです。前述のホスト名によるチェックとキーワードによるチェックは必須ですが、ノード名及びホスト名によるチェックの導入はシステムの運用方針により決定する事ができます。STARS Server は「takaserv-lib」のディレクトリ内に「接続を要求してきたクライアントのターミナル名」+.allow （例、ターミナル名 term1 の場合: term1.allow）というマッチングチェック用ファイルがあった場合、ターミナル名とホスト名のマッチングのチェックを行います。たとえばホスト名が「client-host」 というコンピュータ上で動作する「term1」 というターミナル名のクライアントプログラムが接続を試みてきた場合で、「takaserv-lib」内に「term1.allow」 というファイルがあった場合は、このファイルの中に「client-host」 という名前があるかどうかのチェックが行われます。なお、マッ

チングチェック用ファイルは接続可能ホストのリストファイルと同様、改行で区切られたテキストファイルで、テキストエディタ等で編集や登録を行う事が出来ます。

## イベント配信機能

「STARS のとりあえず」でも触れたように、STARS にはイベントメッセージの配信機能があります。イベント配信要求はあらかじめ「flgon」コマンドを STARS Server に送ることで行います。なお、STARS Server 自体も「System」というノード名を持っているので、イベント配信要求は「flgon」コマンドを「System」を宛先として送信することにより行います。イベント配信機能の例の図では「Term1」が「Dev1」に関してのイベント配信要求を STARS Server に送っています(1)。要求の登録が成功すると STARS Server は要求が登録された旨を示すリプライメッセージを「Term1」に送信します(2)。その後は「Dev1」が宛先を「System」としたイベント情報メッセージをサーバに送信すると(3)、イベント情報メッセージは、要求の登録を行った「Term1」に届けられます。なお、STARS サーバは複数の STARS Client からのイベント配信要求を受け付けることが可能で、複数の STARS Client からの要求がある場合にはイベント情報メッセージは要求を登録したそれぞれの STARS Client に届けられます。



## エイリアス機能

STARS Client はそれぞれユニークなノード名を有しますが、STARS Server に別名を登録する事ができます。この機能を利用すると、クライアントの構成が変わった場合などのプログラムの書き換えを最小限に抑える事ができます。なお、別名の情報は「takaserv-lib」内の「aliases.cfg」 ファイルに記録されています。別名の登録等はこのファイルをテキストエディタ等で編集する事により行います。ファイル書き換え後は STARS Client から STARS Server に対して「loadaliases」コマンドを送出する必要があります。

## Debugger

STARS Server にはデバッグを行う為の機能があります。STARS Server にノード名を「Debugger」（一番初めの文字は大文字であることに注意）として接続すると、STARS Server が STARS Client に送信するメッセージを全て受け取る事ができます。この機能を利用すると STARS Client 間の通信をモニターする事が出来るので、デバッグを行う際には非常に有効です。以下は Telnet クライアントを使用して Debugger としてモニタリングを行っている例です。なお、キーワードについては「stars」の 1 つのみを設定しています。

```
%telnet localhost 6057
Trying 127.0.0.1...
Connected to localhost.kek.jp.
Escape character is '^]'.
6296
Debugger stars
System>Debugger Ok:
System>Debugger Ok:
watcher>keyterm07 gettemp
keyterm07>watcher @gettemp 32.5
watcher>gatekeeper01 isClose
gatekeeper01>watcher @isClose 1
watcher>gatekeeper02 isClose
gatekeeper02>watcher @isClose 1
```

## クライアントプログラムの例

ここでは Perl を利用した簡単なクライアントプログラムの例を示します。ノード名を「term1」として、Telnet クライアントの時と同様、キーワードを 1 つだけにしてキーワードチェックのプロセスを簡略化しています。

```
#! /usr/bin/perl
use IO::Socket;
$host='localhost';
$port=6057;
$|=1;

## Open TCP/IP socket
my $server =
  new IO::Socket::INET(PeerAddr=>$host, PeerPort=>$port, Proto=>'tcp')
  or die "Socket: $!¥n";
select($server);$| = 1;select(STDOUT);

## Get random number from TAK server.
$buf=<$server>;

## Send terminal name and keyword.
print $server "term1 kek¥n";

## Expect connection will be accepted.
$buf=<$server>;chop($buf);s/¥r//;
unless($buf =~ /Ok:$/){
close($server);
die "$buf¥n";
}
print "$buf¥n";
print "Please enter destination: ";
$ds=<STDIN>;
chomp($ds);
print "Please enter commands: ";
$cm=<STDIN>;
chomp($cm);

## Send message
print $server "$ds $cm $lp¥n";
```

## STARS インターフェースライブラリ

STARS Client の開発は、TCP/IP Socket 及びテキストを扱うスキルがあれば、誰でも行う事ができます。そして、利用する開発言語及びオペレーティングシステムは任意です。しかし、TCP/IP Socket を Open したりするのも結構面倒です。また、キーワードのチェックのプロセスも実装しようとすると、かなり面倒なものになります。

STARS では STARS Client の開発を行う際に横着ができるよう、「インターフェースライブラリ」を各言語用に用意しています。これら「インターフェースライブラリ」を使用すると、特に TCP/IP Socket を意識することなく STARS Client の開発を進める事が可能です。また、面倒な「接続時のセキュリティーチェック」のプロセスも自動で行われるので、更に STARS Client の開発が容易となります。現在、利用可能なインターフェースライブラリは以下の通りですが、随時追加してゆくようがんばっています。

- Perl：Perl モジュール、様々な OS で利用可
- C：様々な OS で利用可、メッセージの長さに制限あり、Windows では Call back の利用不可
- ActiveX Control：VB6 等で利用
- .NET：C#、Visual Basic .NET 等で利用可、mono を利用すれば Linux 上でも動作
- Java：Windows、Linux、Android etc.

# 30 STARS Perl Interface モジュールと Perl Wizard

Perl でのプログラミングを行う場合、STARS Perl Interface モジュールや Perl Wizard を使用すると STARS Client の開発効率が飛躍的にアップします。当然ですが、Perl を使用してプログラミングを行う場合は Perl が必要となります。Linux などの OS では既に Perl が利用可能になっているのが殆どですが、Windows では active Perl（http://activestate.com）などのインストールが必要です。

## STARS Perl Interface モジュール

STARS Perl Interface モジュールは STARS のサイトからダウンロード可能です、「perllib」をダウンロードし適当なディレクトリに展開してください。様々な STARS Client を開発する事を考えると、たとえば「c:¥stars」などを作成して、その配下に perllib 等を展開するのがお勧めです。

STARS Perl Interface モジュールを利用すると Perl での STARS クライアントの開発効率が向上します。STARS Perl Interface モジュールを使用するためには、はじめに「stars.pm」 をモジュールのサーチパスにコピーするか、クライアントプログラムのカレントディレクトリにコピーします。あるいは、「stars.pm」 を適当なディレクトリにコピーした上でクライアントプログラム起動の際に「-I directory」 オプションを指定したり、プログラム内で「use lib;」 を使用してもかまいません。次にクライアントプログラムと同一のディレクトリにキーワードリストファイル（STARS Server の takaserv-lib 内と同じもの） をコピー、クライアントプログラム内に「use stars;」 の一文を付け加える事で STARS Perl Interface モジュールが利用可能となります。以下は実際に STARS Perl Interface モジュールを使用したクライアントプログラムの抜粋です。（左の数字は行番号）

```
1   use stars;
2   $svr = stars->new('term1') or die;
3   print $svr->act('System hello')."¥n";
```

1 行目では STARS Perl Interface モジュールの利用を宣言、2 行目で STARS オブジェクトを作成、STARS Server に接続を行っています。この時、2 つめ及び 3 つめの引数は省略可能で、それぞれ「localhost」、「ノード名」+「.key」がデフ

ォルトとなります。ここでキーワードチェックに関する手順は STARS Perl Interface モジュールが自動的に行ってくれます。3 行目では STARS Server に hello コマンドを送出した後、Reply message を受信、表示を行っています。

### *new*

*Usage: $object = stars->new(node$object = stars->new(nodename, [serverhost], [serverport], [keyfile]); name, [serverhost], [serverport], [keyfile]);*

新たに STARS オブジェクトを作成し、STARS Server とのコネクションを確立します。成功するとオブジェクトのリファレンスを返します。

nodename：STARS で使用する自ノード名を指定します。

serverhost：STARS Server が動作しているホスト名を指定します。省略すると 'localhost'が使用されます。

serverport：使用する TCP/IP ソケットの Port 番号を指定します。省略すると 6057 が使用されます。

keyfile：認証に使用するキーワードリストのファイル名を指定します。省略すると nodename+'.key'がファイル名として使用されます。

### *act*

*Usage: val/list = $object->act(message);*

メッセージを送信し、その後自ノードへのメッセージを受け取ります。スカラーコンテキストでは STARS Server からのメッセージがデリミタ(LF)を取り除いたそのままの文字列として返されます。リストコンテキストでは(TermFrom, TermTo, Message)のように分割されて返されます。また、タイムアウト(10s)時には " または(", ", ")が、エラー時には undef または空のリストが返されます。

message：STARS 規定のメッセージ(TermFrom>TermTo Message または TermTo Message)を指定します。なおデリミタ(LF)は必要ありません。

### *gethandle*

*Usage: $fh = $object->gethandle();*

STARS インターフェースオブジェクトの利用しているファイルハンドルを返し

ます。

### *Send*

*Usage: $object->Send(message [, termto]);*

メッセージの送信行います。

message：送信するメッセージを指定します。デリミタ(LF)は stars.pm が自動的に付加します。

termto：termto を指定すると送信メッセージの先頭に termto とスペースが付加されます。

### *Read*

*Usage: val/list = $object->read();*

メッセージの受信を行います。スカラーコンテキストでは STARS SERVER からのメッセージがデリミタ(LF)を取り除いたそのままの文字列として返されます。リストコンテキストでは(TermFrom, TermTo, Message)のように分割されて返されます。受信メッセージがない場合は"または(", ", ")が、接続が切れてしまったようなエラー時には undef または空のリストが返されます。

### *addcallback*

*Usage: $object->addcallback(subroutine, [filehandle], [mode]);*

STARS インターフェースを Callback モードで利用するときに使います。addcallback メソッドを使ってファイルハンドルを設定したあと Mainloop を実行すると、入力が発生するたびに subroutine で設定したサブルーチンが自動的に呼び出されます。

subroutine：入力が発生したときに呼び出されるサブルーチンへのリファレンスを指定します。

filehandle：call back へ登録するファイルハンドルを指定します。省略すると object のファイルハンドルが使用されます。

mode：call back を呼び出す際に渡すメッセージ解析のモードを設定します。モードには'Stars', 'Lf', 'Direct', 'Detect'の 4 つのモードがあり省略すると

'Stars'が使用されます。'Stars'モードの場合、callback へのメッセージは(TermFrom, TermTo, Message)のようなリストを引数として渡されます。'Lf'モードの場合はデリミタが削除された状態のメッセージが引数として渡されます。'Direct'モードではデリミタは一切関係なく受け取ったデータがそのまま引数として渡されます。また、'Detect'モードでは読み込みは行われず、call back のみが行われます。

### *removecallback*

*Usage: $object->removecallback(filehandle);*

addcallback で設定したファイルファンドルを Callback から削除します。
filehandle：Callback から削除するファイルハンドルを指定します。

### *Mainloop*

*Usage: stars->Mainloop( [ subroutine, timeout ] );*

callback モードを実行します。また timeout を設定することで、一定時間後に処理を戻すことが出来ます。入力ハンドルに接続が切れた場合などのエラーが発生すると Mainloop から処理が戻されます。

subroutine：timeout 及び subroutine を設定すると一定間隔で subroutine を実行できます。subroutine には一定時間間隔で呼び出すサブルーチンへのリファレンスを指定します。

timeout：subroutine を呼び出す間隔を指定します。単位は m second です。また、変数へのレファレンスを指定する事により、プログラム実行中ダイナミックに timeout 値を変更することが可能となります。なお、timeout 値に負の値を設定するとタイムアウトは発生しなくなります。

### *Sleep*

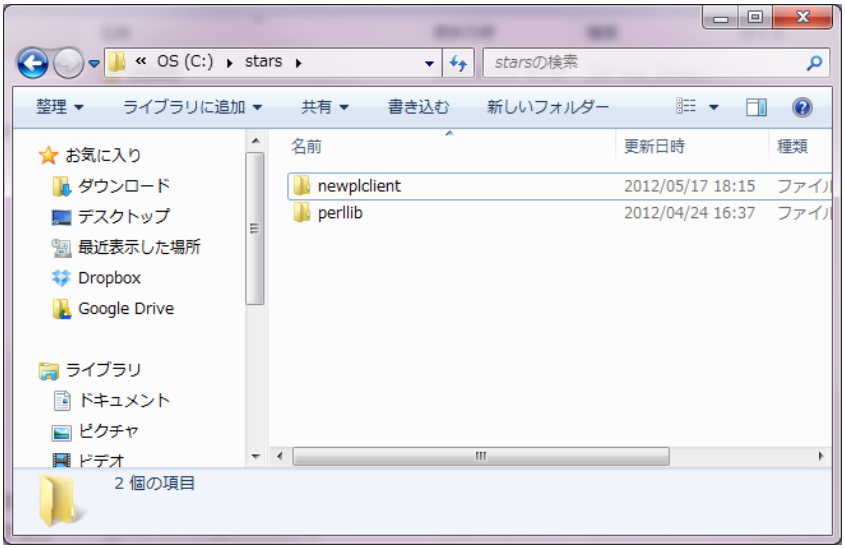
*Usage: $object->sleep(mSec);*

処理を mSec だけ中断します。

mSec：mSec で指定した時間だけ処理を停止することが出来ます。単位は m second です。

## Perl Wizard

Perl Wizard を使用すると Perl を使用した STARS Client の開発が容易に行えます。Wizard を実行するとキーワードリストファイルの自動生成等が自動的に行われ、すぐに STARS Server への接続が可能な STARS Client のテンプレートが作成されます。Perl Wizard は STARS のサイトからダウンロード可能で、アーカイブは「newplclient」となります。ダウンロード後は perl モジュール同様、適当なディレクトリに展開してください。なお、展開された Wizard のディレクトリは（newplclient）STARS Perl Interface モジュールのディレクトリと同じディレクトリになければなりません。（前述の例ですと c:¥stars の下）




newplclient は perllib のディレクトリがあるディレクトリと同じ場所へ

### *Wizard の実行*

ターミナルやコマンドプロンプトを開き、newplclient のディレクトリまで「cd」で移動して「perl newplclient.pl」と入力してください。

```
c:¥stars>cd newplclient
c:¥stars¥newplclient>perl newclient.pl
Make a new Stars client program in Perl.
Please enter client name. (null = cancel) >
```

すると作成する STARS Client のノード名を聞いてきますので、適宜入力します。ここでは「testdev」という名前を例に使って説明します。

```
Please enter client name. (null = cancel) >testdev
```

デフォルトの STARS Server のホスト名、作成するディレクトリについて聞いてくるので、入力を行います。

```
Please enter stars server. (null = cancel) >localhost
Please enter directory for testdev. (null = cancel) >c:¥stars¥testdev
Create c:¥stars¥testdev/stars.pm.
Create c:¥stars¥testdev/testdev.
Create key > c:¥stars¥testdev/testdev.key.
Done.
Please send "c:¥stars¥testdev/testdev.key" to localhost
  with ftp (asc mode).
Hit Enter key.
```

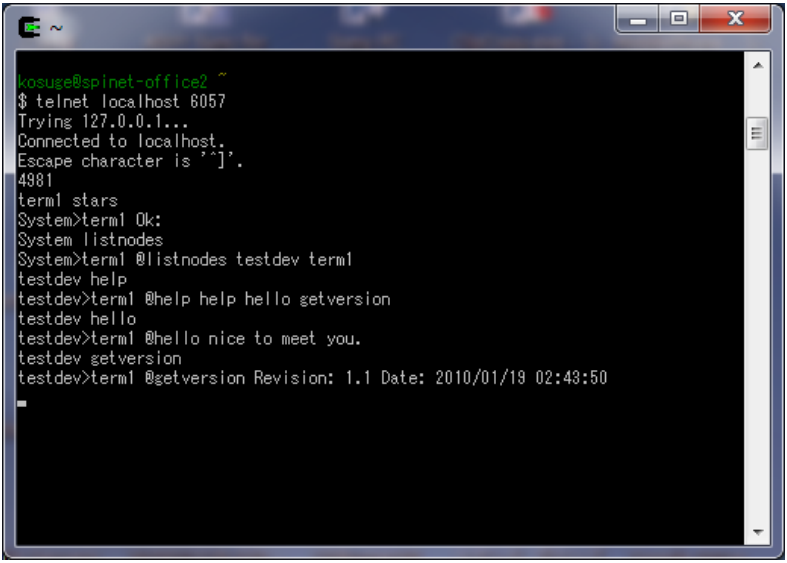
すると、自動的にディレクトリが作成され、キーワードリストファイルも生成されます。その後、このキーワードリストファイルを STARS Server の「takaserv-lib」の下にコピーしてください。

### *作成されたプログラムの実行*

Wizard で作成された STARS Client は既に STARS Server と接続可能です。作成されたプログラムのディレクトリに移動して、実行してみてください。

```
c:¥stars>cd ¥stars¥testdev
c:¥stars¥testdev>perl testdev
```

これで STARS Server が動作していればプログラムが実行されるはずです。Telnet クライアントで STARS Server に接続し、状況を確かめてみましょう。




ここでは Cygwin の Telnet クライアントを使用しています。System に listnodeds コマンドを送信すると、testdev が存在している事がわかります。また、既にデフォルトで help、hello、getversion などのコマンドが準備されているので、testdev にこれらのコマンドを送信すると上のような答えが返ってきます。

### *プログラムの編集*

ここでは今回作成した「testdev」を例に示します。編集にはテキストエディタなどを使用します。

```
#! /usr/bin/perl
↓
use strict;↓
use Getopt::Long;↓
use stars;↓
#############################################################################↓
# testdev↓
$::Version  = '$Revision: 1.1 $';↓
$::Version .= '$Date: 2010/01/19 02:43:50 $';↓
#↓
$::Version =~ s/¥$//g;↓
#############################################################################↓
## ToDo: Set parameters here.↓
$::NodeName   = 'testdev';        #Default node name.↓
$::Server     = 'localhost';        #Default stars server.↓
#############################################################################↓
$::Debug      = ''; #This variable is used for debug mode.↓
                    # You can use like..↓
                    # if($::Debug){blha-blah-blah;}↓
↓
```

Wizard を実行すると以上のように STARS モジュールの宣言などが自動で作成されます。

また、以下の 32 行目では STARS Server へのコネクションが、自動で行われるようになっています。35 行目では STARS Server からメッセージが届いた際に呼び出されるサブルーチンを登録しており、STARS からメッセージが届くと「handler」というサブルーチンが自動で呼び出されます。

```
29 ↓
30 ## Open Stars server. $::tak is a Stars object.↓
31 $::tak = stars->new($::NodeName, $::Server)↓
32 →   or die "Could not connect Stars server";↓
33 ↓
34 ↓
35 $::tak->addcallback(¥&handler);↓
36 stars->Mainloop();↓
37 ↓
38 exit(1);↓
```

なお、呼び出される handler サブルーチンでは if、elsif などを使用し、コマンドを解析し、それに応じた処理を行うようにします。

```
70 ## Handle received messges ##↓
71 →   if($mess eq 'hello'){↓
72 →   →   $rt="nice to meet you.";↓
73 →   }elsif($mess eq 'help'){↓
74 →   →   $rt="help hello getversion";↓
75 →   }elsif($mess eq 'getversion'){↓
76 →   →   $rt=$::Version;↓
77 →   }elsif($mess =~ /^[_@]/){↓
78 →   →   return;↓
79 →   }else{↓
80 →   →   $::Error = "Bad command or parameter.";↓
81 →   →   $rt = '';↓
82 →   }↓
```

それではここで「GetValue」コマンドを実装してみます。はじめにテキストの後ろの方に GetValue が届いたと時に呼び出される GenerateData サブルーチンを追加します。

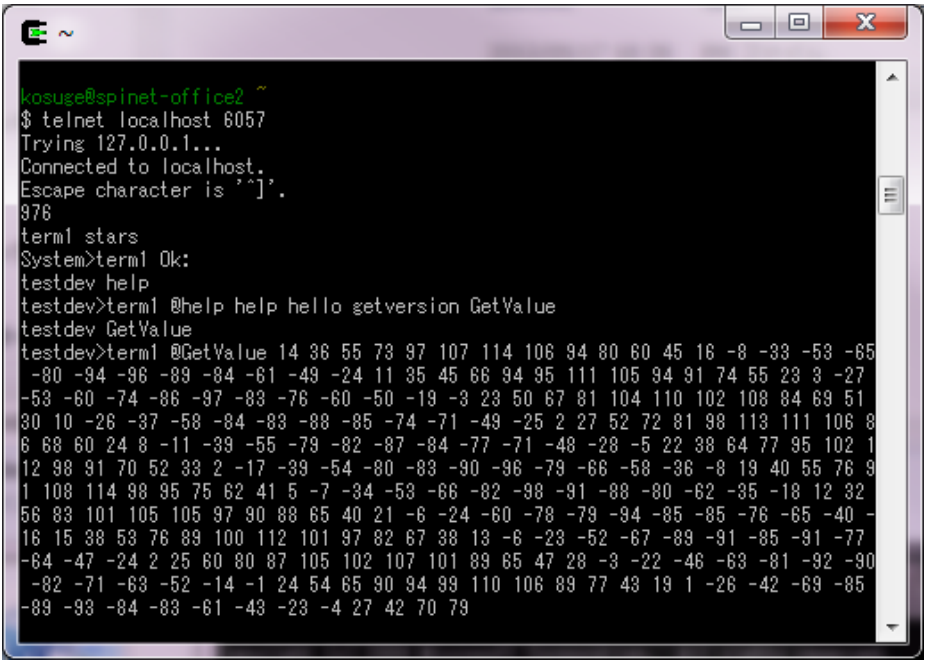
```
sub GenerateData{
	my $lp;
	my @result;
	for($lp = 0; $lp < 256; $lp++){
		$result[$lp] = int(sin($lp/4)*100 + rand(15));
	}
	return(join(" ", @result));
}
```

このサブルーチンは、呼び出されると疑似的なデータを作成し（サイン関数にノイズを付加して 256 個のデータを返す）、文字列として戻します。次に、handler 内のコマンド処理部分を編集し、「GetValue」コマンドを受け取ったらサブルーチン「GenerateData」を呼び出すようにします。

```
## Handle received messges ##
	if($mess eq 'hello'){
		$rt="nice to meet you.";
	}elsif($mess eq 'help'){
		$rt="help hello getversion GetValue";
	}elsif($mess eq 'GetValue'){
		$rt=GenerateData();
	}elsif($mess eq 'getversion'){
		$rt=$::Version;
	}elsif($mess =~ /^[_@]/){
		return;
	}else{
		$::Error = "Bad command or parameter.";
		$rt = '';
	}
```

なお、コマンドが増えたので、help の時に返すメッセージに「GetValue」を追加するのを忘れずに、変更を保存し、testdev を再起動してみましょう。

次に Term1 からコマンドを送ってみます。次のスクリーンショットのようになれば「GetValue」コマンドの実装は成功です。




以上のように STARS Perl Interface モジュールと Perl Wizard を使用すると、簡単に STARS Client が作成できます。また、この時 TCP/IP ソケットやキーワードチェックに関するプロセスと特に考える必要がない事がわかるとおもいます。つまり、とりあえず Perl さえ知っていれば、STARS Client が作成可能になるという事です。Perl が好きな方は、これらの機能を使って、どんどんプログラムを開発しましょう。

# 40 STARS .NET Interface とテンプレート

TARS では.NET フレームワークを用いて STARS Client を作成するためのライブラリ「STARS .NET Interface Library」を用意しています。また、Visual Studio と C#あるいは Visual Basic を使って GUI 開発を行う為のテンプレートもそれぞれ用意しています。

.NET と言うと、Windows だけと思われがちですが、mono（http://www.mono-project.com）を使用すると、.NET 用に開発されたプログラムを Linux や Macintosh 上で動作させる事が出来ます。

ここでは Visual Studio 2010 での利用を例として使用法を説明しますが、Free の開発環境としては Visual Basic 2010 Express や Visual C# 2010 Express なども利用可能で、基本的には同様の使い方となります。なお、現在、STARS のサイトからダウンロード出来るのは Visual Studio 2005 を用いて開発されたものですが、Visual Studio 2010 でも利用可能です。

## STARS .NET Interface Library

本ライブラリは.NET フレームワークを用いたプログラミングに使用するライブラリです。本ライブラリを使用すると、TCP/IP Socket や STARS Server との接続時に行われるキーワードチェックのプロセスを意識することなく STARS Client を開発する事が出来ます。なお、C#や Visual Basic .NET で GUI を開発する場合はテンプレートの利用をお勧めします。その場合は本ライブラリのダウンロードや設定を行う必要はありません。

### *ダウンロードと展開*

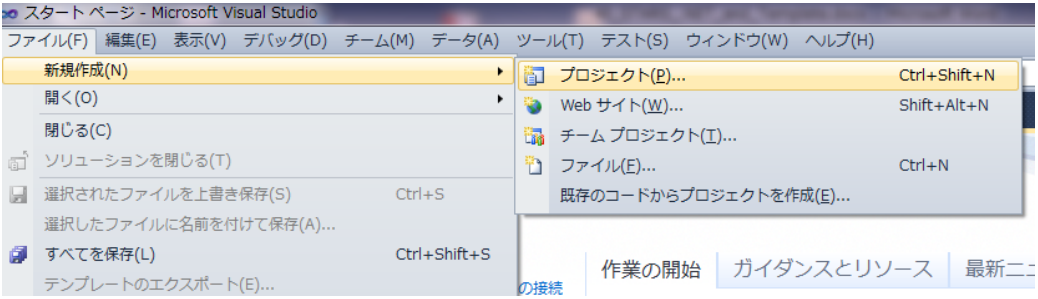
本ライブラリは STARS のダウンロードページから入手してください。ダウンロードするアーカイブの名前は starsinterface.NET_lib です。

ダウンロード後は適当なディレクトリに展開してください。StarsInterface.dll が STARS との通信を行う為のもので、StarsInterfaceWinForm.dll は GUI（Windows Form を使用する場合）を作成する為のものです。また、html ファイルはそれぞれこれらライブラリの使用法です。

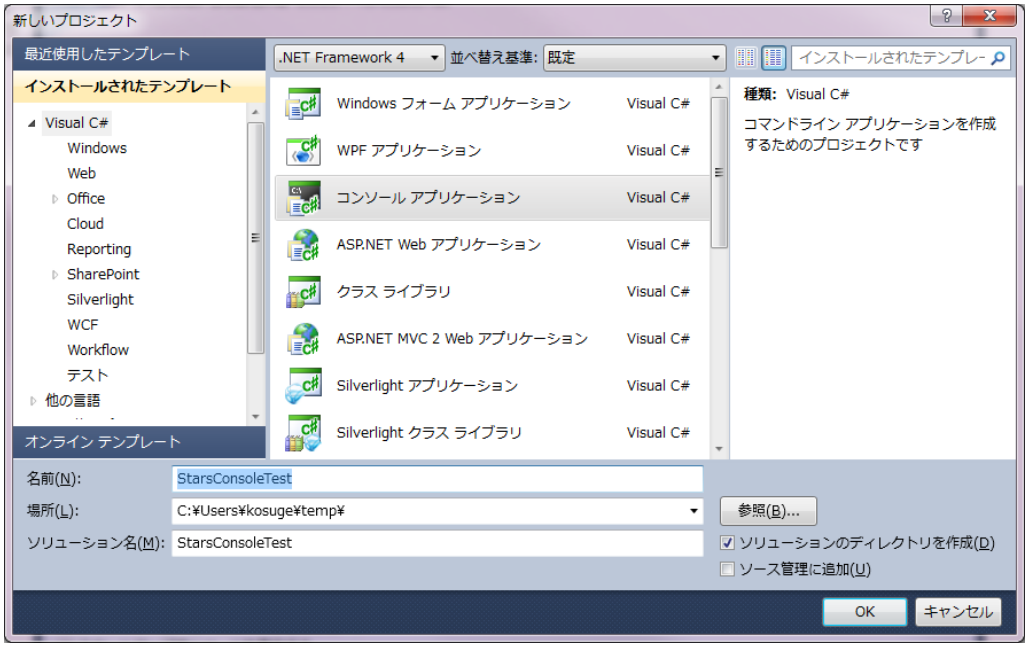
### *プロジェクトの作成*

まず、Visual Studio で新規プロジェクトを作成します。ここでは C#を例に説明

します。Visual Studio を開いたら、「ファイル」=>「新規作成」=>「プロジェクト」を選びます。




次にアプリケーションの種類を選びます。

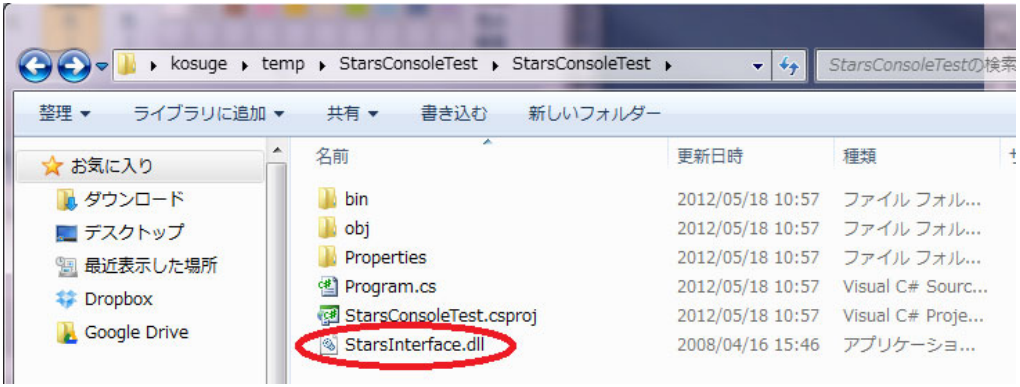



今回の例ではコンソールアプリケーションを作成します。アプリケーションの名前を入れて（ここでは StarsConsolTest としました。）OK をクリックします。

### *ライブラリのコピー*

StarsInterface.dll と StarsInterfaceWinForm.dll はどこか決められた場所に置いてもいいのですが、今回はプロジェクトの中に入れてしまいます。ここではコンソールプログラムですので、StarsInterface.dll のみを、StarsConsoleTest¥StarsConsoleTest の下にコピーします。一度 Visual Studio を

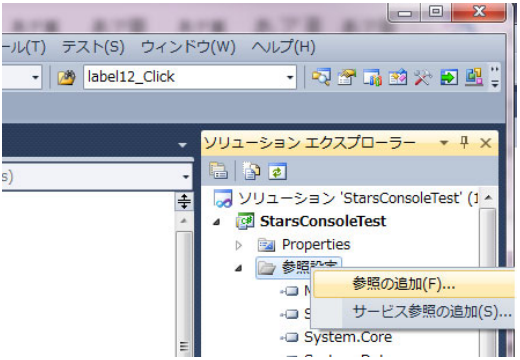
停止してから、コピーを行います。なお、StarsInterfaceWinForm.dll を使用する際も同様の手順でファイルのコピー、次に説明する参照の設定を行います。



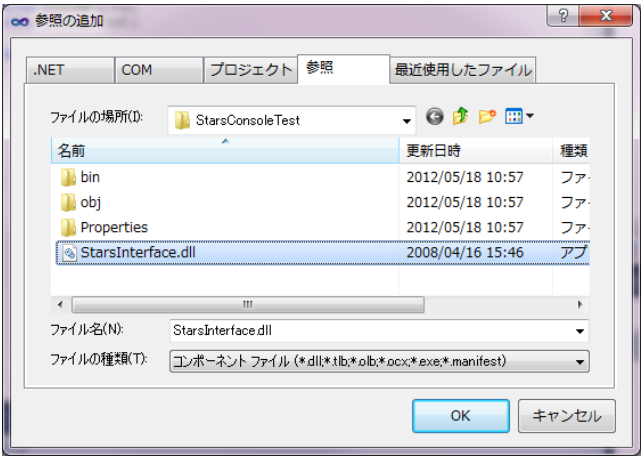

***参照の設定***

次に Visual Studio をもう一度立ち上げ参照の設定を行います。

「ソリューションエクスプローラー」=>「参照設定」=>右クリック=>「参照の追加」を選びます。




参照の追加ウインドウが現れたら、参照タブを選択し、StarsInterface.dll を選び OK をクリックします。




次に Program.cs を編集し、STARS Server と通信するためのコードを入れます。

Program.cs

StarsConsoleTest.Program

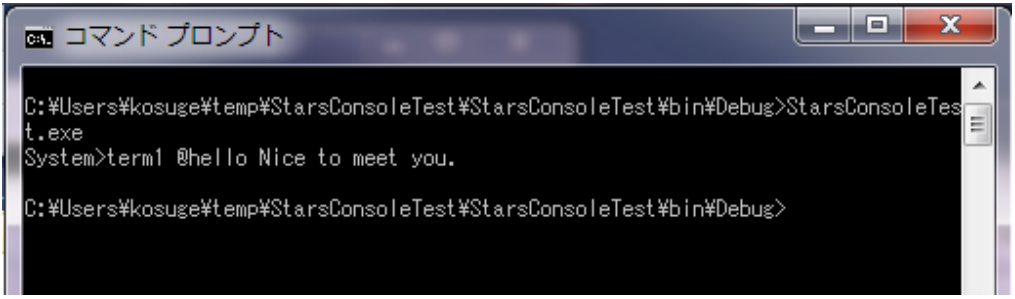
Main(string[] args)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using STARS;

namespace StarsConsoleTest
{
    class Program
    {
        static void Main(string[] args)
        {
            //STARSのオブジェクトを作成。
            StarsInterface stars = new StarsInterface("term1", "localhost");

            //STARS Serverに接続。
            stars.Connect();

            //STARS Serverにhelloコマンドを送信。
            stars.Send("System", "hello");

            //STARS Serverからのリプライを受信。
            StarsMessage rcvMess = stars.Receive();
            Console.WriteLine(rcvMess.allMessage);

            stars.Disconnect();

        }
    }
}
```

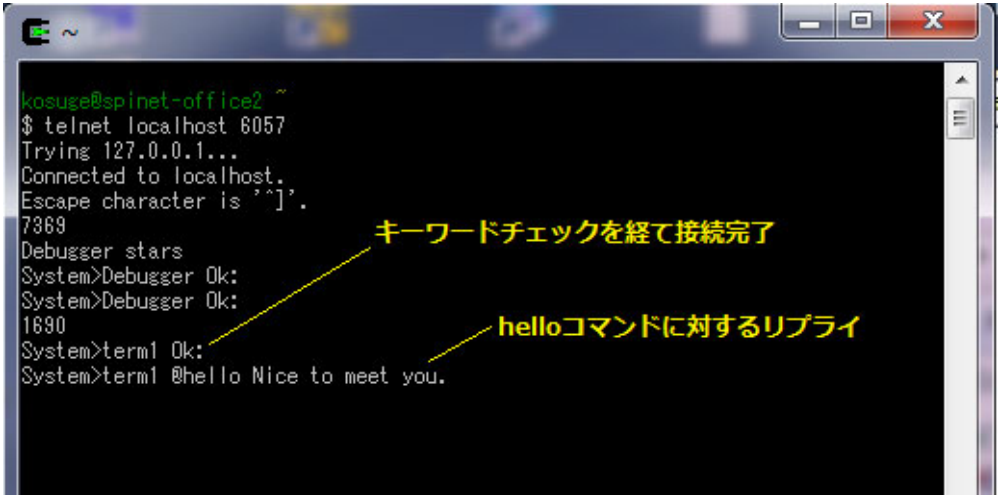
上記のコードはエラー処理をテスト的なものでエラー処理を一切行っていません。実用のものは必ずエラー処理（try、catch 等の利用）を行うようにしてくださ

い。ビルドを行い、成功したら試しに動かしてみます。とりあえずはターゲットを Debug とします。

コマンドプロンプトを開き StarsConsoleTest¥StarsConsoleTest¥bin¥Debug のディレクトリに移動します。また、このディレクトリに STARS Server の「takaserv-lib」にある、term1.key をコピーしておいてください。このクライアントは term1 として STARS Server に接続します。STARS Server 等の準備が出来たら、StarsConsoleTest.exe と入力します。




この例では、初めに STARS サーバに接続、次に hello コマンドを STARS Server に送り、リプライを受け取ってコンソールに表示、その後 STARS Server と切断しています。なお、この例の STARS Client の実行の様子を Debugger 機能でモニタした結果が次のようになります。なお、表示には Cygwin の Telnet クライアントを使用しています。



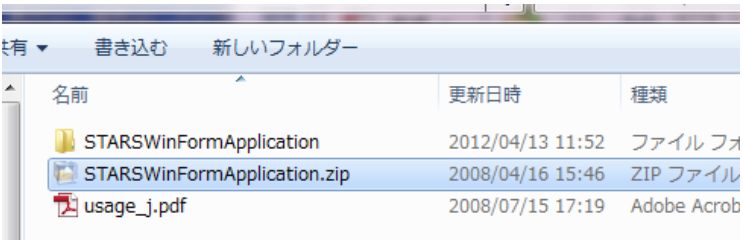
ここでは代表的な STARS .NET Interface Library のメソッドを紹介しましたが、このほかにもコールバック的な動作を行う為のメソッドなどもあります。詳しくは付録のマニュアル等を参照してください。

## テンプレート

STARS Visual Studio 用テンプレートを利用すると C#や Visual Basic を使った GUI STARS Client が容易に作成可能となります。アーカイブ名は winformcs_template が C#用、winformvb_template が Visual Basic 用で、どちらも STARS のダウンロードサイト（Windows 用）で入手可能です。

### *展開と設定*

C#用、Visual Basic 用とも ZIP 形式で圧縮されていますので、適当なディレクトリに展開します。



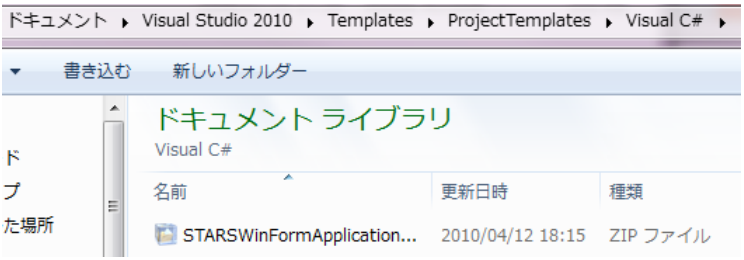

すると、更に STARSWinFormApplication.zip が現れますが、このファイルは展開せずに Visual Studio の Project Template に移動してください。一般的にユーザのホームディレクトリの

「Documents¥Visual Studio 2010¥Templates¥ProjectTemplates¥Visual C#」

が C#用、

「Documents¥Visual Studio 2010¥Templates¥ProjectTemplates¥Visual Basic」

が Visual Basic 用の置き場所となります。

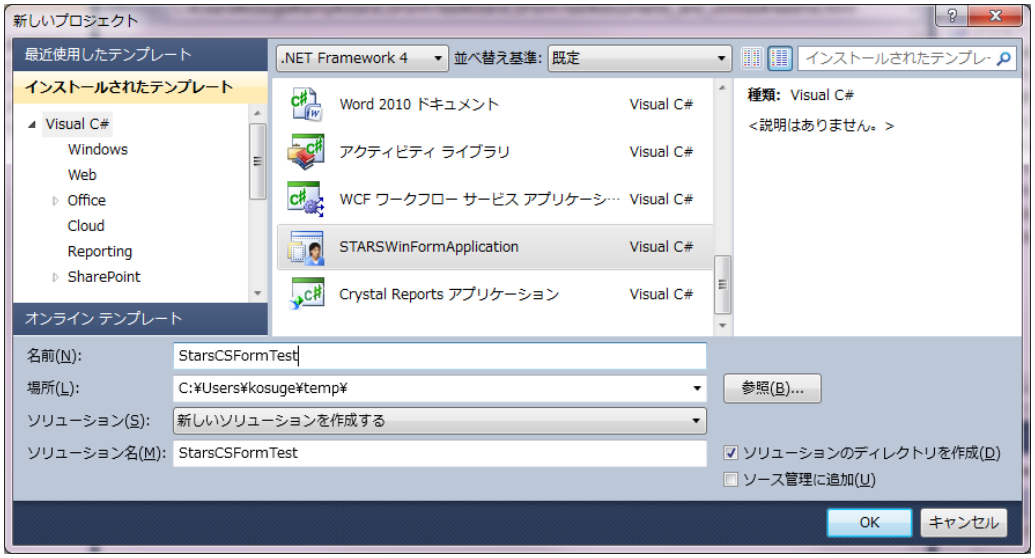


## テンプレートを使用した C#での開発の例

ここではテンプレートを使用して C#での GUI STARS Client を作ってみます。なお、テストを行う為に予め STARS Server を起動しておきます。また、ここでは「STARS Perl Interface モジュールと Perl Wizard」で作成した testdev もテスト用に起動しておくものとします。

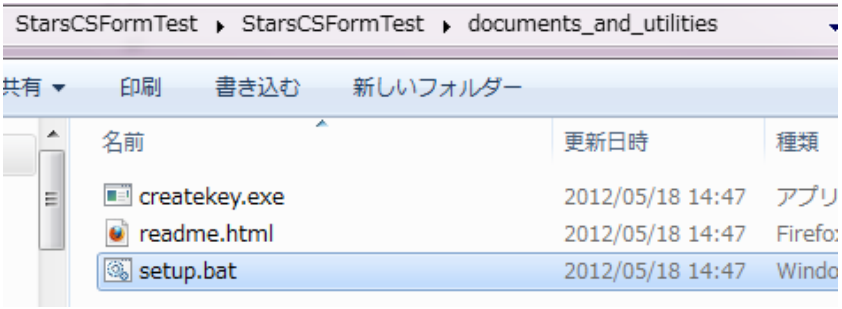
### *新規プロジェクトの作成*

Visual Studio を起動し、「ファイル」=>「新規作成」=>「プロジェクト」を選びます。C#用テンプレートが正しくインストールされていれば、「新しいプロジェクト」ウインドウが開き Visual C#を選ぶと、STARSWinFormApplication が現れますので、それを選択し、プロジェクトの名前を適当に付けて、OK を押します。なお、ここで付けたプロジェクトの名前がデフォルトのノード名として使われます。なお、”-“ハイフンを使用すると”_”アンダースコアに変換されますので注意が必要です。

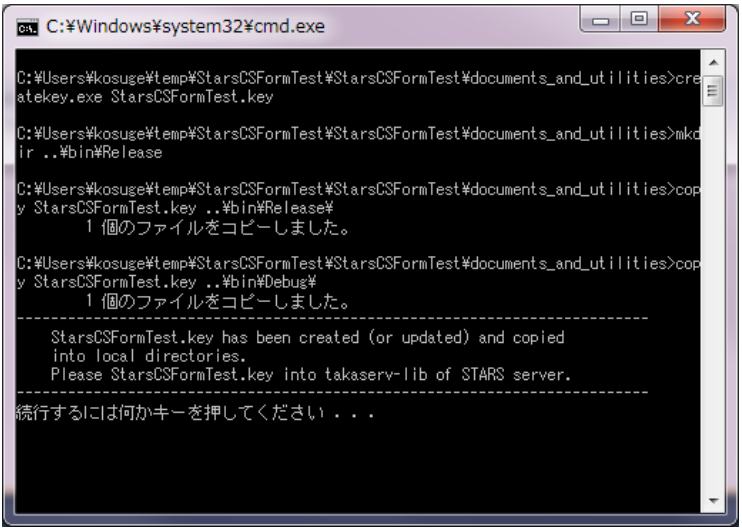



すると、次のような説明が表示されますので、指示に従いバッチファイルを実行します。

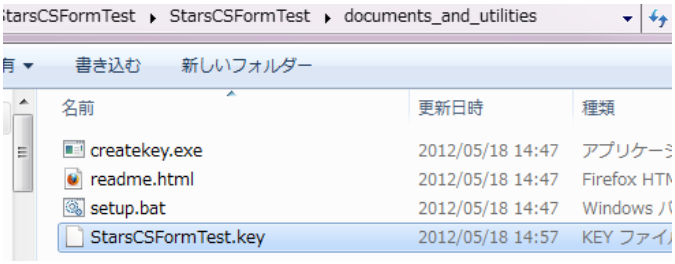

今回作成したプロジェクトの documents_and_utilities ディレクトリをひらき、csetup.bat をダブルクリックします。



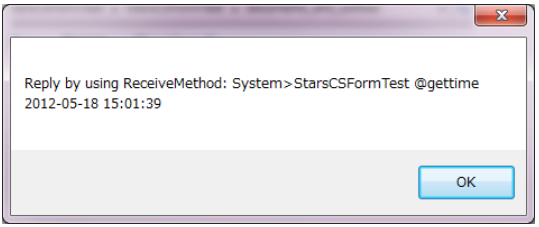

すると、



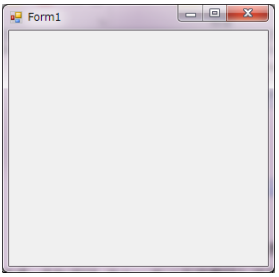

のようにして、キーワードリストファイルが作成されますので、このキーワードファイルを STARS Server の「takaserv-lib」の下にコピーします。




ここまで行うと、既に STARS Server に接続可能な GUI STARS Client が出来上がっています。それでは、Debug モードで実行してみましょう。まず、次のような画面が現れます。




これはメインフォームを表示する前に STARS Server に gettime コマンドを送ってから、リプライを受け取り、メッセージボックスで表示を行うようにしているためです。そのご、OK をクリックすると、そのご、何も配置されていないフォームが表示されます。

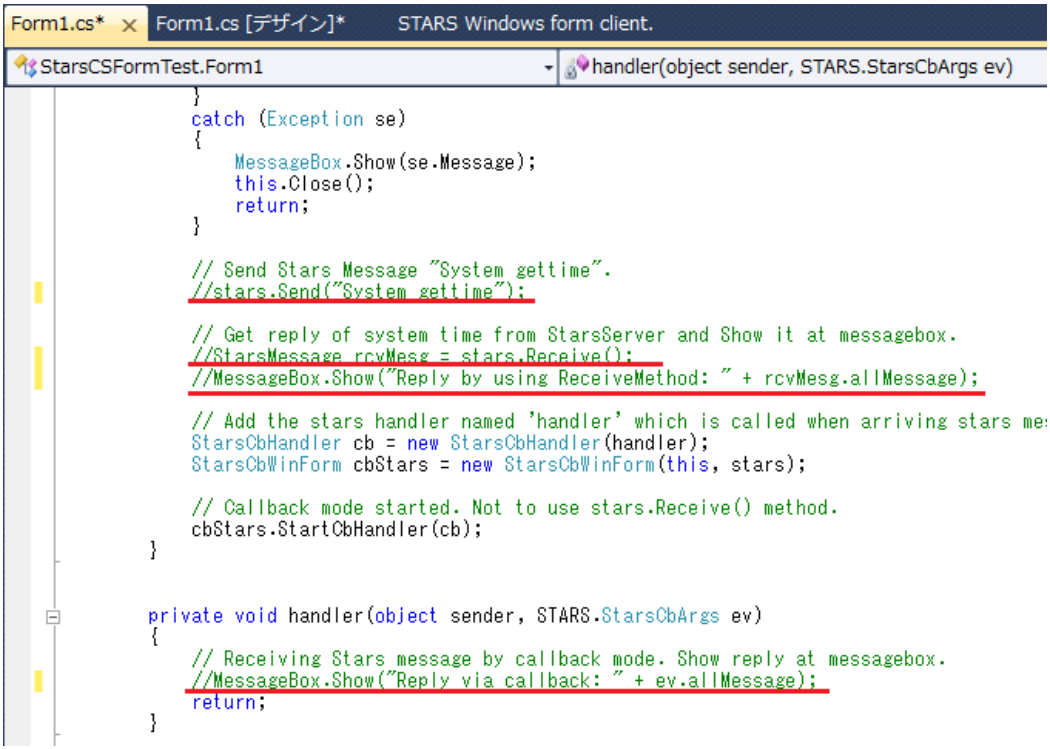



このように、テンプレートを使用すると初めから STARS Server に接続可能な GUI プログラムが自動的に用意されます。

*プログラムの編集*

ここではこのテンプレートを利用して、perl で書かれた STARS Client の testdev からデータを取得してみます。

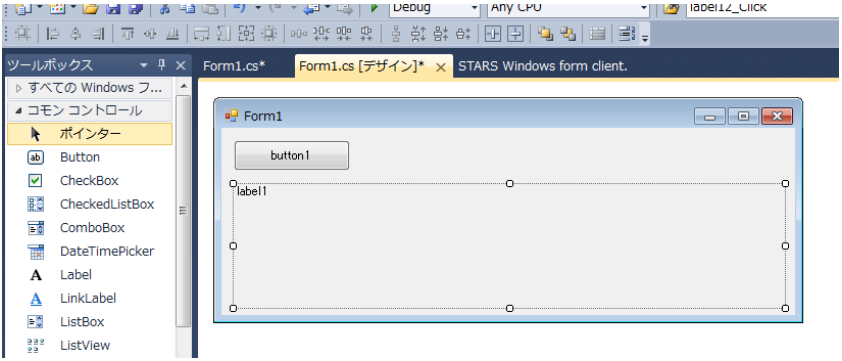
まず、Form1.cs のソースコードを開き、数行をコメントアウトします。




コメントアウトした最初の 3 行が Form1 が表示される前に STARS Server に gettime コマンドを送信、リプライを受け取ったのちメッセージボックスで表示を行っている部分です。

ここで、StartCbHandler メソッドが実行された後は、STARS Server からメッセージが届くたびに handler が呼び出されるようになります。なお、StartCbHandler メソッドが一度でも実行された後は、STARS の Receive メソッドは利用できません。

コメントアウトした 4 行目は STARS Server からメッセージが届いたらそれをメッセージボックスで表示するという部分です。次にデザイン画面で Button と Label を配置します。




ここで、それぞれの名前のうち Button が button1、Label が label1 であり、label1 の AutoSize を False に設定、button1 の Text を「GetValue」のようにする事とします。

そして、button1 をデザイン画面でダブルクリックすると、自動的に button1_click メソッドが作成されるので、ここにボタンが押されたら、testdev に対して GetValue コマンドを発給するよう、仕掛けをいれます。

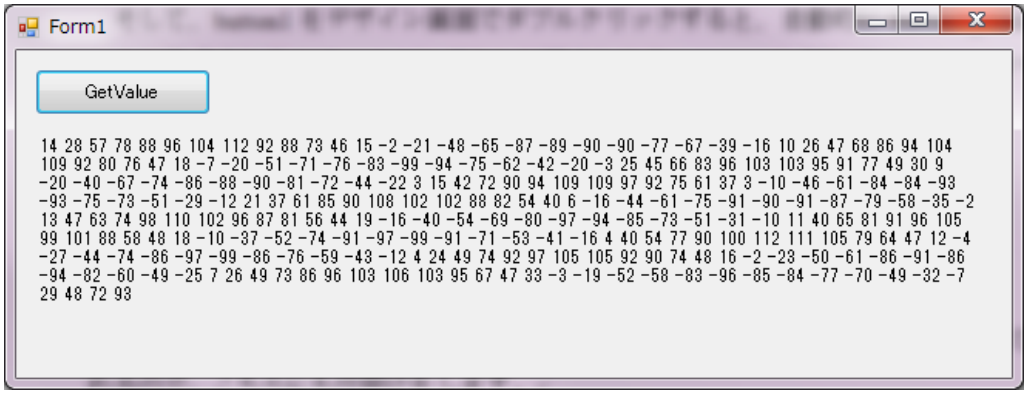
```
        }

        private void button1_Click(object sender, EventArgs e)
        {
            stars.Send("testdev", "GetValue");
        }
    }
```

つぎに、このコマンドに対するリプライが返されてくると handler が自動的によびだされるので、こちらにも仕掛けをします。

```
        private void handler(object sender, STARS.StarsCbArgs ev)
        {
            // Receiving Stars message by callback mode. Show reply at messagebox.
            //MessageBox.Show("Reply via callback: " + ev.allMessage);
            label1.Text = ev.parameters;
            return;
        }
```

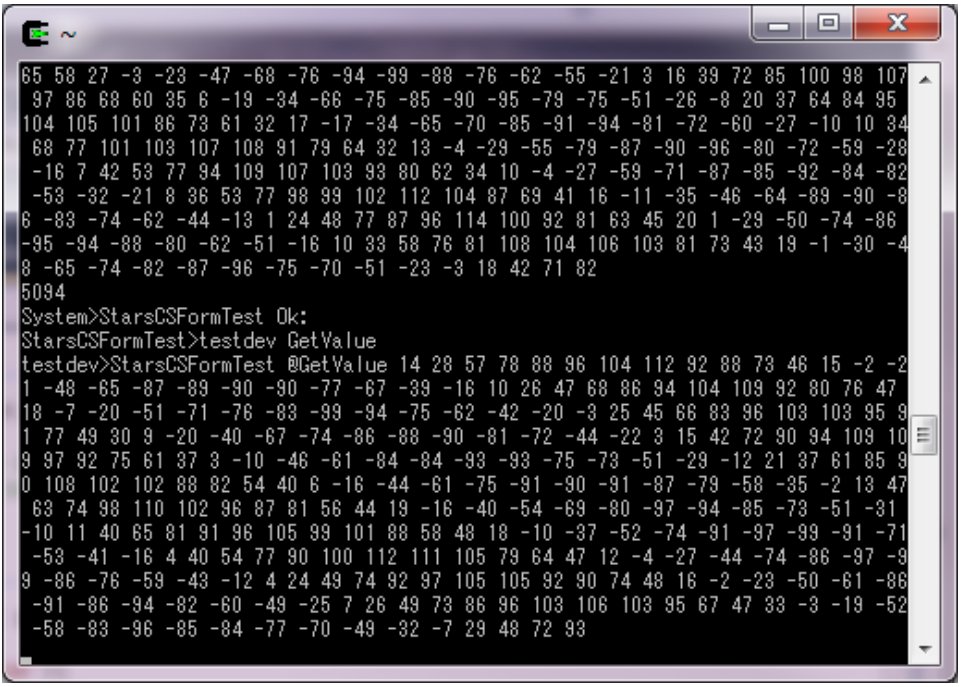
ここではリプライメッセージのなかから、値の部分だけを取り出して、label1 に表示するようにしています。

以上の作業だけで、改造は終了です。デバッグモードで実行してみましょう。




GetValue ボタンをクリックするたびに label1 の内容が変化するのが観測できるはずです。

この様子を Debugger 機能でモニタした結果が次のようになります。なお、表示には Cygwin の Telnet クライアントを使用しています。




以上のように C#用のテンプレートを使用すると、簡単に C# STARS GUI Client を作成する事が出来ます。なお、この例ではエラー処理をかなり省いていますので、通常はもう少し複雑な変更が必要になるかもしれません。