**TCP Application Protocol – TCP/IP based Inter-process Communication**

TCP Application Protocol is an IPC (Inter-process Communication) created to provide common remote interface for between applications. TCP Application Protocol is server/client text message protocol built on top of TCP (Transmission Control Protocol)

**Why TCP?**

Communication between two applications is generally called as IPC (Inter-Process Communication). There are many ways of IPC. TCP/IP is one of them. The advantage of TCP/IP is that the communicating applications can be running on different computers and different locations. Since Internet also works on TCP/IP, remote applications can communicate over the internet. TCP/IP is completely platform independent, standard, and implemented by all operationg systems (and many other devices). *(Reference: Code Project - A Complete TCP Server/Client Communication and RMI Framework in C# .NET – Implementation)*

**Why TCP Application Protocol?**

TCP Application Protocol is created as high level communication protocol to provide a common remote interface between applications which can be easily integrated to any application with minimum coding effort from developer on both server and client application.

**Advantage of TCP Application Protocol**

TCP Application Protocol is server/client implementation where server application will response to request initiate from client application. With TCP Application Protocol, application specific commands can be easily defined by server application. Each registered command keyword can include with one or more optional or mandatory parameter as needed. Incoming message from client will be verify against registered command set and have defined parameters parse and assigned. Developer can focus on implementation of each of the command.

As human readable text message based protocol, no translation and interpretation of machine code required on message passed between server and client. Furthermore, registered commands on server application can be easily retrive by client application without need to refer to user manual or code.

With TcpAppServerPlugin class, command set for plugin components can be developed andadded to host application easily at later stage without required to rebuild the host application. Moreover, client application may instantiate objects on server application remotely. Number of clients allowed to connect to server at a time can also be configured.

**About TCP Application Protocol**

Any TCP client including realterm can be use to interact with application which implement TCP Application protocol since the protocol is implement in human readable text message.

In general, the command between client and server is defined as follow:

* Command sent from client start with command keyword follow by parameters of the defined command if any and terminated with carriage return (\r, 0x0D). Extra parameters will be ignored.  
  <Command Keyword> [Parameter0] … [ParameterN]
* Response from server to client is also begin with command keyword follow by status and response message. Status flag is either OK or ERR.  
  <Command Keyword> <Status> <Response Message / Return Value>

TCP Application Server have predefined command set which created to retrieve information about server application, registered commands and plugins which intended to ease the integration process between client and server application. Simply send Help command to server will return list of registered commands. Other commands such as FunctionList?, ApplicationName?, ApplicationVersion? are created to return information from server application to connected client.

**Architecture of TCP Application**

TcpAppServer (TCP Application Server) and TcpAppClient (TCP Application Client) are derived from TcpServer and TcpClient respectively which is general TCP communication object.

**Getting Started with TCP Application Server**

* Install NuGet Package CodeArtEng.Tcp to project.
* Create an object of TcpAppServer.
* (Optional) Define welcome message which return to client on connect.
* (Optiona) Subscribe to ClientConnected and ClientDisconnected event.
* Start server.
* Server shall be stopped and dispose on application exit to release executing thread.

<Code Example: TcpAppServer>

**Register Application Specific Command**

* Command can be register to server application with RegisterCommand method.
* Optional and Mandatory argument can be added to command as TcpAppParameter object.
* Each command shall associate with a callback function which defined it’s implementation.

<Code Example >

**Working with Plugin**

Plugin provide great extenability to application where new feature and components can be added at later stage. TCP Application Protocol equipped with capability to handle and extend command set in plugin components as well as instantiate object in server application, let’s see how.

**Preparation on Plugin Components**

* Install NuGet Package CodeArtEng.Tcp to plugin project.
* Implement ITcpAppServerPlugin interface on plugin class.
* Create plugin implementation class with TcpAppServerPlugin as base class or private object in plugin class. The TcpAppServerPlugin class will interact with TcpAppServer for command registration and execution.
* Register commands from plugin components to host application using TcpAppServerPlugin object.
* Make sure ExecutePluginCommand and ShowHelp methods are call the corresponding method in TcpAppServerPlugin object.

**Add Plugin Component to Server Application**

On server application, include plugin component object to TcpAppServer upon created with AddPlugin method. It’s important to ensure the Alias name for each plugin component is unique. This method is designed for object which instantiate by server application.

Alternatively, plugin object can be instantiate by client application. To do so, the allowed plugin type must be first registered with TcpAppServer using RegisterPluginType method. Only ITcpAppServerPlugin object type is allowed to register.

Commands registered with plugin object can be retrieve by sending Help <Alias> command from client to server. To execute command in plugin object, just send Execute <Alias> <Command Keyword> [Parameters].

**Managing Connected Clients**

If your server application have no restriction on numbers of client connected at a time, nothing need to be done. Otherwise, you may use the MaxClients property to limit number of connected clients. Alternatively, we can also subscribe to ClientConnected event to accept or reject an incoming connection.

**Future Plan**