Client Bridge について

１．はじめに

Client Bridge は Stars においてブリッジ機能を持つモジュールです。

Tak サーバからみると Stars の１クライアントとして動作します。

そして TAK サーバ同様、Stars クライアントの接続受付が可能なサーバとして動作します。

Client Bridge は、TAK サーバと同等な接続クライアントに対するセキュリティーチェック機構ならびにターミナル名のエイリアシング機能を持っています。

Client Bridge は TAK サーバとで異なり、接続するノード名が重複していても接続が可能です。これは、Client Bridge が内部的にノード名に加えサフィックスを付加してクライアントを管理しているからです。

２．Client Bridge の起動

Client Bridge を起動する前に、Client Bridge のサーバ Port 番号と Stars ノード名をあらかじめ決めておく必要があります。（Client Bridge の起動方法については readme.txt をご確認ください。）

以下、Client Bridge の Server を「localhost」、ServerPort を「7057」、Stars ノード名を「clientBridge」として説明します。

３．Client Bridge サーバへの接続クライアントの Stars ノード名について

Client Bridge サーバへのクライアント接続方法は TAK サーバと同じです。

Client Bridge のサーバ名とポートを指定して接続し、Client Bridge クライアントのノード名とキーワードを入力してください。

```
$ telnet localhost 7057     ← Client Bridge に接続します。
9086                        ← キーワード識別子が返されます。
term1 stars                 ← Stars ノード名とキーワードを入力します。
clientBridge>term1 Ok:      ← Client Bridge への接続が完了しました。
clientBridge whoami         ← Client Bridge に接続した Stars クライアントのノード名を確認します。
clientBridge>term1 @whoami clientBridge.term1.564311283
                            ← 接続クライアントの Stars ノード名が確認返されます。
```

Client Bridge への接続が成功すると、Client Bridge が管理するクライアントの Stars

ノード名は「clientBridge.＜接続ノード名＞.＜サフィックス＞」となります。
＜サフィックス＞は Client Bridge を起動してからの経過時間（マイクロ秒）から作成されるユニークな数字です。

Client Bridge クライアントが他のクライアント宛にメッセージを送信する場合、Client Bridge は送信元となる Client Bridge クライアントのノード名を clientBridge.＜接続ノード名＞.＜サフィックス＞に変換してから他のクライアント宛てにメッセージを送信します。
また他のクライアントが Client Bridge クライアントを宛先とするメッセージを送信する場合、Client Bridge は、宛先を＜接続ノード名＞に変換して Client Bridge クライアント宛てにメッセージを送信します。
ですので、Client Bridge クライアントはサフィックスを含む正式な Stars ノード名を全く意識せずに、他クライアントとメッセージを交換することができます。

４．Client Bridge クライアントからの「System flgon」要求について
Client Bridge クライアントが System flgon コマンドを発行して特定ノードからのイベントメッセージの受信を要求すると、Client Bridge は内部的に「System flgon」を要求した Client Bridge クライアントの情報を保存し、 TAK サーバに対して Client Bridge「clientBridge」を受信先とした「System flgon」要求を実行します。
Client Bridge は TAK サーバからイベントメッセージを受信すると、『System flgon」を要求した Client Bridge クライアントに対してイベントメッセージを送信します。
この仕組みによって、複数のClient Bridge クライアントが同一ノードに対して『System flgon」要求を実行した場合の通信負荷を軽減します。

５．Client Bridge クライアント間の送受信制限
Client Bridge は、TAK サーバでも実現されているコマンドパーミッション機能を利用して、接続ノード名「term1」を除く Client Bridge 接続クライアント間のメッセージのやりとりを禁止しています。
（初期設定では、clientBridge.term1 を状態確認のため例外的に使用できるようになっています。）

６．Client Bridge における接続ノード名「Debugger」について
Client Bridge への接続ノード名『Debugger」については１クライアントのみに接続が制限されます。
Stars のクライアント名は clientBridge.Debugger.1 で常に固定です。

７．Client Bridge の TAK サーバへの自動接続

TAK サーバに接続できない場合、約１５秒のインターバルで再接続を試みます。

再接続が成功すると、それまでにClient Bridge クライアントが実行した『System flgon」要求に関して正しく機能するよう自動処理されますので、あらためてClient Bridge クライアントから「System flgon」要求を実行する必要はありません。

８．Client Bridge のクライアントの接続するノード名の重複がない場合、Client Bridge のクライアントに対してメッセージを送信するとき宛先ノード名のサフィックスを省略することができます。またClient Bridge のクライアントからのリプライメッセージおよびイベントメッセージの送信元ノード名のサフィックスは省略されて返されます。

（ClientBridge に対して接続ノード名 pm16ccntlpnl が１ノードのみ接続している場合）

term1>clientBridge.pm16ccntlpnl hello

clientBridge.pm16ccntlpnl>term1 @hello nice to meet you.

サフィックスを省略しても正しく配信されます。

term1>clientBridge.pm16ccntlpnl.45085546 hello

clientBridge.pm16ccntlpnl>term1 @hello nice to meet you.

サフィックスを付けた場合も正しく配信されます。リプライメッセージの配信元ノード名のサフィックスは省略されます。

（ClientBridge に対して同一接続ノード名 pm16ccntlpnl が複数接続している場合）

term1>clientBridge.pm16ccntlpnl hello

clientBridge>term1 @hello Er: clientBridge.pm16ccntlpnl is down.

サフィックスを省略すると配信先エラーとなります。

term1>clientBridge.pm16ccntlpnl.45085546 hello

clientBridge.pm16ccntlpnl.45085546>term1 @hello nice to meet you.

サフィックスを付けた場合は正しく配信されます。リプライメッセージの配信元ノード名のサフィックスは省略されません。

９．Client Bridge のコマンド

・hello

動作をチェックするための hello コマンドが用意されています。単純に以下のような答えを返してくるだけです。

clientBridge hello

clientBridge>term1 @hello Nice to meet you.

・help

Client Bridge コマンドの一覧を返します。

・whoami（Client Bridge 独自コマンド）

Stars ノード名を返します。エイリアシング機能によりエイリアス名がある場合はエイリアス名が返されます。

・loadaliases

Client Bridge は TAK サーバ同様ターミナル名のエイリアシング機能を備えていて、システム構成が変わった場合にも対応できるよう考慮されています。また、エイ リアスの変更時にシステムを停止する必要はありません。loadaliases コマンドは変更されたエイリアスファイルをシステムに再び読み込みます。

・listaliases

エイリアシングの状況を返します。

・loadpermission

Client Bridge は TAK サーバ同様コマンドパーミッション機能を備えていて、システム構成が変わった場合にも対応できるよう考慮されています。また、エイリアスの変更時にシステムを停止する必要はありません。loadpermission コマンドは変更されたコマンド許可不許可ファイルをシステムに再び読み込みます。

・listnodes

Client Bridge に接続されている全てのノード名を返します。

・gettime

現在の時刻を返します。

・getversion

Client Bridge のバージョンを返します。

・quit もしくは exit

Client Bridge との接続を解除します。

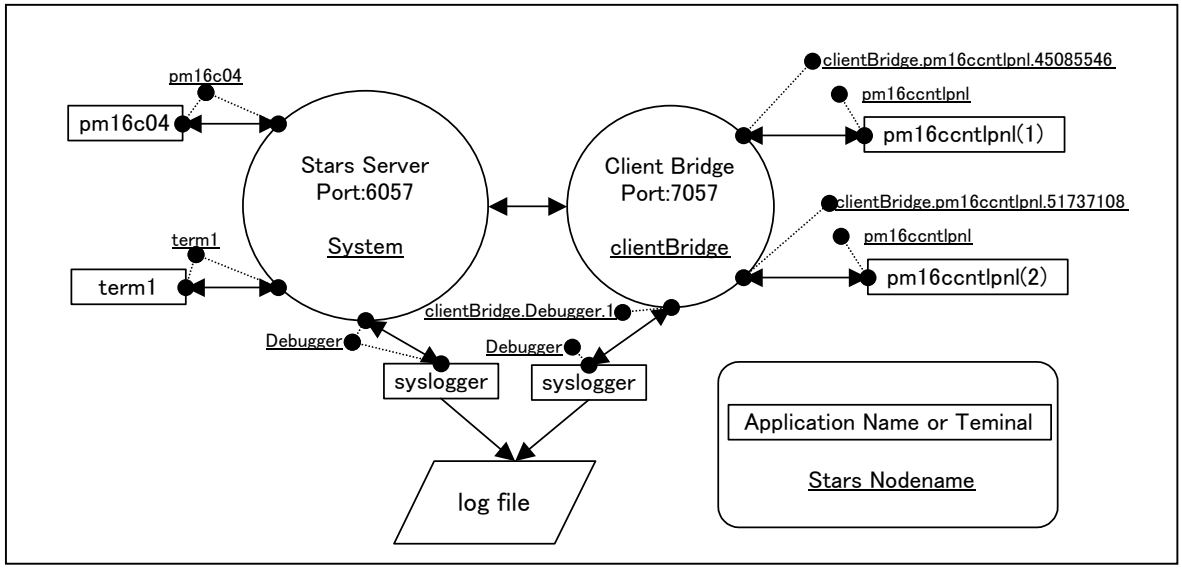
・connect2stars（Client Bridge 独自コマンド）

TAK サーバと接続していなければ接続を試みます。
もし TAK サーバとの接続に失敗するとリプライメッセージとして
@connectstars Er: System is down.を返します。

１０．Client Bridge の実行例

以下、Client Bridge を介在して pm16c04(I/O Client)と pm16ccntlpnl（Application Client）２クライアントの通信例を紹介します。

＜構成＞



モジュール名 ／ターミナル名	Stars ノード名	役割・機能
Stars Server	System	TAK サーバ
Client Bridge	ClientBridge	Client Bridge
Pm16c04	pm16c04	Tsuji Con pm16c04 の I/O Client
Pm16ccntlpnl(1),(2)	clientBridge.pm16ccntlpnl.45085546 clientBridge.pm16ccntlpnl.517371085	pm16c04 を操作する Application Client 今回は２クライアントを接続
syslogger(1)	Debugger	TAK サーバ側のログを出力する
syslogger(2)	ClientBridge.Debugger.1	Client Bridge 側のログを出力する
term1	term1	ターミナル（TAK サーバ接続）

今回は TAK サーバと Client Bridge それぞれに syslogger モジュールを接続させ、同一ファイルに対してログを出力するよう設定しています。そのためログファイルでは同一メッセージが複数出力される場合があります。

＜シナリオ＞

1. Stars Server を起動します。
2. Stars Server に対して syslogger を立ち上げます。Nodename: Debugger
3. Stars Server に対して pm16c04 を立ち上げます。Nodename: pm16c04
4. Stars Server に対して term1 を立ち上げます。Nodename: term1
5. Client Bridge を起動します。Client Bridge Server Port:7057　Nodename: clientBridge
6. Client Bridge に対して syslogger を立ち上げます。Nodename: clientBridge.Debugger.1
7. term1 から「System listnodes」を実行します。
   pm16c04、Debugger、term1、clientBridge の接続が確認できます。

   2007-07-24 11:25:00.080 System>term1 @listnodes clientBridge Debugger term1 pm16c04

8. term1 から「clientBridge listnodes」を実行します。
   clientBridge.Debugger.1 の接続が確認できます。

   2007-07-24 11:25:04.235 clientBridge>term1 @listnodes clientBridge.Debugger.1

9. Client Bridge に対して１つ目の pm16ccntlpnl を立ち上げます。
   pm16ccntlpnl は起動時 System flgon を実行して pm16c04 のイベントメッセージの受信を要求します。Client Bridge は、pm16ccntlpnl でなく clientBridge がイベントメッセージを受信するよう System flgon を実行します。

   2007-07-24 11:27:06.084 clientBridge.pm16ccntlpnl.45085546>System flgon pm16c04

   2007-07-24 11:27:06.085 clientBridge>System flgon pm16c04

   2007-07-24 11:27:06.080 System>clientBridge @flgon Node pm16c04 has been registered.

   2007-07-24 11:27:06.086 System>pm16ccntlpnl @flgon Node pm16c04 has been registered.

10. Client Bridge に対して２つ目の pm16ccntlpnl を立ち上げます。
    pm16ccntlpnl は起動時 System flgon を実行して pm16c04 のイベントメッセージの受信を要求します。Client Bridge は内部的に２つ目の pm16ccntlpnl をメッセージ配信対象に加えます。

11. どちらか片方の pm16ccntlpnl から pm16c04 のリモートローカルの切り替えを実行します。
    pm16c04 からリモートローカル状態の変化がイベントメッセージとして返されます。

    2007-07-24 11:32:42.703 clientBridge.pm16ccntlpnl.517371085>pm16c04 SetFunction 0

    2007-07-24 11:32:42.703 clientBridge.pm16ccntlpnl.517371085>pm16c04 SetFunction 0

    2007-07-24 11:32:42.750 pm16c04>clientBridge _ChangedFunction 0

    2007-07-24 11:32:42.752 pm16c04>clientBridge.pm16ccntlpnl.517371085 @SetFunction 0 Ok:

2007-07-24 11:32:42.756 pm16c04>pm16ccntlpnl @SetFunction 0 Ok:

2007-07-24 11:32:42.758 pm16c04>clientBridge _ChangedFunction 0

2007-07-24 11:32:42.758 pm16c04> pm16ccntlpnl _ChangedFunction 0

2007-07-24 11:32:42.759 pm16c04> pm16ccntlpnl _ChangedFunction 0

（以上です）