```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using FileHelpers;
using System.Data.Odbc;
using System.Xml;
using System.Xml.Linq;


namespace Avatar_Solutions
{
    public class AvatarConnection
    {
        public IEnumerable<XElement> Rules;

        public IEnumerable<XElement> currentRule;
        public string CurrentRuleTitle;
        public string BillableCode;
        public List<string> Componets;
        public List<string> ServiceExclude;
        public List<string> FSExclude;
        public List<string> FSInclude;
        public List<string> ProgramExclude;
        public List<string> ProgramInclude;
        public int ReqFTF;
        public int ReqTot;
        public List<ServiceRecord> ServList { get; set; }
        public bool OverrideStaff {get; set; }
        public string OverrideStaffID { get; set; }
        public ServiceRecord.ServiceType currentServiceType { get; set; }

        OdbcConnectionStringBuilder connBuilder;
       public OdbcConnection conn { get; set; }
       public bool opened { get; private set; }
        public string uname;

        public List<BillableProvider> staffList;
        public List<PractitionerRecord> staffPracGuarNumb;

        [DelimitedRecord("\t")]
        public class PractitionerRecord
        {
            public string Practitioner { get; set; }
            public string GuarantorID { get; set; }
            public string Program { get; set; }
            public string EffectiveDate { get; set; }
            public string ExpirationDate { get; set; }
            public string NPINumber { get; set; }
            public string TaxonomyCode { get; set; }
            public string SecondaryIdentificationQualifier { get; set; }
            public string SecondaryIdentificationNumber { get; set; }
            public string SupervisingPractitioner { get; set; }
            public string ReferenceIdentificationQualifier_837P2010ABREF01 { get; set; }
            public string PaytoProviderAdditionalIdentifier_837P2010ABREF02 { get; set; }
            public string ClaimNoteReferenceCode_837P2300NTE01 { get; set; }
            public string ClaimNoteText_837P2300NTE02 { get; set; }
            public string ServiceFacilityNameLastorOrganizationName_837P2310D2420CNM103 { get; set; }
            public string ServiceFacilityIdentificationCodeQualifier_837P2310D2420CNM108 { get; set; }
            public string ServiceFacilityIdentificationCode_837P2310D2420CNM109 { get; set; }
            public string ServiceFacilityAddressInformation_837P2310D2420CN301 { get; set; }
            public string ServiceFacilityZipCode_837P2310D2420CN403 { get; set; }
            public string ServiceFacilityCity_837P2310D2420CN401 { get; set; }
            public string ServiceFacilityState_837P2310D2420CN402 { get; set; }
            public string ServiceFacilityReferenceIdentificationQualifier_837P2310D2420CREF01 { get; set; }
            public string ServiceFacilityReferenceIdentification_837P2310D2420CREF02 { get; set; }
            public string ServiceNoteReferenceCode_837P2400NTE01 { get; set; }
            public string ServiceNoteText_837P2400NTE02 { get; set; }
            public string ClaimNoteReferenceCode_837I2300NTE01 { get; set; }
            public string ClaimNoteText_837I2300NTE02 { get; set; }
            public string BillingNoteReferenceCode_837I2300NTE01 { get; set; }
            public string BillingNoteText_837I2300NTE02 { get; set; }
            public string ClaimNoteReferenceCode_837D2300NTE01 { get; set; }
            public string ClaimNoteText_837D2300NTE02 { get; set; }
            public string ServiceNoteReferenceCode_837D2400NTE01 { get; set; }
            public string ServiceNoteText_837D2400NTE02 { get; set; }
            public string PractitionerSecondaryIdentificationQualifier2_837ElectronicBilling { get; set; }
            public string PractitionerSecondaryIdentifier2_837ElectronicBilling { get; set; }
            public string ReferenceIdentificationQualifier_837I2330DREF01 { get; set; }
            public string ReferenceIdentification_837I2330DREF02 { get; set; }
            public string PractitionerBillingQualifierOther_FormLocator17a_HCFA1500NPIVersionOnly { get; set; }
            public string PractitionerBillingOther_FormLocator17a_HCFA1500NPIVersionOnly { get; set; }
            public string PractitionerBillingNPI_FormLocator17b_HCFA1500NPIVersionOnly { get; set; }
            public string SupplementalInformation_FormLocator24D_HCFA1500NPIVersionOnly { get; set; }
            public string PractitionerBillingNPI_FormLocator24J_HCFA1500NPIVersionOnly { get; set; }
            public string AdditionalProviderID_ADAFormLocator58 { get; set; }
            public string PractitionerBillingQualifierOther_FormLocator33B_HCFA1500NPIVersionOnly { get; set; }
            public string PractitionerBillingOther_FormLocator33B_HCFA1500NPIVersionOnly { get; set; }
            public string Comments { get; set; }

            public PractitionerRecord()
            {

            }
        }




        public class BillableProvider
        {
            public string name { get; set;}
            public string ID { get; set; }
            public string Taxonomy { get; set; }
            public string npi { get; set; }

            public BillableProvider(string n, string id, string tax, string np)
            {
                name = n;
                ID = id;
                Taxonomy = tax;
                npi = np;
            }

            public BillableProvider()
            {

            }
```

```csharp


        }




        internal AvatarConnection(string SYSTEMCODE, string Username, string Pasword)
        {
            uname = Username;
            connBuilder = new OdbcConnectionStringBuilder();
            connBuilder.Driver = "InterSystems ODBC";
            connBuilder.Add("SERVER", "127.0.0.1");
            connBuilder.Add("PORT", "4972");
            connBuilder.Add("DATABASE", "AVPM");
            connBuilder.Add("UID", SYSTEMCODE + ":" + Username);
            connBuilder.Add("PWD", Pasword);

          //  System.Windows.Forms.MessageBox.Show(connBuilder.ConnectionString);

            try
            {
                conn = new OdbcConnection(connBuilder.ConnectionString);
                conn.Open();
               // System.Windows.Forms.MessageBox.Show("Connection Opened!");
                conn.Close();
                opened = true;

                Componets = new List<string>();
                ServiceExclude = new List<string>();
                FSExclude = new List<string>();
                FSInclude = new List<string>();
                ProgramExclude = new List<string>();
                ProgramInclude = new List<string>();
                //StaffPrac
                ServList = new List<ServiceRecord>();
                staffList = new List<BillableProvider>();
                staffPracGuarNumb = new List<PractitionerRecord>();


            } catch(Exception e)
            {
                System.Windows.Forms.MessageBox.Show("Connection Failed, " + Environment.NewLine + "if Avatar is opened and System Code is correct," + Environment.NewLine + "please correct login information.");
                //throw e;
            }


            //OdbcCommand cmd = con.CreateCommand();

            //cmd.CommandText = "SELECT Facility from table_facility_defaults";

        }

        public void GetStaff()
        {
            string stm = "select staffid, name, npi_number, taxonomy_code FROM SYSTEM.staff_current_demographics";

            using (OdbcCommand cmd = new OdbcCommand(stm, conn))
            {
                using (OdbcDataReader rdr = cmd.ExecuteReader())
                {
                    while (rdr.Read())
                    {
                        this.staffList.Add(new BillableProvider
                        {
                            ID = rdr["staffid"].ToString(),
                            name = rdr["name"].ToString(),
                            npi = rdr["npi_number"].ToString(),
                            Taxonomy = rdr["taxonomy_code"].ToString()
                        });
                    }





                }
            }
        }

        public decimal GetTotalPracNumber()
        {
            string stm = "select count(*) as total from SYSTEM.table_prac_num_guar_prog";
            decimal total;
            using (OdbcCommand cmd = new OdbcCommand(stm, conn))
            {

                using (OdbcDataReader rdr = cmd.ExecuteReader())
                {

                    rdr.Read();
                    total = decimal.Parse(rdr["total"].ToString());
                }
            }

            return total;
        }

        public void GetPracNumbersRecords()
        {
            string stm = "SELECT claim_note_text_837d, claim_note_ref_837d_code,
            bill_note_text_837i, bill_note_ref_837i_code, svc_fac_name_837p, STAFF_ID,
            GUARANTOR_ID, program_code, effective_date, expiration_date, npi_number,
            taxonomy_code, second_ident2_qual_code, second_ident2_number,
            supervising_pract_id, ref_id_qua_837p_code, ref_id_837p,
            claim_note_ref_837p_code, claim_note_text_837p, svc_fac_id_qua_code_837p,
            svc_fac_address_837p, svc_fac_zip_837p, svc_fac_city_837p,
            svc_fac_state_837p, svc_fac_ref_id_code_837p, svc_fac_ref_id_837p,
            svc_note_ref_837p_code, svc_note_text_837p, claim_note_ref_837i_code,
            claim_note_text_837i, svc_note_ref_837d_code, svc_note_text_837d,
            second_ident2_qual_code, second_ident2_number, ref_id_qua_837i_code,
            ref_id_837i, hcfa_npi_17a_qua_code, hcfa_npi_17a, hcfa_npi_17b,
            hcfa_npi_loc_24d_supp, hcfa_npi_loc_33a_npi, add_prov_id_dental_f158,
            hcfa_npi_loc_33b_code, hcfa_npi_loc_33b_other, comments  from
            SYSTEM.table_prac_num_guar_prog";

            using (OdbcCommand cmd = new OdbcCommand(stm, conn))
            {
                using (OdbcDataReader rdr = cmd.ExecuteReader())
                {
                    while (rdr.Read())
                    {
                        this.staffPracGuarNumb.Add(new PractitionerRecord
                        {
                            Practitioner = rdr["STAFF_ID"].ToString(),
                            GuarantorID = rdr["GUARANTOR_ID"].ToString(),
                            Program = rdr["program_code"].ToString(),
                            EffectiveDate = rdr["effective_date"].ToString(),
                            ExpirationDate = rdr["expiration_date"].ToString(),
                            NPINumber = rdr["npi_number"].ToString(),
                            TaxonomyCode = rdr["taxonomy_code"].ToString(),
                            SecondaryIdentificationQualifier =
                            rdr["second_ident2_qual_code"].ToString(),
                            SecondaryIdentificationNumber =
                            rdr["second_ident2_number"].ToString(),
                            SupervisingPractitioner =
                            rdr["supervising_pract_id"].ToString(),

                            ReferenceIdentificationQualifier_837P2010ABREF01 =
                            rdr["ref_id_qua_837p_code"].ToString(),

                            PaytoProviderAdditionalIdentifier_837P2010ABREF02 =
                            rdr["ref_id_837p"].ToString(),

                            ClaimNoteReferenceCode_837P2300NTE01 =
                            rdr["claim_note_ref_837p_code"].ToString(),

                            ClaimNoteText_837P2300NTE02 =
                            rdr["claim_note_text_837p"].ToString(),


                            ServiceFacilityNameLastorOrganizationName_837P2310D2420CNM103
                             = rdr["svc_fac_name_837p"].ToString(),

                            ServiceFacilityIdentificationCodeQualifier_837P2310D2420CNM10
                            8 = rdr["svc_fac_ref_id_code_837p"].ToString(),

                            ServiceFacilityIdentificationCode_837P2310D2420CNM109 =
                              rdr["svc_fac_ref_id_837p"].ToString(),

                            ServiceFacilityAddressInformation_837P2310D2420CN301 =
                            rdr["svc_fac_address_837p"].ToString(),

                            ServiceFacilityZipCode_837P2310D2420CN403 =
                            rdr["svc_fac_zip_837p"].ToString(),

                            ServiceFacilityCity_837P2310D2420CN401 =
                            rdr["svc_fac_city_837p"].ToString(),

                            ServiceFacilityState_837P2310D2420CN402 =
                            rdr["svc_fac_state_837p"].ToString(),


                            ServiceFacilityReferenceIdentificationQualifier_837P2310D2420
                            CREF01 = rdr["svc_fac_ref_id_code_837p"].ToString(),
                            ServiceFacilityReferenceIdentification_837P2310D2420CREF02
                            = rdr["svc_fac_ref_id_code_837p"].ToString(),
                            ServiceNoteReferenceCode_837P2400NTE01 =
                            rdr["svc_note_ref_837p_code"].ToString(),

                            ServiceNoteText_837P2400NTE02 =
                            rdr["svc_note_text_837p"].ToString(),

                            ClaimNoteReferenceCode_837I2300NTE01 =
                            rdr["claim_note_ref_837i_code"].ToString(),

                            ClaimNoteText_837I2300NTE02 =
                            rdr["claim_note_text_837i"].ToString(),

                            BillingNoteReferenceCode_837I2300NTE01 =
                            rdr["bill_note_ref_837i_code"].ToString(),

                            BillingNoteText_837I2300NTE02 =
                            rdr["bill_note_text_837i"].ToString(),
ClaimNoteReferenceCode_837D2300NTE01 = rdr["claim_note_ref_837d_code"].ToString(),

                            ClaimNoteText_837D2300NTE02 =
                            rdr["claim_note_text_837d"].ToString(),

                            ServiceNoteReferenceCode_837D2400NTE01 =
                            rdr["svc_note_ref_837d_code"].ToString(),

                            ServiceNoteText_837D2400NTE02 =
                            rdr["svc_note_text_837d"].ToString(),

                            PractitionerSecondaryIdentificationQualifier2_837ElectronicBi
                            lling = rdr["second_ident2_qual_code"].ToString(),

                            PractitionerSecondaryIdentifier2_837ElectronicBilling =
                            rdr["second_ident2_number"].ToString(),

                            ReferenceIdentificationQualifier_837I2330DREF01 =
                            rdr["ref_id_qua_837i_code"].ToString(),

                            ReferenceIdentification_837I2330DREF02 =
                            rdr["ref_id_837i"].ToString(),

                            PractitionerBillingQualifierOther_FormLocator17a_HCFA1500NPIV
                            ersionOnly = rdr["hcfa_npi_17a_qua_code"].ToString(),

                            PractitionerBillingOther_FormLocator17a_HCFA1500NPIVersionOnl
                            y = rdr["hcfa_npi_17a"].ToString(),

                            PractitionerBillingNPI_FormLocator17b_HCFA1500NPIVersionOnly
                            = rdr["hcfa_npi_17b"].ToString(),

                            SupplementalInformation_FormLocator24D_HCFA1500NPIVersionOnly
                             = rdr["hcfa_npi_loc_24d_supp"].ToString(),

                            PractitionerBillingNPI_FormLocator24J_HCFA1500NPIVersionOnly
                            = rdr["hcfa_npi_loc_33a_npi"].ToString(),

                            AdditionalProviderID_ADAFormLocator58 =
                            rdr["add_prov_id_dental_fl58"].ToString(),

                            PractitionerBillingQualifierOther_FormLocator33B_HCFA1500NPIV
                            ersionOnly = rdr["hcfa_npi_loc_33b_code"].ToString(),

                            PractitionerBillingOther_FormLocator33B_HCFA1500NPIVersionOnl
                            y = rdr["hcfa_npi_loc_33b_other"].ToString(),

                            Comments = rdr["comments"].ToString()
                        });
                    }





                }
            }

            //return true;
        }

        public OdbcCommand SQLCaseMgmt(string FROM, string Thru, List<string>
        Componets, List<string> ExcludePrograms = null, List<string> ExcludeFS = null)
        {
            OdbcCommand cmd = new OdbcCommand();
           // cmd.CommandText = "SELECT PATID, Service, COUNT(*) as Tot, Program,
           Staffid, DOS" +
           //                  " FROM pmhc_CaseMgmt " +
           //                 "WHERE ((DOS >= '" + FROM + "' and DOS <= '" + Thru + "')" +
           //                  "AND Service IN
            ('770','770F','780','780F','790','790F','771','771F','T2023'))" +
            //                  "GROUP BY PATID, Service " +
            //                  "ORDER BY PATID";

            StringBuilder bldr = new StringBuilder();
            bldr.Append("SELECT PATID, Service, COUNT(*) as Tot, Program, Staffid, DOS,
            Episode ");
            bldr.Append(" FROM pmhc_CaseMgmt ");
            bldr.Append("WHERE (DOS >= '" + FROM + "' and DOS <= '" + Thru + "' ");

            if (Componets != null)
            {
                bldr.Append(IterateExlusion("Service", false, Componets, true));
            }
            if (ServiceExclude != null)
            {
                bldr.Append(IterateExlusion("Service", false, ServiceExclude, true));
            }

            if (ProgramExclude != null)
            {
                bldr.Append(IterateExlusion("Program", true, ProgramExclude));
            }
            if (ProgramInclude != null)
            {
                bldr.Append(IterateExlusion("Program", false, ProgramInclude));
            }

            if (ExcludeFS != null)
            {
                bldr.Append(IterateExlusion("Guar",true, ExcludeFS));
            }

            if (FSInclude != null)
            {
                bldr.Append(IterateExlusion("Guar", false, FSInclude));
            }

            bldr.Append(")");
            bldr.Append(" GROUP BY PATID, Service ");
            bldr.Append(" ORDER BY PATID");

            cmd.CommandText = bldr.ToString();


            return cmd;
        }

        string IterateExlusion(string exlusionField, bool notIn, List<string>
        ExcludePrograms, bool isComponets = false)
        {
            StringBuilder bldr = new StringBuilder();

            if (ExcludePrograms != null)
            {
                for (int i = 0; i < ExcludePrograms.Count; i++)
                {
                    if (ExcludePrograms.Count != 1 && i == ExcludePrograms.Count - 1)
                    {
                        if (isComponets)
                        {
                            bldr.Append(",'" + ExcludePrograms[i] + "', '" +
                            BillableCode + "')");
                        } else
                        {
                            bldr.Append(",'" + ExcludePrograms[i] + "')");
                        }

                    }
                    else if (i != 0)
                    {
                         bldr.Append(",'" + ExcludePrograms[i] + "'");
                    }
                    else
                    {
                        if (notIn)
                        {
                            if (ExcludePrograms.Count == 0)
                            {
                                bldr.Append(" AND " + exlusionField + " NOT IN ('" +
                                ExcludePrograms[i] + "')");

                            } else if (ExcludePrograms.Count == 1)
                            {
                                bldr.Append(" AND " + exlusionField + " NOT IN ('" +
                                ExcludePrograms[i] + "')");

                            }
                            else
                            {
                                bldr.Append(" AND " + exlusionField + " NOT IN ('" +
                                ExcludePrograms[i] + "'");
                            }
                        }
                        else
                        {
                            if (ExcludePrograms.Count == 0)
                            {
                                bldr.Append(" AND " + exlusionField + " IN ('" +
                                ExcludePrograms[i] + "')");

                            }
                            else if (ExcludePrograms.Count == 1)
                            {
                                bldr.Append(" AND " + exlusionField + " IN ('" +
                                ExcludePrograms[i] + "')");

                            }
                            else
                            {
                                bldr.Append(" AND " + exlusionField + " IN ('" +
                                ExcludePrograms[i] + "'");

                            }
                        }
                    }
                }

                return bldr.ToString();
            } else
            {
                throw new Exception("No items in Exlusion Critiera");
            }
        }

        public void LoadFile()
        {
            XElement info = XElement.Load("Requirements.xml");
            Rules = from el in info.Elements()
                    select el;
        }



        public void LoadRule(string RuleName)
        {
            currentRule = from el in Rules.Elements()
                          where el.Name == "Name" && el.Value == RuleName
```

```
                            select el.Parent;

                ProcessRule();

                CurrentRuleTitle = RuleName;


            }

            //Process current rule.
            //This sets all logic for building the SQL Query
            public void ProcessRule()
            {

                foreach (var item in currentRule.Elements("Billable"))
                {
                    BillableCode = item.Value;
                }

                Componets.Clear();
                ProgramExclude.Clear();
                FSExclude.Clear();
                ServiceExclude.Clear();
                ProgramInclude.Clear();
                FSInclude.Clear();
                OverrideStaff = false;
                OverrideStaffID = "";

                foreach (var item in currentRule.Elements("Componets").Elements("Componet"))
                {
                    Componets.Add(item.Value);
                }

                foreach (var item in
                currentRule.Elements("ExcludeServices").Elements("ExcludeService"))
                {
                    ServiceExclude.Add(item.Value);
                }

                foreach (var item in
                currentRule.Elements("ExcludePrograms").Elements("ExcludeProgram"))
                {
                    ProgramExclude.Add(item.Value);
                }

                foreach (var item in
                currentRule.Elements("IncludePrograms").Elements("IncludeProgram"))
                {
                    ProgramInclude.Add(item.Value);
                }

                foreach (var item in
                currentRule.Elements("IncludeFSs").Elements("IncludeFS"))
                {
                    FSInclude.Add(item.Value);
                }

                foreach (var item in currentRule.Elements("Rules"))
                {
                    if (item.Element("FTF").Name == "FTF")
                    {
                        ReqFTF = int.Parse(item.Element("FTF").Value);
                    }

                    if (item.Element("Total").Name == "Total")
                    {
                        ReqTot = int.Parse(item.Element("Total").Value);
                    }

                    if (item.Element("OverrideStaff") != null && item.Element("OverrideStaff").Name == "OverrideStaff")
                    {
                        OverrideStaff = true;
                        OverrideStaffID = item.Element("OverrideStaff").Value;
                    }

                    if (item.Element("ServiceType") != null && item.Element("ServiceType").Name == "ServiceType")
                    {
                        currentServiceType = (ServiceRecord.ServiceType)int.Parse(item.Element("ServiceType").Value);
                    } else
                    {
                        currentServiceType = ServiceRecord.ServiceType.Other;
                    }
                }

               // System.Windows.Forms.MessageBox.Show("BIllable- " + BillableCode + " FTF-" + ReqFTF.ToString() + " TOT-" + ReqTot.ToString());

                //if (((XElement)currentRule.Elements("ExcludePrograms"))))
            }

            public void ProcessServiceExportFile()
            {
                FileHelperEngine<ServiceRecord> engine = new FileHelperEngine<ServiceRecord>();
                var pathWithEnv = @"%USERPROFILE%\Desktop\OutputServiceFile.txt";
                var filePath = Environment.ExpandEnvironmentVariables(pathWithEnv);
                engine.WriteFile(filePath, this.ServList);

                System.Windows.Forms.MessageBox.Show("File Processed and sent to Desktop.");
            }


        }




        [DelimitedRecord("\t")]
        public class ServiceRecord
        {

            private ServiceRecord()
            {
                this.SPI = "100";
            }

            public enum ServiceType { ACT, CaseMgmt, StaffOnly, IDD, Other }

            public ServiceRecord(DateTime SerDate, int ep, string SerCode, string ProgramCode, string staffid, string clid, ServiceType srvType = ServiceType.Other)
            {
                this.DOS = SerDate;
                this.Service = SerCode;
                this.Program = ProgramCode;
                this.Pracitioner = staffid;
                this.ClientID = clid;
                this.Episode = ep;
                if (srvType == ServiceType.ACT || srvType == ServiceType.CaseMgmt)
                {
                    this.SPI = "100";
                    this.Location = "1";
                }

                if (srvType == ServiceType.IDD)
                {
                    this.SPI = "200";
                    this.Location = "10";

                }
            }
            [FieldOrder(1)]
            [FieldConverter(ConverterKind.Date, "MMddyyyy")]
            public DateTime DOS;
            [FieldOrder(2)]
            public string Service { get; set; }
            [FieldOrder(3)]
            public string Program { get; set; }
            [FieldOrder(4)]
            public string Pracitioner { get; set; }
            [FieldOrder(5)]
            public string ClientID { get; set; }
            [FieldOrder(6)]
            public int? Episode { get; set; }
            [FieldOrder(7)]
            public string GroupNumber { get; set; }
            [FieldOrder(8)]
            public int? NumberOfClients { get; set; }
            [FieldOrder(9)]
            public int? Duration { get; set; }
            [FieldOrder(10)]
            public string Location { get; set; }
            [FieldOrder(11)]
            public int? Units { get; set; }
            [FieldOrder(12)]
            public double? Cost { get; set; }
            [FieldOrder(13)]
            public string isBillable { get; set; }
            [FieldOrder(14)]
            public string costaff1 { get; set; }
            [FieldOrder(15)]
            public string costaff2 { get; set; }
            [FieldOrder(16)]
            public string refferStaff { get; set; }
            [FieldOrder(17)]
            public string refferProvider { get; set; }
            [FieldOrder(18)]
            public string MedDiag1 { get; set; }
            [FieldOrder(19)]
            public string MedDiag2 { get; set; }

            [FieldOrder(20)]
            public string Modifiers { get; set; }

            [FieldOrder(21)]
            public string Service_Start_Time { get; set; }
            [FieldOrder(22)]
            public string Service_End_Time { get; set; }
            [FieldOrder(23)]
            public string Tooth_Number { get; set; }
            [FieldOrder(24)]
            public string Tooth_Status_Code { get; set; }
            [FieldOrder(25)]
            public string Tooth_Surface_Code { get; set; }
            [FieldOrder(26)]
            public string Oral_Cavity_Designation_Code { get; set; }
            [FieldOrder(27)]
            public string Prosthesis_Crown_Or_Inlay_Code { get; set; }
            [FieldOrder(28)]
            public string Prior_Placement_Date { get; set; }
            [FieldOrder(29)]
            public string Note_Reference_Text_837_Billing { get; set; }
            [FieldOrder(30)]
            public string Note_Reference_Code_837_Billing { get; set; }
            [FieldOrder(31)]
            public string Appointment_Date { get; set; }
            [FieldOrder(32)]
            public string Appointment_Start_Time { get; set; }
            [FieldOrder(33)]
            public string Appointment_End_Time { get; set; }
            [FieldOrder(34)]
            public string Appointment_Site { get; set; }
            [FieldOrder(35)]
            public string Appointment_Status { get; set; }
            [FieldOrder(36)]
            public string Appointment_Notes { get; set; }
            [FieldOrder(37)]
            public string Appointment_Confirmed { get; set; }
            [FieldOrder(38)]
            public string Confirmation_Notes { get; set; }
            [FieldOrder(39)]
            public string Missed_Visit { get; set; }
            [FieldOrder(40)]
            public string Missed_Visit_Service_Code { get; set; }
            [FieldOrder(41)]
            public string Missed_Visit_Notes { get; set; }
            [FieldOrder(42)]
            public string Medical_Diagnosis_3 { get; set; }
            [FieldOrder(43)]
            public string Medical_Diagnosis_4 { get; set; }
            [FieldOrder(44)]
            public string Discipline { get; set; }
            [FieldOrder(45)]
            public string Service_Status { get; set; }
            [FieldOrder(46)]
            public string Initial_Treatment_Date_2300DTP03 { get; set; }
            [FieldOrder(47)]
            public string Date_Of_Accident_2300DTP03 { get; set; }
            [FieldOrder(48)]
            public string Form_Locator_10_Auto_Accident_Place { get; set; }
            [FieldOrder(49)]
            public string Related_Causes_Information_2300CLM11 { get; set; }
            [FieldOrder(50)]
            public string First_Billing_Date { get; set; }
            [FieldOrder(51)]
            public string Additional_Service_Information { get; set; }
            [FieldOrder(52)]
            public string Psychotherapy_AddOn_Duration { get; set; }
            [FieldOrder(53)]
            public string AddOn_Notes { get; set; }
            [FieldOrder(54)]
            public string ICD10_Diagnosis_Code_1 { get; set; }
            [FieldOrder(55)]
            public string ICD10_Diagnosis_Code_2 { get; set; }
            [FieldOrder(56)]
            public string ICD10_Diagnosis_Code_3 { get; set; }
            [FieldOrder(57)]
            public string ICD10_Diagnosis_Code_4 { get; set; }
            [FieldOrder(58)]
            public string Reserved_for_future_development { get; set; }
            [FieldOrder(59)]
            public string Reserved_for_future_development1 { get; set; }
            [FieldOrder(60)]
            public string Reserved_for_future_development2 { get; set; }
            [FieldOrder(61)]
            public string Reserved_for_future_development3 { get; set; }
            [FieldOrder(62)]
            public string Reserved_for_future_development4 { get; set; }
            [FieldOrder(63)]
            public string Reserved_for_future_development5 { get; set; }
            [FieldOrder(64)]
            public string Reserved_for_future_development6 { get; set; }
```

```
        [FieldOrder(65)]
        public string Reserved_for_future_development7 { get; set; }
        [FieldOrder(66)]
        public string Reserved_for_future_development8 { get; set; }
        [FieldOrder(67)]
        public string Reserved_for_future_development9 { get; set; }
        [FieldOrder(68)]
        public string Reserved_for_future_development10 { get; set; }
        [FieldOrder(69)]
        public string Reserved_for_future_development11 { get; set; }
        [FieldOrder(70)]
        public string Reserved_for_future_development12 { get; set; }
        [FieldOrder(71)]
        public string Reserved_for_future_development13 { get; set; }
        [FieldOrder(72)]
        public string Reserved_for_future_development14 { get; set; }
        [FieldOrder(73)]
        public string Reserved_for_future_development15 { get; set; }
        [FieldOrder(74)]
        public string Reserved_for_future_development16 { get; set; }
        [FieldOrder(75)]
        public string Reserved_for_future_development17 { get; set; }
        [FieldOrder(76)]
        public string Reserved_for_future_development18 { get; set; }
        [FieldOrder(77)]
        public string Reserved_for_future_development19 { get; set; }
        [FieldOrder(78)]
        public string Reserved_for_future_development20 { get; set; }
        [FieldOrder(79)]
        public string Reserved_for_future_development21 { get; set; }
        [FieldOrder(80)]
        public string Reserved_for_future_development22 { get; set; }
        [FieldOrder(81)]
        public string Reserved_for_future_development23 { get; set; }
        [FieldOrder(82)]
        public string Reserved_for_future_development24 { get; set; }
        [FieldOrder(83)]
        public string Reserved_for_future_development25 { get; set; }
        [FieldOrder(84)]
        public string Reserved_for_future_development26 { get; set; }
        [FieldOrder(85)]
        public string Reserved_for_future_development27 { get; set; }
        [FieldOrder(86)]
        public string Reserved_for_future_development28 { get; set; }
        [FieldOrder(87)]
        public string Reserved_for_future_development29 { get; set; }
        [FieldOrder(88)]
        public string Reserved_for_future_development30 { get; set; }
        [FieldOrder(89)]
        public string Reserved_for_future_development31 { get; set; }
        [FieldOrder(90)]
        public string Reserved_for_future_development32 { get; set; }
        [FieldOrder(91)]
        public string Reserved_for_future_development33 { get; set; }
        [FieldOrder(92)]
        public string Reserved_for_future_development34 { get; set; }
        [FieldOrder(93)]
        public string Reserved_for_future_development35 { get; set; }
        [FieldOrder(94)]
        public string Reserved_for_future_development36 { get; set; }
        [FieldOrder(95)]
        public string Reserved_for_future_development37 { get; set; }
        [FieldOrder(96)]
        public string Reserved_for_future_development38 { get; set; }
        [FieldOrder(97)]
        public string Reserved_for_future_development39 { get; set; }
        [FieldOrder(98)]
        public string Reserved_for_future_development40 { get; set; }
        [FieldOrder(99)]
        public string Reserved_for_future_development41 { get; set; }
        [FieldOrder(100)]
        public string SS_Treatment_Free_Text_10_1 { get; set; }
        [FieldOrder(101)]
        public string SS_Treatment_Free_Text_10_2 { get; set; }
        [FieldOrder(102)]
        public string SS_Treatment_Free_Text_10_3 { get; set; }
        [FieldOrder(103)]
        public string SS_Treatment_Free_Text_10_4 { get; set; }
        [FieldOrder(104)]
        public string SS_Treatment_Free_Text_10_5 { get; set; }
        [FieldOrder(105)]
        public string SS_Treatment_Free_Text_40_1 { get; set; }
        [FieldOrder(106)]
        public string SS_Treatment_Free_Text_40_2 { get; set; }
        [FieldOrder(107)]
        public string SS_Treatment_Free_Text_40_3 { get; set; }
        [FieldOrder(108)]
        public string SS_Treatment_Free_Text_40_4 { get; set; }
        [FieldOrder(109)]
        public string SS_Treatment_Free_Text_40_5 { get; set; }
        [FieldOrder(110)]
        public string SS_Treatment_Free_Text_80_1 { get; set; }
        [FieldOrder(111)]
        public string SS_Treatment_Free_Text_80_2 { get; set; }
        [FieldOrder(112)]
        public string SS_Treatment_Free_Text_80_3 { get; set; }
        [FieldOrder(113)]
        public string SS_Treatment_Free_Text_80_4 { get; set; }
        [FieldOrder(114)]
        public string SS_Treatment_Free_Text_80_5 { get; set; }
        [FieldOrder(115)]
        public string SS_Treatment_Scrolling_Free_Text_1 { get; set; }
        [FieldOrder(116)]
        public string SS_Treatment_Scrolling_Free_Text_2 { get; set; }
        [FieldOrder(117)]
        public string SS_Treatment_Scrolling_Free_Text_3 { get; set; }
        [FieldOrder(118)]
        public string SS_Treatment_Scrolling_Free_Text_4 { get; set; }
        [FieldOrder(119)]
        public string SS_Treatment_Scrolling_Free_Text_5 { get; set; }
        [FieldOrder(120)]
        public string SPI { get; set; }
        [FieldOrder(121)]
        public string SS_Treatment_Dictionary_2 { get; set; }
        [FieldOrder(122)]
        public string SS_Treatment_Dictionary_3 { get; set; }
        [FieldOrder(123)]
        public string SS_Treatment_Dictionary_4 { get; set; }
        [FieldOrder(124)]
        public string SS_Treatment_Dictionary_5 { get; set; }
        [FieldOrder(125)]
        public string SS_Treatment_Dictionary_6 { get; set; }
        [FieldOrder(126)]
        public string SS_Treatment_Dictionary_7 { get; set; }
        [FieldOrder(127)]
        public string SS_Treatment_Dictionary_8 { get; set; }
        [FieldOrder(128)]
        public string SS_Treatment_Dictionary_9 { get; set; }
        [FieldOrder(129)]
        public string SS_Treatment_ICD9_Code_1 { get; set; }
        [FieldOrder(130)]
        public string SS_Treatment_ICD9_Code_2 { get; set; }
        [FieldOrder(131)]
        public string SS_Treatment_ICD9_Code_3 { get; set; }
        [FieldOrder(132)]
        public string SS_Treatment_ICD9_Code_4 { get; set; }
        [FieldOrder(133)]
        public string SS_Treatment_ICD9_Code_5 { get; set; }
        [FieldOrder(134)]
        public string SS_Treatment_Date_1 { get; set; }
        [FieldOrder(135)]
        public string SS_Treatment_Date_2 { get; set; }
        [FieldOrder(136)]
        public string SS_Treatment_Date_3 { get; set; }
        [FieldOrder(137)]
        public string SS_Treatment_Date_4 { get; set; }
        [FieldOrder(138)]
        public string SS_Treatment_Date_5 { get; set; }
        [FieldOrder(139)]
        public string SS_Treatment_Practitioner_1 { get; set; }
        [FieldOrder(140)]
        public string SS_Treatment_Practitioner_2 { get; set; }
        [FieldOrder(141)]
        public string SS_Treatment_Practitioner_3 { get; set; }
        [FieldOrder(142)]
        public string SS_Treatment_Practitioner_4 { get; set; }
        [FieldOrder(143)]
        public string SS_Treatment_Practitioner_5 { get; set; }
        [FieldOrder(144)]
        public string SS_Treatment_Guarantor_1 { get; set; }
        [FieldOrder(145)]
        public string SS_Treatment_Guarantor_2 { get; set; }
        [FieldOrder(146)]
        public string SS_Treatment_Guarantor_3 { get; set; }
        [FieldOrder(147)]
        public string SS_Treatment_Guarantor_4 { get; set; }
        [FieldOrder(148)]
        public string SS_Treatment_Guarantor_5 { get; set; }
        [FieldOrder(149)]
        public string SS_Treatment_DSM_Code_1 { get; set; }
        [FieldOrder(150)]
        public string SS_Treatment_DSM_Code_2 { get; set; }
        [FieldOrder(151)]
        public string SS_Treatment_DSM_Code_3 { get; set; }
        [FieldOrder(152)]
        public string SS_Treatment_DSM_Code_4 { get; set; }
        [FieldOrder(153)]
        public string SS_Treatment_DSM_Code_5 { get; set; }
        [FieldOrder(154)]
        public string SS_Treatment_Multiple_Select_Dictionary_1 { get; set; }
        [FieldOrder(155)]
        public string SS_Treatment_Multiple_Select_Dictionary_2 { get; set; }
        [FieldOrder(156)]
        public string Guarantor_1 { get; set; }
        [FieldOrder(157)]
        public string Amount1 { get; set; }
        [FieldOrder(158)]
        public string Guarantor_2 { get; set; }
        [FieldOrder(159)]
        public string Amount2 { get; set; }
        [FieldOrder(160)]
        public string Guarantor_3 { get; set; }
        [FieldOrder(161)]
        public string Amount3 { get; set; }
        [FieldOrder(162)]
        public string Guarantor_4 { get; set; }
        [FieldOrder(163)]
        public string Amount4 { get; set; }
        [FieldOrder(164)]
        public string Guarantor_5 { get; set; }
        [FieldOrder(165)]
        public string Amount5 { get; set; }
        [FieldOrder(166)]
        public string Guarantor_6 { get; set; }
        [FieldOrder(167)]
        public string Amount6 { get; set; }
        [FieldOrder(168)]
        public string Guarantor_7 { get; set; }
        [FieldOrder(169)]
        public string Amount7 { get; set; }
        [FieldOrder(170)]
        public string Guarantor_8 { get; set; }
        [FieldOrder(171)]
        public string Amount8 { get; set; }
        [FieldOrder(172)]
        public string Guarantor_9 { get; set; }
        [FieldOrder(173)]
        public string Amount9 { get; set; }
        [FieldOrder(174)]
        public string Guarantor_10 { get; set; }
        [FieldOrder(175)]
        public string Amount10 { get; set; }
        [FieldOrder(176)]
        public string SS_Treatment_Integer_1 { get; set; }
        [FieldOrder(177)]
        public string SS_Treatment_Integer_2 { get; set; }
        [FieldOrder(178)]
        public string SS_Treatment_Integer_3 { get; set; }
        [FieldOrder(179)]
        public string SS_Treatment_Integer_4 { get; set; }
        [FieldOrder(180)]
        public string SS_Treatment_Integer_5 { get; set; }
        [FieldOrder(181)]
        public string SS_Treatment_Integer_6 { get; set; }
        [FieldOrder(182)]
        public string IncidentTo_Practitioner { get; set; }



    }

    enum month { January = 1, February, March, April, May, June, July, August,
    September, October, November, December }




}

```

```
using FileHelpers;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Avatar_Solutions
{
    public partial class LoginForm : Form
    {
        public LoginForm()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

            this.combo_systemCodes.Text = "LIVE";
        }

        private void button1_Click(object sender, EventArgs e)
        {
            AvatarConnection con = new AvatarConnection(combo_systemCodes.Text,
            txt_Username.Text, txt_pwd.Text);

            if (con.opened)
            {
                this.Hide();

                MainMenu frm = new MainMenu(con);
                frm.ShowDialog();

                if (frm.DialogResult == DialogResult.Retry)
                {
                    this.Show();
                    frm = null;
                } else
                {
                    this.Close();
                }
            }



        }

        private void combo_systemCodes_Validating(object sender, CancelEventArgs e)
        {
            if (combo_systemCodes.Text == "" || combo_systemCodes.Text == null)
            {
                errorProvider1.SetError(combo_systemCodes, "System Code Required");
                combo_systemCodes.Focus();
                e.Cancel = true;

            } else
            {
                errorProvider1.SetError(combo_systemCodes, "");
            }
        }

        private void txt_Username_Validating(object sender, CancelEventArgs e)
        {
            if (txt_Username.Text == "" || txt_Username.Text == null)
```

```csharp
                {
                    errorProvider1.SetError(txt_Username, "Username Required");
                    ((Control)sender).Focus();
                    e.Cancel = true;
                }
                else
                {
                    errorProvider1.SetError(txt_Username, "");
                }
            }

            private void txt_pwd_Validating(object sender, CancelEventArgs e)
            {
                if (txt_pwd.Text == "" || txt_pwd.Text == null)
                {
                    errorProvider1.SetError(txt_pwd, "Password Required");
                    ((Control)sender).Focus();
                    e.Cancel = true;

                }
                else
                {
                    errorProvider1.SetError(txt_pwd, "");
                }
            }

            private void button2_Click(object sender, EventArgs e)
            {
                /*
                GL.GL sys = new GL.GL("wallace4");
                sys.GetPrograms();
                MessageBox.Show(sys.Rus.Count.ToString());
                */
            }

            private void button3_Click(object sender, EventArgs e)
            {
             /*   AvatarConnection con = new AvatarConnection("LIVE", "wallace4","Dw832806");
                con.conn.Open();
                con.GetStaff();
              //  MessageBox.Show(con.staffList.Count.ToString());



                //MessageBox.Show(con.GetTotalPracNumber().ToString());

                con.conn.Close();*/
            }

            private void backgroundWorker1_DoWork(object sender, DoWorkEventArgs e)
            {
                //con.GetPracNumbersRecords();

                e.Result = true;
            }
        }
    }
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.Odbc;
using System.Net.Mail;
namespace Avatar_Solutions
{
    public partial class Processor : Form
    {
        public Processor()
        {
            InitializeComponent();
        }

        private AvatarConnection avConn;
        private OdbcConnection conn;
        private OdbcCommand cmd;
        private OdbcDataAdapter DA;
        DataTable dtbl;

        private List<Report> rptList;
        private List<ServiceRecord> srvRecord;

        int TotalCreated = 0;
        int TotalError = 0;
        int TotalAlreadyCreated = 0;

        public Processor(AvatarConnection con)
        {
            InitializeComponent();
            avConn = con;
            dtbl = new DataTable();
            conn = avConn.conn;

            srvRecord = new List<ServiceRecord>();
            rptList = new List<Report>();


            DA = new OdbcDataAdapter();

            dataGridView1.DataSource = dtbl;


        }

        private void GetData()
        {
            DateTime FromDate = DateTime.Parse(((int)comboBox1.SelectedItem).ToString()
            + "/01/" + comboBox2.Text);
            DateTime ThruDate = FromDate.AddMonths(1).AddDays(-1);

            string sendFrom = FromDate.ToString("yyyy-MM-dd");
            string sendThru = ThruDate.ToString("yyyy-MM-dd");
            //string sendFrom = From.Value.ToString("yyyy-MM-dd");
            //string sendThru = ThruDate.Value.ToString("yyyy-MM-dd");
            //MessageBox.Show("From: " + sendFrom + Environment.NewLine + "Thru: " +
            sendThru);
            DA.SelectCommand = avConn.SQLCaseMgmt(sendFrom, sendThru,
            avConn.Componets,avConn.ProgramExclude);
            DA.SelectCommand.Connection = conn;
            DA.Fill(dtbl);
        }

```

```csharp
        private void ProcessData()
        {
            int ReqFTF = avConn.ReqFTF;
            int ReqTot = avConn.ReqTot;
            string billableCode = avConn.BillableCode;

            //This Code Generates Billable Charges
            int NonFTF = 0;
            int FTF = 0;
            int Total = 0;
            bool FoundBillable = false;
            bool started = true;
            string PreviousClient = "";


            int Episode = 0;
            int PrvEpisode = 0;
            string currentClient = "";
            string Service = "";
            int Tot = 0;
            string Program = "";
            string StaffID = "";
            DateTime DOS = DateTime.Now;

            string PrvService = "";
            //int Tot = 0;
            string PrvProgram = "";
            string PrvStaffID = "";
            DateTime PrvDOS = DateTime.Now;

            int i = 0;
            foreach(DataRow row in dtbl.Rows)
            {
                i++;

                currentClient = (string)row["PATID"];


                if (i == dtbl.Rows.Count || started)
                {
                    PreviousClient = currentClient;
                    started = false;

                    Service = (string)row["Service"];
                    Tot = (int)row["Tot"];
                    Program = (string)row["Program"];
                    StaffID = (string)row["STAFFID"];
                    Episode = (int)row["EPISODE"];
                    if (row["DOS"] != DBNull.Value)
                    {
                        DOS = Convert.ToDateTime(row["DOS"]);
                    }
                    else
                    {
                        throw new Exception("No DateTime");
                    }
                }

     /*           if (avConn.CurrentRuleTitle == "ACT")
                {
                    started = false;
                }*/

               if (i == dtbl.Rows.Count)
                {
                    currentClient = "";
                    FoundBillable = false;
                    FTF = 0;
                    Total = 0;
                    NonFTF = 0;

                    if (Service == billableCode)
                    {
                        FoundBillable = true;

                    }
                    else if (Service[Service.Length - 1] == 'F')
                    {
                        FTF += Tot;
                        Total += Tot;
                    }
                    else
                    {
                        NonFTF += Tot;
                        Total += Tot;
                    }

                    PrvDOS = DOS;
                    PrvProgram = Program;
                    PrvService = Service;
                    PrvStaffID = StaffID;
                    PrvEpisode = (int)row["EPISODE"];

                }


                if (PreviousClient != currentClient)
                {

                       //If override staff logic then change staff id HERE since always hit.
                       if (avConn.OverrideStaff)
                       {
                        PrvStaffID = avConn.OverrideStaffID;
                       }
                       //end staff override logic

                    if (!FoundBillable && FTF >= ReqFTF && Total >= ReqTot)
                    {
                        //GenerateCharge
                        rptList.Add(new Report(DOS, PreviousClient, FTF, Total, true, "Charge Created"));
                        avConn.ServList.Add(new ServiceRecord(PrvDOS, PrvEpisode, billableCode, PrvProgram, PrvStaffID, PreviousClient, avConn.currentServiceType));
                        TotalCreated += 1;
                    }
                    else if (FoundBillable)
                    {
                        //WasAlreadyCharges
                        rptList.Add(new Report(DOS, PreviousClient, FTF, Total, false, "Already Created"));
                        TotalAlreadyCreated += 1;
                    } else if (FTF >= ReqFTF && Total < ReqTot)
                    {
                        //Not Enough Services
                        rptList.Add(new Report(DOS, PreviousClient, FTF, Total, false, "Not Enough Services"));
                        TotalError += 1;
                    }
                    else if (FTF <= ReqFTF)
                    {
                        //Not Enough FTF
                        rptList.Add(new Report(DOS, PreviousClient, FTF, Total, false, "Not Enough FTF"));
                        TotalError += 1;
                    }
                    else
                    {
                        //Other Error
                        rptList.Add(new Report(DOS, PreviousClient, FTF, Total, false,
                        "Other Err"));
                        TotalError += 1;
                    }



                    NonFTF = 0;
                    FTF = 0;
                    Total = 0;
                    FoundBillable = false;

                }

                //Always runs -- was miscounted
                Service = (string)row["Service"];
                Tot = (int)row["Tot"];
                Program = (string)row["Program"];
                StaffID = (string)row["STAFFID"];


                if (row["DOS"] != DBNull.Value)
                {
                    DOS = Convert.ToDateTime(row["DOS"]);
                }
                else
                {
                    throw new Exception("No DateTime");
                }
                if (Service == billableCode)
                    {
                        FoundBillable = true;

                    } else if (Service[Service.Length - 1] == 'F')
                    {
                        FTF += Tot;
                        Total += Tot;
                    } else
                    {
                        NonFTF += Tot;
                        Total += Tot;
                    }

                PrvDOS = DOS;
                PrvProgram = Program;
                PrvService = Service;
                PrvStaffID = StaffID;
                PreviousClient = currentClient;
                PrvEpisode = (int)row["EPISODE"];
            }


        }

        private void Test_Test_Load(object sender, EventArgs e)
        {
            this.Text += " Rule: " + avConn.CurrentRuleTitle + " Type: " +
            avConn.currentServiceType.ToString() + " Billable Code: " +
            avConn.BillableCode;
            this.comboBox1.DataSource = Enum.GetValues(typeof(Avatar_Solutions.month));

            comboBox2.Text = DateTime.Today.ToString("yyyy");
            comboBox1.SelectedIndex = DateTime.Today.Month - 2;

        }

        private void button1_Click(object sender, EventArgs e)
        {
            dtbl.Clear();
            this.GetData();
            comboBox1.SelectedIndex += 1;
        }

        private void button2_Click(object sender, EventArgs e)
        {
            this.ProcessData();

            Report_Viewer frm = new
            Report_Viewer(rptList,TotalCreated,TotalAlreadyCreated,TotalError,avConn.Curr
            entRuleTitle,From.Value, ThruDate.Value);
            frm.ShowDialog();

           // SendMessage();

        }


    }
}
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<PreprocessorRules>
  <PreprocessorRule>

    <Name>Case Mgmt (Reg)</Name>
    <Billable>T2023</Billable>
    <Componets>
      <Componet FTF="N">770</Componet>
      <Componet FTF="Y">770F</Componet>
      <Componet FTF="N">771</Componet>
      <Componet FTF="Y">771F</Componet>
      <Componet FTF="N">780</Componet>
      <Componet FTF="Y">780F</Componet>
      <Componet FTF="N">790</Componet>
      <Componet FTF="Y">790F</Componet>
    </Componets>
    <ExcludePrograms>
      <ExcludeProgram>9105</ExcludeProgram>
      <ExcludeProgram>9115</ExcludeProgram>
      <ExcludeProgram>9350</ExcludeProgram>
      <ExcludeProgram>9712</ExcludeProgram>
      <ExcludeProgram>9963</ExcludeProgram>
      <ExcludeProgram>9980</ExcludeProgram>
      <ExcludeProgram>9385</ExcludeProgram>
    </ExcludePrograms>
    <Rules>
      <FTF>2</FTF>
      <Total>4</Total>
      <ServiceType>1</ServiceType>
    </Rules>

  </PreprocessorRule>

  <PreprocessorRule>

    <Name>Case Mgmt (Complex)</Name>
    <Billable>T2023TG</Billable>
    <Componets>
      <Componet FTF="N">777</Componet>
      <Componet FTF="Y">777F</Componet>
      <Componet FTF="N">778</Componet>
      <Componet FTF="Y">778F</Componet>
      <Componet FTF="N">781</Componet>
      <Componet FTF="Y">781F</Componet>
      <Componet FTF="N">791</Componet>
      <Componet FTF="Y">791F</Componet>
    </Componets>
    <ExcludePrograms>
      <ExcludeProgram>9105</ExcludeProgram>
      <ExcludeProgram>9115</ExcludeProgram>
      <ExcludeProgram>9350</ExcludeProgram>
      <ExcludeProgram>9712</ExcludeProgram>
      <ExcludeProgram>9963</ExcludeProgram>
      <ExcludeProgram>9980</ExcludeProgram>
      <ExcludeProgram>9385</ExcludeProgram>
    </ExcludePrograms>
    <Rules>
      <FTF>3</FTF>
      <Total>5</Total>
      <ServiceType>1</ServiceType>
    </Rules>

  </PreprocessorRule>

  <PreprocessorRule>

    <Name>ACT</Name>
```

```xml
      <Billable>H0040</Billable>
      <Componets>
        <Componet FTF="N">863</Componet>
        <Componet FTF="Y">863F</Componet>
      </Componets>

      <IncludePrograms>
        <IncludeProgram>9525</IncludeProgram>
      </IncludePrograms>

      <Rules>
        <FTF>1</FTF>
        <Total>10</Total>
        <ServiceType>0</ServiceType>
      </Rules>

    </PreprocessorRule>



    <PreprocessorRule>
      <Name>Waiver (Case Mgmt)</Name>
      <Billable>T2022</Billable>
      <Componets>
        <Componet FTF="N">770</Componet>
        <Componet FTF="Y">770F</Componet>
        <Componet FTF="N">771</Componet>
        <Componet FTF="Y">771F</Componet>
        <Componet FTF="N">780</Componet>
        <Componet FTF="Y">780F</Componet>
        <Componet FTF="N">790</Componet>
        <Componet FTF="Y">790F</Componet>
      </Componets>
      <IncludeFSs>
        <IncludeFS>1121</IncludeFS>
        <IncludeFS>1126</IncludeFS>
        <IncludeFS>1145</IncludeFS>
        <IncludeFS>1146</IncludeFS>
      </IncludeFSs>
      <Rules>
        <FTF>0</FTF>
        <Total>1</Total>
        <OverrideStaff>9995</OverrideStaff>
        <ServiceType>3</ServiceType>
      </Rules>
    </PreprocessorRule>


    <PreprocessorRule>
      <Name>Waiver (Fin. Mgmt)</Name>
      <Billable>T2040</Billable>

      <IncludeFSs>
        <IncludeFS>1121</IncludeFS>
        <IncludeFS>1146</IncludeFS>
      </IncludeFSs>

      <Rules>
        <FTF>0</FTF>
        <Total>1</Total>
        <OverrideStaff>9995</OverrideStaff>
        <ServiceType>3</ServiceType>
      </Rules>
    </PreprocessorRule>

  <!--
    <PreprocessorRule>
```

```xml
      <Name>Pasarr (140)</Name>
      <Billable>T2022</Billable>

      <IncludeFSs>
        <IncludeFS>1128</IncludeFS>
      </IncludeFSs>

      <IncludePrograms>
        <IncludeProgram>2530</IncludeProgram>
      </IncludePrograms>

      <Rules>
        <FTF>0</FTF>
        <Total>1</Total>
        <OverrideStaff>9995</OverrideStaff>
        <ServiceType>3</ServiceType>
      </Rules>
    </PreprocessorRule>
  -->
    <PreprocessorRule>
      <Name>IDD (140)</Name>
      <Billable>T2022</Billable>

      <IncludeFSs>
        <IncludeFS>1071</IncludeFS>
      </IncludeFSs>

      <IncludePrograms>
        <IncludeProgram>2530</IncludeProgram>
        <IncludeProgram>2330</IncludeProgram>
        <IncludeProgram>3330</IncludeProgram>
        <IncludeProgram>4330</IncludeProgram>
        <IncludeProgram>5330</IncludeProgram>
        <IncludeProgram>6330</IncludeProgram>
        <IncludeProgram>7330</IncludeProgram>
        <IncludeProgram>8330</IncludeProgram>
      </IncludePrograms>

      <Rules>
        <FTF>0</FTF>
        <Total>1</Total>
        <OverrideStaff>9995</OverrideStaff>
        <ServiceType>3</ServiceType>
      </Rules>
    </PreprocessorRule>



  </PreprocessorRules>
```