Service Packet Creation Walkthrough

Quick overview

This walkthrough is for setting up a new packet or new application on the service. This requires changing the source code of the service. If this document confuses you there is a SNPServiceDocumentation doc that explains JSon Packet structure and the architecture in detail.

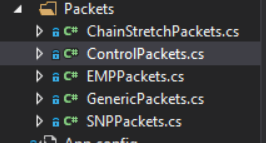
One more thing there is several pieces of documentation to update as you create a new packet. Without these this system can get very disorganized and I am trying to avoid that. Please create any documentation for packets you create as discussed in this documentation.

Finally context as to how the program is structured. All message incoming code and controller code is in SNPService.cs From there it will branch out to the different applications.

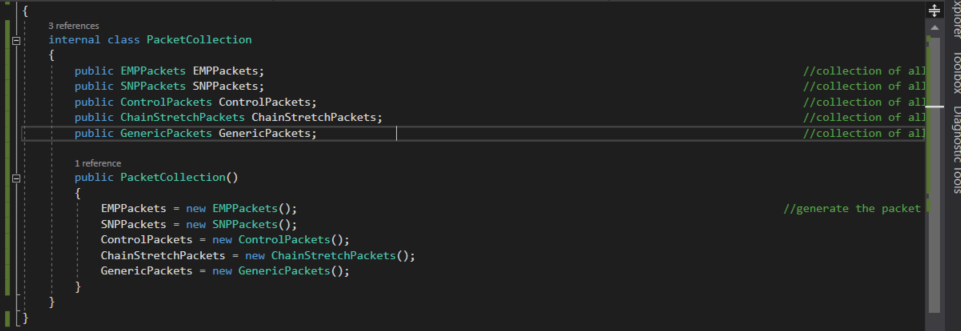
Creation Of New Packets to a new Application

Creating the Packet Class.

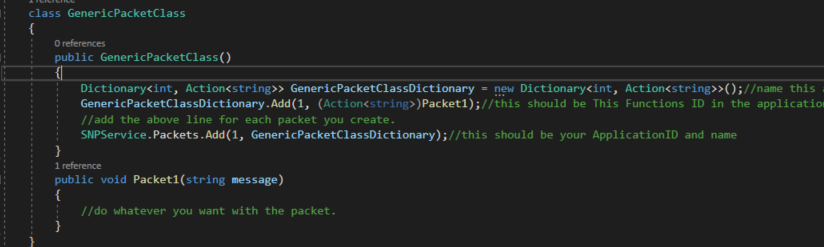
Inside of the Packet Folder there is a class for each applications packets. Create a new class for your packet named for the application followed by Packets. If there is any confusion on how to do this a very simple version is available in GenericPacketClass.cs or a more flushed out one in ControlPackets.cs



Next inside of PacketCollection.cs go to the Variable Section at the top and add a version of the class like the others have done. Then add your constructor in the Packet Collection Constructor.



Next we need to setup Routing for your packet. This is done in a 2 layer dictionary. Inside of your constructor you should see the following code by default



Foreach packet you setup you must add to your classes dictionary the functions ID and the function you want called. The above code sets the Function Packet1 to be called with the contents of the message if the Function ID is equal to 1 and the application ID is equal to one. The application ID is set in the following line where you add the Applications Dictionary to the Overall Dictionary with the id of 1

Next back in your packet class design the functions to do whatever you would like to do on the packet. For SQL Packets a good sample Packet is RunSQLCommand in GenericPackets.cs where receivedPacket["Command"].ToString() is replaced with the Command you are running to ENGDB.

Next Document everything you have added by Adding onto Packet Samples inside of TXT Scraps with the application and packets added as well as a sample for the Packets ( do this in notepad++ so that you can add the entire packet).

Next Document everything in the SNP Word document. To do this you add the application and packets like the others there as well as a description for any new fields you are adding.

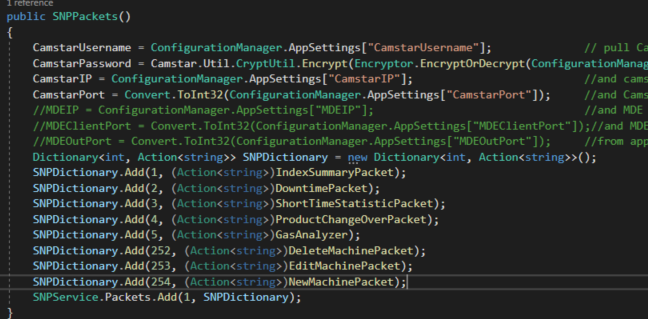
If you are creating Tables in SQL it is nice to have a description table with descriptions of the columns.

Creation Of New Packets to an Existing Application

Find the Packet class. It should be labeled as ApplicationPackets.

Create whatever function you are going to be using in that class as a public void returning function similar to those there. ( just copy past some code and change it to do what you want don’t reinvent the wheel).

Next at the top of that application there should be a Dictionary labled for the application.(SNP Shown as its most likely what you are looking for)



To add your packet to the routing dictionary you have to copy one of the other SNPDictionary.Adds and replace the Int on the left with your packets ID and the function on the right with your packets function. Once that’s done everything will be called like magic!

Next Document everything you have added by Adding onto Packet Samples inside of TXT Scraps with the application and packets added as well as a sample for the Packets ( do this in notepad++ so that you can add the entire packet).

Next Document everything in the SNP Word document. To do this you add the application and packets like the others there as well as a description for any new fields you are adding.

If you are creating Tables in SQL it is nice to have a description table with descriptions of the columns. Or add to the existing description table.