

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using FileHelpers;
7  using System.Data.Odbc;
8  using System.Xml;
9  using System.Xml.Linq;
10
11
12 namespace Avatar_Solutions
13 {
14     public class AvatarConnection
15     {
16         public IEnumerable<XElement> Rules;
17
18         public IEnumerable<XElement> currentRule;
19         public string CurrentRuleTitle;
20         public string BillableCode;
21         public List<string> Componets;
22         public List<string> ServiceExclude;
23         public List<string> FSExclude;
24         public List<string> FSInclude;
25         public List<string> ProgramExclude;
26         public List<string> ProgramInclude;
27         public int ReqFTF;
28         public int ReqTot;
29         public List<ServiceRecord> ServList { get; set; }
30         public bool OverrideStaff { get; set; }
31         public string OverrideStaffID { get; set; }
32         public ServiceRecord.ServiceType currentServiceType { get; set; }
33
34         OdbcConnectionStringBuilder connBuilder;
35         public OdbcConnection conn { get; set; }
36         public bool opened { get; private set; }
37         public string uname;
38
39         public List<BillableProvider> staffList;
40         public List<PractitionerRecord> staffPracGuarNumb;
41
42         [DelimitedRecord("\t")]
43         public class PractitionerRecord
44         {
45             public string Practitioner { get; set; }
46             public string GuarantorID { get; set; }
47             public string Program { get; set; }
48             public string EffectiveDate { get; set; }
49             public string ExpirationDate { get; set; }
50             public string NPINumber { get; set; }
51             public string TaxonomyCode { get; set; }
52             public string SecondaryIdentificationQualifier { get; set; }
53             public string SecondaryIdentificationNumber { get; set; }
54             public string SupervisingPractitioner { get; set; }
55             public string ReferenceIdentificationQualifier_837P2010ABREF01 { get; set; }
56             public string PaytoProviderAdditionalIdentifier_837P2010ABREF02 { get; set; }
57             public string ClaimNoteReferenceCode_837P2300NTE01 { get; set; }
58             public string ClaimNoteText_837P2300NTE02 { get; set; }
59             public string ServiceFacilityNameLastorOrganizationName_837P2310D2420CNM103
60             { get; set; }
61             public string
62             ServiceFacilityIdentificationCodeQualifier_837P2310D2420CNM108 { get; set; }
63             public string ServiceFacilityIdentificationCode_837P2310D2420CNM109 { get;
64             set; }
65             public string ServiceFacilityAddressInformation_837P2310D2420CN301 { get;
66             set; }
67             public string ServiceFacilityZipCode_837P2310D2420CN403 { get; set; }
68             public string ServiceFacilityCity_837P2310D2420CN401 { get; set; }
69             public string ServiceFacilityState_837P2310D2420CN402 { get; set; }

```

```

66     public string
        ServiceFacilityReferenceIdentificationQualifier_837P2310D2420CREF01 { get;
        set; }
67     public string ServiceFacilityReferenceIdentification_837P2310D2420CREF02 {
        get; set; }
68     public string ServiceNoteReferenceCode_837P2400NTE01 { get; set; }
69     public string ServiceNoteText_837P2400NTE02 { get; set; }
70     public string ClaimNoteReferenceCode_837I2300NTE01 { get; set; }
71     public string ClaimNoteText_837I2300NTE02 { get; set; }
72     public string BillingNoteReferenceCode_837I2300NTE01 { get; set; }
73     public string BillingNoteText_837I2300NTE02 { get; set; }
74     public string ClaimNoteReferenceCode_837D2300NTE01 { get; set; }
75     public string ClaimNoteText_837D2300NTE02 { get; set; }
76     public string ServiceNoteReferenceCode_837D2400NTE01 { get; set; }
77     public string ServiceNoteText_837D2400NTE02 { get; set; }
78     public string
        PractitionerSecondaryIdentificationQualifier2_837ElectronicBilling { get;
        set; }
79     public string PractitionerSecondaryIdentifier2_837ElectronicBilling { get;
        set; }
80     public string ReferenceIdentificationQualifier_837I2330DREF01 { get; set; }
81     public string ReferenceIdentification_837I2330DREF02 { get; set; }
82     public string
        PractitionerBillingQualifierOther_FormLocator17a_HCFA1500NPIVersionOnly {
        get; set; }
83     public string
        PractitionerBillingOther_FormLocator17a_HCFA1500NPIVersionOnly { get; set; }
84     public string PractitionerBillingNPI_FormLocator17b_HCFA1500NPIVersionOnly
        { get; set; }
85     public string SupplementalInformation_FormLocator24D_HCFA1500NPIVersionOnly
        { get; set; }
86     public string PractitionerBillingNPI_FormLocator24J_HCFA1500NPIVersionOnly
        { get; set; }
87     public string AdditionalProviderID_ADAFormLocator58 { get; set; }
88     public string
        PractitionerBillingQualifierOther_FormLocator33B_HCFA1500NPIVersionOnly {
        get; set; }
89     public string
        PractitionerBillingOther_FormLocator33B_HCFA1500NPIVersionOnly { get; set; }
90     public string Comments { get; set; }
91
92     public PractitionerRecord()
93     {
94
95     }
96 }
97
98
99
100
101 public class BillableProvider
102 {
103     public string name { get; set; }
104     public string ID { get; set; }
105     public string Taxonomy { get; set; }
106     public string npa { get; set; }
107
108     public BillableProvider(string n, string id, string tax, string npa)
109     {
110         name = n;
111         ID = id;
112         Taxonomy = tax;
113         npa = npa;
114     }
115
116     public BillableProvider()
117     {
118
119     }

```



```

186         name = rdr["name"].ToString(),
187         npi = rdr["npi_number"].ToString(),
188         Taxonomy = rdr["taxonomy_code"].ToString()
189     });
190     }
191
192     }
193
194
195
196     }
197 }
198
199
200 public decimal GetTotalPracNumber()
201 {
202     string stm = "select count(*) as total from SYSTEM.table_prac_num_guar_prog";
203     decimal total;
204     using (OdbcCommand cmd = new OdbcCommand(stm, conn))
205     {
206
207         using (OdbcDataReader rdr = cmd.ExecuteReader())
208         {
209
210             rdr.Read();
211             total = decimal.Parse(rdr["total"].ToString());
212         }
213     }
214
215     return total;
216 }
217
218 public void GetPracNumbersRecords()
219 {
220     string stm = "SELECT claim_note_text_837d, claim_note_ref_837d_code,
221     bill_note_text_837i, bill_note_ref_837i_code, svc_fac_name_837p, STAFF_ID,
222     GUARANTOR_ID, program_code, effective_date, expiration_date, npi_number,
223     taxonomy_code, second_ident2_qual_code, second_ident2_number,
224     supervising_pract_id, ref_id_qua_837p_code, ref_id_837p,
225     claim_note_ref_837p_code, claim_note_text_837p, svc_fac_id_qua_code_837p,
226     svc_fac_address_837p, svc_fac_zip_837p, svc_fac_city_837p,
227     svc_fac_state_837p, svc_fac_ref_id_code_837p, svc_fac_ref_id_837p,
228     svc_note_ref_837p_code, svc_note_text_837p, claim_note_ref_837i_code,
229     claim_note_text_837i, svc_note_ref_837d_code, svc_note_text_837d,
230     second_ident2_qual_code, second_ident2_number, ref_id_qua_837i_code,
231     ref_id_837i, hcfa_npi_17a_qua_code, hcfa_npi_17a, hcfa_npi_17b,
232     hcfa_npi_loc_24d_supp, hcfa_npi_loc_33a_npi, add_prov_id_dental_fl58,
233     hcfa_npi_loc_33b_code, hcfa_npi_loc_33b_other, comments from
234     SYSTEM.table_prac_num_guar_prog";
235
236     using (OdbcCommand cmd = new OdbcCommand(stm, conn))
237     {
238         using (OdbcDataReader rdr = cmd.ExecuteReader())
239         {
240             while (rdr.Read())
241             {
242                 this.staffPracGuarNumb.Add(new PractitionerRecord
243                 {
244                     Practitioner = rdr["STAFF_ID"].ToString(),
245                     GuarantorID = rdr["GUARANTOR_ID"].ToString(),
246                     Program = rdr["program_code"].ToString(),
247                     EffectiveDate = rdr["effective_date"].ToString(),
248                     ExpirationDate = rdr["expiration_date"].ToString(),
249                     NPINumber = rdr["npi_number"].ToString(),
250                     TaxonomyCode = rdr["taxonomy_code"].ToString(),
251                     SecondaryIdentificationQualifier =
252                         rdr["second_ident2_qual_code"].ToString(),
253                     SecondaryIdentificationNumber =
254                         rdr["second_ident2_number"].ToString(),

```

```

239
240     SupervisingPractitioner =
241         rdr["supervising_pract_id"].ToString(),
242
243     ReferenceIdentificationQualifier_837P2010ABREF01 =
244         rdr["ref_id_qua_837p_code"].ToString(),
245
246     PaytoProviderAdditionalIdentifier_837P2010ABREF02 =
247         rdr["ref_id_837p"].ToString(),
248
249     ClaimNoteReferenceCode_837P2300NTE01 =
250         rdr["claim_note_ref_837p_code"].ToString(),
251
252     ClaimNoteText_837P2300NTE02 =
253         rdr["claim_note_text_837p"].ToString(),
254
255     ServiceFacilityNameLastorOrganizationName_837P2310D2420CNM103
256         = rdr["svc_fac_name_837p"].ToString(),
257
258     ServiceFacilityIdentificationCodeQualifier_837P2310D2420CNM10
259     8 = rdr["svc_fac_ref_id_code_837p"].ToString(),
260
261     ServiceFacilityIdentificationCode_837P2310D2420CNM109 =
262         rdr["svc_fac_ref_id_837p"].ToString(),
263
264     ServiceFacilityAddressInformation_837P2310D2420CN301 =
265         rdr["svc_fac_address_837p"].ToString(),
266
267     ServiceFacilityZipCode_837P2310D2420CN403 =
268         rdr["svc_fac_zip_837p"].ToString(),
269
270     ServiceFacilityCity_837P2310D2420CN401 =
271         rdr["svc_fac_city_837p"].ToString(),
272
273     ServiceFacilityState_837P2310D2420CN402 =
274         rdr["svc_fac_state_837p"].ToString(),
275
276     ServiceFacilityReferenceIdentificationQualifier_837P2310D2420
277     CREF01 = rdr["svc_fac_ref_id_code_837p"].ToString(),
278     ServiceFacilityReferenceIdentification_837P2310D2420CREF02
279     = rdr["svc_fac_ref_id_code_837p"].ToString(),
280     ServiceNoteReferenceCode_837P2400NTE01 =
281         rdr["svc_note_ref_837p_code"].ToString(),
282
283     ServiceNoteText_837P2400NTE02 =
284         rdr["svc_note_text_837p"].ToString(),
285
286     ClaimNoteReferenceCode_837I2300NTE01 =
287         rdr["claim_note_ref_837i_code"].ToString(),
288
289     ClaimNoteText_837I2300NTE02 =
290         rdr["claim_note_text_837i"].ToString(),
291
292     BillingNoteReferenceCode_837I2300NTE01 =
293         rdr["bill_note_ref_837i_code"].ToString(),
294
295     BillingNoteText_837I2300NTE02 =
296         rdr["bill_note_text_837i"].ToString(),
297
298     ClaimNoteReferenceCode_837D2300NTE01 = rdr["claim_note_ref_837d_code"].ToString(),
299
300     ClaimNoteText_837D2300NTE02 =
301         rdr["claim_note_text_837d"].ToString(),
302
303     ServiceNoteReferenceCode_837D2400NTE01 =
304         rdr["svc_note_ref_837d_code"].ToString(),
305
306     ServiceNoteText_837D2400NTE02 =

```

```

283         rdr["svc_note_text_837d"].ToString(),
284
285         PractitionerSecondaryIdentificationQualifier2_837ElectronicBi
286         lling = rdr["second_ident2_qual_code"].ToString(),
287
288         PractitionerSecondaryIdentifier2_837ElectronicBilling =
289         rdr["second_ident2_number"].ToString(),
290
291         ReferenceIdentificationQualifier_837I2330DREF01 =
292         rdr["ref_id_qua_837i_code"].ToString(),
293
294         ReferenceIdentification_837I2330DREF02 =
295         rdr["ref_id_837i"].ToString(),
296
297         PractitionerBillingQualifierOther_FormLocator17a_HCFA1500NPIV
298         ersionOnly = rdr["hcfa_npi_17a_qua_code"].ToString(),
299
300         PractitionerBillingOther_FormLocator17a_HCFA1500NPIVersionOnl
301         y = rdr["hcfa_npi_17a"].ToString(),
302
303         PractitionerBillingNPI_FormLocator17b_HCFA1500NPIVersionOnly
304         = rdr["hcfa_npi_17b"].ToString(),
305
306         SupplementalInformation_FormLocator24D_HCFA1500NPIVersionOnly
307         = rdr["hcfa_npi_loc_24d_supp"].ToString(),
308
309         PractitionerBillingNPI_FormLocator24J_HCFA1500NPIVersionOnly
310         = rdr["hcfa_npi_loc_33a_npi"].ToString(),
311
312         AdditionalProviderID_ADARFormLocator58 =
313         rdr["add_prov_id_dental_fl58"].ToString(),
314
315         PractitionerBillingQualifierOther_FormLocator33B_HCFA1500NPIV
316         ersionOnly = rdr["hcfa_npi_loc_33b_code"].ToString(),
317
318         PractitionerBillingOther_FormLocator33B_HCFA1500NPIVersionOnl
319         y = rdr["hcfa_npi_loc_33b_other"].ToString(),
320
321         Comments = rdr["comments"].ToString()
322     });
323 }
324
325 }
326
327 //return true;
328 }
329
330 public OdbcCommand SQLCaseMgmt(string FROM, string Thru, List<string>
331 Componets, List<string> ExcludePrograms = null, List<string> ExcludeFS = null)
332 {
333     OdbcCommand cmd = new OdbcCommand();
334     // cmd.CommandText = "SELECT PATID, Service, COUNT(*) as Tot, Program,
335     Staffid, DOS" +
336     //
337     //         " FROM pmhc_CaseMgmt " +
338     //         "WHERE ((DOS >= '" + FROM + "' and DOS <= '" + Thru + "')" +
339     //         "AND Service IN

```

```

329 ('770','770F','780','780F','790','790F','771','771F','T2023'))"+
330 //          "GROUP BY PATID, Service " +
331 //          "ORDER BY PATID";
332
333 StringBuilder bldr = new StringBuilder();
334 bldr.Append("SELECT PATID, Service, COUNT(*) as Tot, Program, Staffid, DOS,
335 Episode ");
336 bldr.Append(" FROM pmhc_CaseMgmt ");
337 bldr.Append("WHERE (DOS >= '" + FROM + "' and DOS <= '" + Thru + "' ");
338
339 if (Componets != null)
340 {
341     bldr.Append(IterateExclusion("Service", false, Componets, true));
342 }
343 if (ServiceExclude != null)
344 {
345     bldr.Append(IterateExclusion("Service", false, ServiceExclude, true));
346 }
347 if (ProgramExclude != null)
348 {
349     bldr.Append(IterateExclusion("Program", true, ProgramExclude));
350 }
351 if (ProgramInclude != null)
352 {
353     bldr.Append(IterateExclusion("Program", false, ProgramInclude));
354 }
355 if (ExcludeFS != null)
356 {
357     bldr.Append(IterateExclusion("Guar", true, ExcludeFS));
358 }
359 if (FSInclude != null)
360 {
361     bldr.Append(IterateExclusion("Guar", false, FSInclude));
362 }
363
364 bldr.Append(")");
365 bldr.Append(" GROUP BY PATID, Service ");
366 bldr.Append(" ORDER BY PATID");
367
368 cmd.CommandText = bldr.ToString();
369
370
371
372 return cmd;
373 }
374
375 string IterateExclusion(string exclusionField, bool notIn, List<string>
376 ExcludePrograms, bool isComponets = false)
377 {
378     StringBuilder bldr = new StringBuilder();
379
380     if (ExcludePrograms != null)
381     {
382         for (int i = 0; i < ExcludePrograms.Count; i++)
383         {
384             if (ExcludePrograms.Count != 1 && i == ExcludePrograms.Count - 1)
385             {
386                 if (isComponets)
387                 {
388                     bldr.Append(", '" + ExcludePrograms[i] + "', '" +
389 BillableCode + "'");
390                 } else
391                 {
392                     bldr.Append(", '" + ExcludePrograms[i] + "'");
393                 }
394             }
395         }
396     }
397 }

```

```

394         else if (i != 0)
395         {
396             bldr.Append(", '" + ExcludePrograms[i] + "'");
397         }
398         else
399         {
400             if (notIn)
401             {
402                 if (ExcludePrograms.Count == 0)
403                 {
404                     bldr.Append(" AND " + exclusionField + " NOT IN ('" +
405                         ExcludePrograms[i] + "')");
406                 } else if (ExcludePrograms.Count == 1)
407                 {
408                     bldr.Append(" AND " + exclusionField + " NOT IN ('" +
409                         ExcludePrograms[i] + "')");
410                 }
411                 else
412                 {
413                     bldr.Append(" AND " + exclusionField + " NOT IN ('" +
414                         ExcludePrograms[i] + "')");
415                 }
416             } else
417             {
418                 if (ExcludePrograms.Count == 0)
419                 {
420                     bldr.Append(" AND " + exclusionField + " IN ('" +
421                         ExcludePrograms[i] + "')");
422                 }
423                 else if (ExcludePrograms.Count == 1)
424                 {
425                     bldr.Append(" AND " + exclusionField + " IN ('" +
426                         ExcludePrograms[i] + "')");
427                 }
428                 else
429                 {
430                     bldr.Append(" AND " + exclusionField + " IN ('" +
431                         ExcludePrograms[i] + "')");
432                 }
433             }
434         }
435     }
436
437     return bldr.ToString();
438 } else
439 {
440     throw new Exception("No items in Exclusion Criteria");
441 }
442 }
443
444 public void LoadFile()
445 {
446     XElement info = XElement.Load("Requirements.xml");
447     Rules = from el in info.Elements()
448            select el;
449 }
450
451
452
453 public void LoadRule(string RuleName)
454 {
455     currentRule = from el in Rules.Elements()
456                  where el.Name == "Name" && el.Value == RuleName

```



```

457         select el.Parent;
458
459     ProcessRule();
460
461     CurrentRuleTitle = RuleName;
462
463
464 }
465
466 //Process current rule.
467 //This sets all logic for building the SQL Query
468 public void ProcessRule()
469 {
470
471     foreach (var item in currentRule.Elements("Billable"))
472     {
473         BillableCode = item.Value;
474     }
475
476     Componets.Clear();
477     ProgramExclude.Clear();
478     FSExclude.Clear();
479     ServiceExclude.Clear();
480     ProgramInclude.Clear();
481     FSInclude.Clear();
482     OverrideStaff = false;
483     OverrideStaffID = "";
484
485     foreach (var item in currentRule.Elements("Componets").Elements("Componet"))
486     {
487         Componets.Add(item.Value);
488     }
489
490     foreach (var item in
491 currentRule.Elements("ExcludeServices").Elements("ExcludeService"))
492     {
493         ServiceExclude.Add(item.Value);
494     }
495
496     foreach (var item in
497 currentRule.Elements("ExcludePrograms").Elements("ExcludeProgram"))
498     {
499         ProgramExclude.Add(item.Value);
500     }
501
502     foreach (var item in
503 currentRule.Elements("IncludePrograms").Elements("IncludeProgram"))
504     {
505         ProgramInclude.Add(item.Value);
506     }
507
508     foreach (var item in
509 currentRule.Elements("IncludeFSs").Elements("IncludeFS"))
510     {
511         FSInclude.Add(item.Value);
512     }
513
514     foreach (var item in currentRule.Elements("Rules"))
515     {
516         if (item.Element("FTF").Name == "FTF")
517         {
518             ReqFTF = int.Parse(item.Element("FTF").Value);
519         }
520
521         if (item.Element("Total").Name == "Total")
522         {
523             ReqTot = int.Parse(item.Element("Total").Value);
524         }
525     }
526 }

```

```

522         if (item.Element("OverrideStaff") != null &&
523             item.Element("OverrideStaff").Name == "OverrideStaff")
524         {
525             OverrideStaff = true;
526             OverrideStaffID = item.Element("OverrideStaff").Value;
527         }
528
529         if (item.Element("ServiceType") != null &&
530             item.Element("ServiceType").Name == "ServiceType")
531         {
532             currentServiceType =
533                 (ServiceRecord.ServiceType)int.Parse(item.Element("ServiceType").Value);
534         } else
535         {
536             currentServiceType = ServiceRecord.ServiceType.Other;
537         }
538
539         // System.Windows.Forms.MessageBox.Show("Billable- " + BillableCode + "
540         // FTF-" + ReqFTF.ToString() + " TOT-" + ReqTot.ToString());
541
542         //if (((XElement)currentRule.Elements("ExcludePrograms"))))
543     }
544
545     public void ProcessServiceExportFile()
546     {
547         FileHelperEngine<ServiceRecord> engine = new
548             FileHelperEngine<ServiceRecord>();
549         var pathWithEnv = @"%USERPROFILE%\Desktop\OutputServiceFile.txt";
550         var filePath = Environment.ExpandEnvironmentVariables(pathWithEnv);
551         engine.WriteFile(filePath, this.ServList);
552
553         System.Windows.Forms.MessageBox.Show("File Processed and sent to Desktop.");
554     }
555
556
557
558     [DelimitedRecord("\t")]
559     public class ServiceRecord
560     {
561
562         private ServiceRecord()
563         {
564             this.SPI = "100";
565         }
566
567         public enum ServiceType { ACT, CaseMgmt, StaffOnly, IDD, Other }
568
569         public ServiceRecord(DateTime SerDate, int ep, string SerCode, string
570             ProgramCode, string staffid, string clid, ServiceType srvType =
571             ServiceType.Other)
572         {
573             this.DOS = SerDate;
574             this.Service = SerCode;
575             this.Program = ProgramCode;
576             this.Pracitioner = staffid;
577             this.ClientID = clid;
578             this.Episode = ep;
579             if (srvType == ServiceType.ACT || srvType == ServiceType.CaseMgmt)
580             {
581                 this.SPI = "100";
582                 this.Location = "1";
583             }
584         }
585     }

```

```

583         if (srvType == ServiceType.IDD)
584         {
585             this.SPI = "200";
586             this.Location = "10";
587         }
588     }
589
590     [FieldOrder(1)]
591     [FieldConverter(ConverterKind.Date, "MMddyyyy")]
592     public DateTime DOS;
593     [FieldOrder(2)]
594     public string Service { get; set; }
595     [FieldOrder(3)]
596     public string Program { get; set; }
597     [FieldOrder(4)]
598     public string Pracitioner { get; set; }
599     [FieldOrder(5)]
600     public string ClientID { get; set; }
601     [FieldOrder(6)]
602     public int? Episode { get; set; }
603     [FieldOrder(7)]
604     public string GroupNumber { get; set; }
605     [FieldOrder(8)]
606     public int? NumberOfClients { get; set; }
607     [FieldOrder(9)]
608     public int? Duration { get; set; }
609     [FieldOrder(10)]
610     public string Location { get; set; }
611     [FieldOrder(11)]
612     public int? Units { get; set; }
613     [FieldOrder(12)]
614     public double? Cost { get; set; }
615     [FieldOrder(13)]
616     public string isBillable { get; set; }
617     [FieldOrder(14)]
618     public string costaff1 { get; set; }
619     [FieldOrder(15)]
620     public string costaff2 { get; set; }
621     [FieldOrder(16)]
622     public string refferStaff { get; set; }
623     [FieldOrder(17)]
624     public string refferProvider { get; set; }
625     [FieldOrder(18)]
626     public string MedDiag1 { get; set; }
627     [FieldOrder(19)]
628     public string MedDiag2 { get; set; }
629
630     [FieldOrder(20)]
631     public string Modifiers { get; set; }
632
633     [FieldOrder(21)]
634     public string Service_Start_Time { get; set; }
635     [FieldOrder(22)]
636     public string Service_End_Time { get; set; }
637     [FieldOrder(23)]
638     public string Tooth_Number { get; set; }
639     [FieldOrder(24)]
640     public string Tooth_Status_Code { get; set; }
641     [FieldOrder(25)]
642     public string Tooth_Surface_Code { get; set; }
643     [FieldOrder(26)]
644     public string Oral_Cavity_Designation_Code { get; set; }
645     [FieldOrder(27)]
646     public string Prosthesis_Crown_Or_Inlay_Code { get; set; }
647     [FieldOrder(28)]
648     public string Prior_Placement_Date { get; set; }
649     [FieldOrder(29)]
650     public string Note_Reference_Text_837_Billing { get; set; }
651     [FieldOrder(30)]

```

```
652     public string Note_Reference_Code_837_Billing { get; set; }
653     [FieldOrder(31)]
654     public string Appointment_Date { get; set; }
655     [FieldOrder(32)]
656     public string Appointment_Start_Time { get; set; }
657     [FieldOrder(33)]
658     public string Appointment_End_Time { get; set; }
659     [FieldOrder(34)]
660     public string Appointment_Site { get; set; }
661     [FieldOrder(35)]
662     public string Appointment_Status { get; set; }
663     [FieldOrder(36)]
664     public string Appointment_Notes { get; set; }
665     [FieldOrder(37)]
666     public string Appointment_Confirmed { get; set; }
667     [FieldOrder(38)]
668     public string Confirmation_Notes { get; set; }
669     [FieldOrder(39)]
670     public string Missed_Visit { get; set; }
671     [FieldOrder(40)]
672     public string Missed_Visit_Service_Code { get; set; }
673     [FieldOrder(41)]
674     public string Missed_Visit_Notes { get; set; }
675     [FieldOrder(42)]
676     public string Medical_Diagnosis_3 { get; set; }
677     [FieldOrder(43)]
678     public string Medical_Diagnosis_4 { get; set; }
679     [FieldOrder(44)]
680     public string Discipline { get; set; }
681     [FieldOrder(45)]
682     public string Service_Status { get; set; }
683     [FieldOrder(46)]
684     public string Initial_Treatment_Date_2300DTP03 { get; set; }
685     [FieldOrder(47)]
686     public string Date_Of_Accident_2300DTP03 { get; set; }
687     [FieldOrder(48)]
688     public string Form_Locator_10_Auto_Accident_Place { get; set; }
689     [FieldOrder(49)]
690     public string Related_Causes_Information_2300CLM11 { get; set; }
691     [FieldOrder(50)]
692     public string First_Billing_Date { get; set; }
693     [FieldOrder(51)]
694     public string Additional_Service_Information { get; set; }
695     [FieldOrder(52)]
696     public string Psychotherapy_AddOn_Duration { get; set; }
697     [FieldOrder(53)]
698     public string AddOn_Notes { get; set; }
699     [FieldOrder(54)]
700     public string ICD10_Diagnosis_Code_1 { get; set; }
701     [FieldOrder(55)]
702     public string ICD10_Diagnosis_Code_2 { get; set; }
703     [FieldOrder(56)]
704     public string ICD10_Diagnosis_Code_3 { get; set; }
705     [FieldOrder(57)]
706     public string ICD10_Diagnosis_Code_4 { get; set; }
707     [FieldOrder(58)]
708     public string Reserved_for_future_development { get; set; }
709     [FieldOrder(59)]
710     public string Reserved_for_future_development1 { get; set; }
711     [FieldOrder(60)]
712     public string Reserved_for_future_development2 { get; set; }
713     [FieldOrder(61)]
714     public string Reserved_for_future_development3 { get; set; }
715     [FieldOrder(62)]
716     public string Reserved_for_future_development4 { get; set; }
717     [FieldOrder(63)]
718     public string Reserved_for_future_development5 { get; set; }
719     [FieldOrder(64)]
720     public string Reserved_for_future_development6 { get; set; }
```

```
721     [FieldOrder(65)]
722     public string Reserved_for_future_development7 { get; set; }
723     [FieldOrder(66)]
724     public string Reserved_for_future_development8 { get; set; }
725     [FieldOrder(67)]
726     public string Reserved_for_future_development9 { get; set; }
727     [FieldOrder(68)]
728     public string Reserved_for_future_development10 { get; set; }
729     [FieldOrder(69)]
730     public string Reserved_for_future_development11 { get; set; }
731     [FieldOrder(70)]
732     public string Reserved_for_future_development12 { get; set; }
733     [FieldOrder(71)]
734     public string Reserved_for_future_development13 { get; set; }
735     [FieldOrder(72)]
736     public string Reserved_for_future_development14 { get; set; }
737     [FieldOrder(73)]
738     public string Reserved_for_future_development15 { get; set; }
739     [FieldOrder(74)]
740     public string Reserved_for_future_development16 { get; set; }
741     [FieldOrder(75)]
742     public string Reserved_for_future_development17 { get; set; }
743     [FieldOrder(76)]
744     public string Reserved_for_future_development18 { get; set; }
745     [FieldOrder(77)]
746     public string Reserved_for_future_development19 { get; set; }
747     [FieldOrder(78)]
748     public string Reserved_for_future_development20 { get; set; }
749     [FieldOrder(79)]
750     public string Reserved_for_future_development21 { get; set; }
751     [FieldOrder(80)]
752     public string Reserved_for_future_development22 { get; set; }
753     [FieldOrder(81)]
754     public string Reserved_for_future_development23 { get; set; }
755     [FieldOrder(82)]
756     public string Reserved_for_future_development24 { get; set; }
757     [FieldOrder(83)]
758     public string Reserved_for_future_development25 { get; set; }
759     [FieldOrder(84)]
760     public string Reserved_for_future_development26 { get; set; }
761     [FieldOrder(85)]
762     public string Reserved_for_future_development27 { get; set; }
763     [FieldOrder(86)]
764     public string Reserved_for_future_development28 { get; set; }
765     [FieldOrder(87)]
766     public string Reserved_for_future_development29 { get; set; }
767     [FieldOrder(88)]
768     public string Reserved_for_future_development30 { get; set; }
769     [FieldOrder(89)]
770     public string Reserved_for_future_development31 { get; set; }
771     [FieldOrder(90)]
772     public string Reserved_for_future_development32 { get; set; }
773     [FieldOrder(91)]
774     public string Reserved_for_future_development33 { get; set; }
775     [FieldOrder(92)]
776     public string Reserved_for_future_development34 { get; set; }
777     [FieldOrder(93)]
778     public string Reserved_for_future_development35 { get; set; }
779     [FieldOrder(94)]
780     public string Reserved_for_future_development36 { get; set; }
781     [FieldOrder(95)]
782     public string Reserved_for_future_development37 { get; set; }
783     [FieldOrder(96)]
784     public string Reserved_for_future_development38 { get; set; }
785     [FieldOrder(97)]
786     public string Reserved_for_future_development39 { get; set; }
787     [FieldOrder(98)]
788     public string Reserved_for_future_development40 { get; set; }
789     [FieldOrder(99)]
```

```
790     public string Reserved_for_future_development41 { get; set; }
791     [FieldOrder(100)]
792     public string SS_Treatment_Free_Text_10_1 { get; set; }
793     [FieldOrder(101)]
794     public string SS_Treatment_Free_Text_10_2 { get; set; }
795     [FieldOrder(102)]
796     public string SS_Treatment_Free_Text_10_3 { get; set; }
797     [FieldOrder(103)]
798     public string SS_Treatment_Free_Text_10_4 { get; set; }
799     [FieldOrder(104)]
800     public string SS_Treatment_Free_Text_10_5 { get; set; }
801     [FieldOrder(105)]
802     public string SS_Treatment_Free_Text_40_1 { get; set; }
803     [FieldOrder(106)]
804     public string SS_Treatment_Free_Text_40_2 { get; set; }
805     [FieldOrder(107)]
806     public string SS_Treatment_Free_Text_40_3 { get; set; }
807     [FieldOrder(108)]
808     public string SS_Treatment_Free_Text_40_4 { get; set; }
809     [FieldOrder(109)]
810     public string SS_Treatment_Free_Text_40_5 { get; set; }
811     [FieldOrder(110)]
812     public string SS_Treatment_Free_Text_80_1 { get; set; }
813     [FieldOrder(111)]
814     public string SS_Treatment_Free_Text_80_2 { get; set; }
815     [FieldOrder(112)]
816     public string SS_Treatment_Free_Text_80_3 { get; set; }
817     [FieldOrder(113)]
818     public string SS_Treatment_Free_Text_80_4 { get; set; }
819     [FieldOrder(114)]
820     public string SS_Treatment_Free_Text_80_5 { get; set; }
821     [FieldOrder(115)]
822     public string SS_Treatment_Scrolling_Free_Text_1 { get; set; }
823     [FieldOrder(116)]
824     public string SS_Treatment_Scrolling_Free_Text_2 { get; set; }
825     [FieldOrder(117)]
826     public string SS_Treatment_Scrolling_Free_Text_3 { get; set; }
827     [FieldOrder(118)]
828     public string SS_Treatment_Scrolling_Free_Text_4 { get; set; }
829     [FieldOrder(119)]
830     public string SS_Treatment_Scrolling_Free_Text_5 { get; set; }
831     [FieldOrder(120)]
832     public string SPI { get; set; }
833     [FieldOrder(121)]
834     public string SS_Treatment_Dictionary_2 { get; set; }
835     [FieldOrder(122)]
836     public string SS_Treatment_Dictionary_3 { get; set; }
837     [FieldOrder(123)]
838     public string SS_Treatment_Dictionary_4 { get; set; }
839     [FieldOrder(124)]
840     public string SS_Treatment_Dictionary_5 { get; set; }
841     [FieldOrder(125)]
842     public string SS_Treatment_Dictionary_6 { get; set; }
843     [FieldOrder(126)]
844     public string SS_Treatment_Dictionary_7 { get; set; }
845     [FieldOrder(127)]
846     public string SS_Treatment_Dictionary_8 { get; set; }
847     [FieldOrder(128)]
848     public string SS_Treatment_Dictionary_9 { get; set; }
849     [FieldOrder(129)]
850     public string SS_Treatment_ICD9_Code_1 { get; set; }
851     [FieldOrder(130)]
852     public string SS_Treatment_ICD9_Code_2 { get; set; }
853     [FieldOrder(131)]
854     public string SS_Treatment_ICD9_Code_3 { get; set; }
855     [FieldOrder(132)]
856     public string SS_Treatment_ICD9_Code_4 { get; set; }
857     [FieldOrder(133)]
858     public string SS_Treatment_ICD9_Code_5 { get; set; }
```

```
859     [FieldOrder(134)]
860     public string SS_Treatment_Date_1 { get; set; }
861     [FieldOrder(135)]
862     public string SS_Treatment_Date_2 { get; set; }
863     [FieldOrder(136)]
864     public string SS_Treatment_Date_3 { get; set; }
865     [FieldOrder(137)]
866     public string SS_Treatment_Date_4 { get; set; }
867     [FieldOrder(138)]
868     public string SS_Treatment_Date_5 { get; set; }
869     [FieldOrder(139)]
870     public string SS_Treatment_Practitioner_1 { get; set; }
871     [FieldOrder(140)]
872     public string SS_Treatment_Practitioner_2 { get; set; }
873     [FieldOrder(141)]
874     public string SS_Treatment_Practitioner_3 { get; set; }
875     [FieldOrder(142)]
876     public string SS_Treatment_Practitioner_4 { get; set; }
877     [FieldOrder(143)]
878     public string SS_Treatment_Practitioner_5 { get; set; }
879     [FieldOrder(144)]
880     public string SS_Treatment_Guarantor_1 { get; set; }
881     [FieldOrder(145)]
882     public string SS_Treatment_Guarantor_2 { get; set; }
883     [FieldOrder(146)]
884     public string SS_Treatment_Guarantor_3 { get; set; }
885     [FieldOrder(147)]
886     public string SS_Treatment_Guarantor_4 { get; set; }
887     [FieldOrder(148)]
888     public string SS_Treatment_Guarantor_5 { get; set; }
889     [FieldOrder(149)]
890     public string SS_Treatment_DSM_Code_1 { get; set; }
891     [FieldOrder(150)]
892     public string SS_Treatment_DSM_Code_2 { get; set; }
893     [FieldOrder(151)]
894     public string SS_Treatment_DSM_Code_3 { get; set; }
895     [FieldOrder(152)]
896     public string SS_Treatment_DSM_Code_4 { get; set; }
897     [FieldOrder(153)]
898     public string SS_Treatment_DSM_Code_5 { get; set; }
899     [FieldOrder(154)]
900     public string SS_Treatment_Multiple_Select_Dictionary_1 { get; set; }
901     [FieldOrder(155)]
902     public string SS_Treatment_Multiple_Select_Dictionary_2 { get; set; }
903     [FieldOrder(156)]
904     public string Guarantor_1 { get; set; }
905     [FieldOrder(157)]
906     public string Amount1 { get; set; }
907     [FieldOrder(158)]
908     public string Guarantor_2 { get; set; }
909     [FieldOrder(159)]
910     public string Amount2 { get; set; }
911     [FieldOrder(160)]
912     public string Guarantor_3 { get; set; }
913     [FieldOrder(161)]
914     public string Amount3 { get; set; }
915     [FieldOrder(162)]
916     public string Guarantor_4 { get; set; }
917     [FieldOrder(163)]
918     public string Amount4 { get; set; }
919     [FieldOrder(164)]
920     public string Guarantor_5 { get; set; }
921     [FieldOrder(165)]
922     public string Amount5 { get; set; }
923     [FieldOrder(166)]
924     public string Guarantor_6 { get; set; }
925     [FieldOrder(167)]
926     public string Amount6 { get; set; }
927     [FieldOrder(168)]
```

```

928     public string Guarantor_7 { get; set; }
929     [FieldOrder(169)]
930     public string Amount7 { get; set; }
931     [FieldOrder(170)]
932     public string Guarantor_8 { get; set; }
933     [FieldOrder(171)]
934     public string Amount8 { get; set; }
935     [FieldOrder(172)]
936     public string Guarantor_9 { get; set; }
937     [FieldOrder(173)]
938     public string Amount9 { get; set; }
939     [FieldOrder(174)]
940     public string Guarantor_10 { get; set; }
941     [FieldOrder(175)]
942     public string Amount10 { get; set; }
943     [FieldOrder(176)]
944     public string SS_Treatment_Integer_1 { get; set; }
945     [FieldOrder(177)]
946     public string SS_Treatment_Integer_2 { get; set; }
947     [FieldOrder(178)]
948     public string SS_Treatment_Integer_3 { get; set; }
949     [FieldOrder(179)]
950     public string SS_Treatment_Integer_4 { get; set; }
951     [FieldOrder(180)]
952     public string SS_Treatment_Integer_5 { get; set; }
953     [FieldOrder(181)]
954     public string SS_Treatment_Integer_6 { get; set; }
955     [FieldOrder(182)]
956     public string IncidentTo_Practitioner { get; set; }
957
958
959
960 }
961
962 enum month { January = 1, February, March, April, May, June, July, August,
September, October, November, December }

```

```

963
964
965
966
967 }
968

```



```
1  using FileHelpers;
2  using System;
3  using System.Collections.Generic;
4  using System.ComponentModel;
5  using System.Data;
6  using System.Drawing;
7  using System.Linq;
8  using System.Text;
9  using System.Threading.Tasks;
10 using System.Windows.Forms;
11
12 namespace Avatar_Solutions
13 {
14     public partial class LoginForm : Form
15     {
16         public LoginForm()
17         {
18             InitializeComponent();
19         }
20
21         private void Form1_Load(object sender, EventArgs e)
22         {
23
24             this.combo_systemCodes.Text = "LIVE";
25         }
26
27         private void button1_Click(object sender, EventArgs e)
28         {
29             AvatarConnection con = new AvatarConnection(combo_systemCodes.Text,
30                 txt_Username.Text, txt_pwd.Text);
31
32             if (con.opened)
33             {
34                 this.Hide();
35
36                 MainMenu frm = new MainMenu(con);
37                 frm.ShowDialog();
38
39                 if (frm.DialogResult == DialogResult.Retry)
40                 {
41                     this.Show();
42                     frm = null;
43                 } else
44                 {
45                     this.Close();
46                 }
47             }
48
49         }
50
51     }
52
53     private void combo_systemCodes_Validating(object sender, CancelEventArgs e)
54     {
55         if (combo_systemCodes.Text == "" || combo_systemCodes.Text == null)
56         {
57             errorProvider1.SetError(combo_systemCodes, "System Code Required");
58             combo_systemCodes.Focus();
59             e.Cancel = true;
60         } else
61         {
62             errorProvider1.SetError(combo_systemCodes, "");
63         }
64     }
65
66     private void txt_Username_Validating(object sender, CancelEventArgs e)
67     {
68         if (txt_Username.Text == "" || txt_Username.Text == null)
```

```

69         {
70             errorHandler1.SetError(txt_Username, "Username Required");
71             ((Control)sender).Focus();
72             e.Cancel = true;
73         }
74         else
75         {
76             errorHandler1.SetError(txt_Username, "");
77         }
78     }
79
80     private void txt_pwd_Validating(object sender, CancelEventArgs e)
81     {
82         if (txt_pwd.Text == "" || txt_pwd.Text == null)
83         {
84             errorHandler1.SetError(txt_pwd, "Password Required");
85             ((Control)sender).Focus();
86             e.Cancel = true;
87         }
88         else
89         {
90             errorHandler1.SetError(txt_pwd, "");
91         }
92     }
93
94     private void button2_Click(object sender, EventArgs e)
95     {
96         /*
97          *
98          * GL.GL sys = new GL.GL("wallace4");
99          * sys.GetPrograms();
100          * MessageBox.Show(sys.Rus.Count.ToString());
101          */
102     }
103
104     private void button3_Click(object sender, EventArgs e)
105     {
106         /* AvatarConnection con = new AvatarConnection("LIVE", "wallace4", "Dw832806");
107          * con.conn.Open();
108          * con.GetStaff();
109          * // MessageBox.Show(con.staffList.Count.ToString());
110          *
111          *
112          *
113          *
114          * //MessageBox.Show(con.GetTotalPracNumber().ToString());
115          *
116          * con.conn.Close();*/
117     }
118
119     private void backgroundWorker1_DoWork(object sender, DoWorkEventArgs e)
120     {
121         //con.GetPracNumbersRecords();
122
123         e.Result = true;
124     }
125 }
126
127

```

```

1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9  using System.Windows.Forms;
10 using System.Data.Odbc;
11 using System.Net.Mail;
12 namespace Avatar_Solutions
13 {
14     public partial class Processor : Form
15     {
16         public Processor()
17         {
18             InitializeComponent();
19         }
20
21         private AvatarConnection avConn;
22         private OdbcConnection conn;
23         private OdbcCommand cmd;
24         private OdbcDataAdapter DA;
25         DataTable dtbl;
26
27         private List<Report> rptList;
28         private List<ServiceRecord> srvRecord;
29
30         int TotalCreated = 0;
31         int TotalError = 0;
32         int TotalAlreadyCreated = 0;
33
34         public Processor(AvatarConnection con)
35         {
36             InitializeComponent();
37             avConn = con;
38             dtbl = new DataTable();
39             conn = avConn.conn;
40
41             srvRecord = new List<ServiceRecord>();
42             rptList = new List<Report>();
43
44             DA = new OdbcDataAdapter();
45
46             dataGridView1.DataSource = dtbl;
47
48
49         }
50
51         private void GetData()
52         {
53             DateTime FromDate = DateTime.Parse(((int)comboBox1.SelectedItem).ToString()
54             + "/01/" + comboBox2.Text);
55             DateTime ThruDate = FromDate.AddMonths(1).AddDays(-1);
56
57             string sendFrom = FromDate.ToString("yyyy-MM-dd");
58             string sendThru = ThruDate.ToString("yyyy-MM-dd");
59             //string sendFrom = From.Value.ToString("yyyy-MM-dd");
60             //string sendThru = ThruDate.Value.ToString("yyyy-MM-dd");
61             //MessageBox.Show("From: " + sendFrom + Environment.NewLine + "Thru: " +
62             sendThru);
63             DA.SelectCommand = avConn.SQLCaseMgmt(sendFrom, sendThru,
64             avConn.Componets, avConn.ProgramExclude);
65             DA.SelectCommand.Connection = conn;
66             DA.Fill(dtbl);
67         }
68     }
69 }

```

```

67     private void ProcessData()
68     {
69         int ReqFTF = avConn.ReqFTF;
70         int ReqTot = avConn.ReqTot;
71         string billableCode = avConn.BillableCode;
72
73         //This Code Generates Billable Charges
74         int NonFTF = 0;
75         int FTF = 0;
76         int Total = 0;
77         bool FoundBillable = false;
78         bool started = true;
79         string PreviousClient = "";
80
81
82         int Episode = 0;
83         int PrvEpisode = 0;
84         string currentClient = "";
85         string Service = "";
86         int Tot = 0;
87         string Program = "";
88         string StaffID = "";
89         DateTime DOS = DateTime.Now;
90
91         string PrvService = "";
92         //int Tot = 0;
93         string PrvProgram = "";
94         string PrvStaffID = "";
95         DateTime PrvDOS = DateTime.Now;
96
97         int i = 0;
98         foreach(DataRow row in dtbl.Rows)
99         {
100             i++;
101
102             currentClient = (string)row["PATID"];
103
104
105             if (i == dtbl.Rows.Count || started)
106             {
107                 PreviousClient = currentClient;
108                 started = false;
109
110                 Service = (string)row["Service"];
111                 Tot = (int)row["Tot"];
112                 Program = (string)row["Program"];
113                 StaffID = (string)row["STAFFID"];
114                 Episode = (int)row["EPISODE"];
115                 if (row["DOS"] != DBNull.Value)
116                 {
117                     DOS = Convert.ToDateTime(row["DOS"]);
118                 }
119                 else
120                 {
121                     throw new Exception("No DateTime");
122                 }
123             }
124
125             /*         if (avConn.CurrentRuleTitle == "ACT")
126                 {
127                     started = false;
128                 }*/
129
130             if (i == dtbl.Rows.Count)
131             {
132                 currentClient = "";
133                 FoundBillable = false;
134                 FTF = 0;
135                 Total = 0;

```

```

136         NonFTF = 0;
137
138         if (Service == billableCode)
139         {
140             FoundBillable = true;
141
142         }
143         else if (Service[Service.Length - 1] == 'F')
144         {
145             FTF += Tot;
146             Total += Tot;
147         }
148         else
149         {
150             NonFTF += Tot;
151             Total += Tot;
152         }
153
154         PrvDOS = DOS;
155         PrvProgram = Program;
156         PrvService = Service;
157         PrvStaffID = StaffID;
158         PrvEpisode = (int)row["EPISODE"];
159
160     }
161
162
163     if (PreviousClient != currentClient)
164     {
165
166         //If override staff logic then change staff id HERE since always
167         //hit.
168         if (avConn.OverrideStaff)
169         {
170             PrvStaffID = avConn.OverrideStaffID;
171         }
172         //end staff override logic
173
174         if (!FoundBillable && FTF >= ReqFTF && Total >= ReqTot)
175         {
176             //GenerateCharge
177             rptList.Add(new Report(DOS, PreviousClient, FTF, Total, true,
178                                     "Charge Created"));
179             avConn.ServList.Add(new ServiceRecord(PrvDOS, PrvEpisode,
180                                     billableCode, PrvProgram, PrvStaffID, PreviousClient,
181                                     avConn.currentServiceType));
182             TotalCreated += 1;
183         }
184         else if (FoundBillable)
185         {
186             //WasAlreadyCharges
187             rptList.Add(new Report(DOS, PreviousClient, FTF, Total, false,
188                                     "Already Created"));
189             TotalAlreadyCreated += 1;
190         }
191         else if (FTF >= ReqFTF && Total < ReqTot)
192         {
193             //Not Enough Services
194             rptList.Add(new Report(DOS, PreviousClient, FTF, Total, false,
195                                     "Not Enough Services"));
196             TotalError += 1;
197         }
198         else if (FTF <= ReqFTF)
199         {
200             //Not Enough FTF
201             rptList.Add(new Report(DOS, PreviousClient, FTF, Total, false,
202                                     "Not Enough FTF"));
203             TotalError += 1;
204         }
205         else

```

```

198         {
199             //Other Error
200             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)

```

```
264     {
265         dtbl.Clear();
266         this.GetData();
267         comboBox1.SelectedIndex += 1;
268     }
269
270 private void button2_Click(object sender, EventArgs e)
271 {
272     this.ProcessData();
273
274     Report_Viewer frm = new
275     Report_Viewer(rptList, TotalCreated, TotalAlreadyCreated, TotalError, avConn.Curr
276     entRuleTitle, From.Value, ThruDate.Value);
277     frm.ShowDialog();
278
279     // SendMessage();
280 }
281
282 }
283
284 }
```

```

1  <?xml version="1.0" encoding="utf-8" ?>
2  <PreprocessorRules>
3    <PreprocessorRule>
4
5      <Name>Case Mgmt (Reg)</Name>
6      <Billable>T2023</Billable>
7      <Componets>
8        <Componet FTF="N">770</Componet>
9        <Componet FTF="Y">770F</Componet>
10       <Componet FTF="N">771</Componet>
11       <Componet FTF="Y">771F</Componet>
12       <Componet FTF="N">780</Componet>
13       <Componet FTF="Y">780F</Componet>
14       <Componet FTF="N">790</Componet>
15       <Componet FTF="Y">790F</Componet>
16     </Componets>
17     <ExcludePrograms>
18       <ExcludeProgram>9105</ExcludeProgram>
19       <ExcludeProgram>9115</ExcludeProgram>
20       <ExcludeProgram>9350</ExcludeProgram>
21       <ExcludeProgram>9712</ExcludeProgram>
22       <ExcludeProgram>9963</ExcludeProgram>
23       <ExcludeProgram>9980</ExcludeProgram>
24       <ExcludeProgram>9385</ExcludeProgram>
25     </ExcludePrograms>
26     <Rules>
27       <FTF>2</FTF>
28       <Total>4</Total>
29       <ServiceType>1</ServiceType>
30     </Rules>
31
32   </PreprocessorRule>
33
34   <PreprocessorRule>
35
36     <Name>Case Mgmt (Complex)</Name>
37     <Billable>T2023TG</Billable>
38     <Componets>
39       <Componet FTF="N">777</Componet>
40       <Componet FTF="Y">777F</Componet>
41       <Componet FTF="N">778</Componet>
42       <Componet FTF="Y">778F</Componet>
43       <Componet FTF="N">781</Componet>
44       <Componet FTF="Y">781F</Componet>
45       <Componet FTF="N">791</Componet>
46       <Componet FTF="Y">791F</Componet>
47     </Componets>
48     <ExcludePrograms>
49       <ExcludeProgram>9105</ExcludeProgram>
50       <ExcludeProgram>9115</ExcludeProgram>
51       <ExcludeProgram>9350</ExcludeProgram>
52       <ExcludeProgram>9712</ExcludeProgram>
53       <ExcludeProgram>9963</ExcludeProgram>
54       <ExcludeProgram>9980</ExcludeProgram>
55       <ExcludeProgram>9385</ExcludeProgram>
56     </ExcludePrograms>
57     <Rules>
58       <FTF>3</FTF>
59       <Total>5</Total>
60       <ServiceType>1</ServiceType>
61     </Rules>
62
63   </PreprocessorRule>
64
65   <PreprocessorRule>
66
67     <Name>ACT</Name>

```



```

68     <Billable>H0040</Billable>
69     <Componets>
70         <Componet FTF="N">863</Componet>
71         <Componet FTF="Y">863F</Componet>
72     </Componets>
73
74     <IncludePrograms>
75         <IncludeProgram>9525</IncludeProgram>
76     </IncludePrograms>
77
78     <Rules>
79         <FTF>1</FTF>
80         <Total>10</Total>
81         <ServiceType>0</ServiceType>
82     </Rules>
83
84 </PreprocessorRule>
85
86
87
88 <PreprocessorRule>
89     <Name>Waiver (Case Mgmt)</Name>
90     <Billable>T2022</Billable>
91     <Componets>
92         <Componet FTF="N">770</Componet>
93         <Componet FTF="Y">770F</Componet>
94         <Componet FTF="N">771</Componet>
95         <Componet FTF="Y">771F</Componet>
96         <Componet FTF="N">780</Componet>
97         <Componet FTF="Y">780F</Componet>
98         <Componet FTF="N">790</Componet>
99         <Componet FTF="Y">790F</Componet>
100    </Componets>
101    <IncludeFSs>
102        <IncludeFS>1121</IncludeFS>
103        <IncludeFS>1126</IncludeFS>
104        <IncludeFS>1145</IncludeFS>
105        <IncludeFS>1146</IncludeFS>
106    </IncludeFSs>
107    <Rules>
108        <FTF>0</FTF>
109        <Total>1</Total>
110        <OverrideStaff>9995</OverrideStaff>
111        <ServiceType>3</ServiceType>
112    </Rules>
113 </PreprocessorRule>
114
115
116 <PreprocessorRule>
117     <Name>Waiver (Fin. Mgmt)</Name>
118     <Billable>T2040</Billable>
119
120     <IncludeFSs>
121         <IncludeFS>1121</IncludeFS>
122         <IncludeFS>1146</IncludeFS>
123     </IncludeFSs>
124
125     <Rules>
126         <FTF>0</FTF>
127         <Total>1</Total>
128         <OverrideStaff>9995</OverrideStaff>
129         <ServiceType>3</ServiceType>
130     </Rules>
131 </PreprocessorRule>
132
133 <!--
134 <PreprocessorRule>

```

```
135     <Name>Pasarr (140)</Name>
136     <Billable>T2022</Billable>
137
138     <IncludeFSs>
139         <IncludeFS>1128</IncludeFS>
140     </IncludeFSs>
141
142     <IncludePrograms>
143         <IncludeProgram>2530</IncludeProgram>
144     </IncludePrograms>
145
146     <Rules>
147         <FTF>0</FTF>
148         <Total>1</Total>
149         <OverrideStaff>9995</OverrideStaff>
150         <ServiceType>3</ServiceType>
151     </Rules>
152 </PreprocessorRule>
153 -->
154 <PreprocessorRule>
155     <Name>IDD (140)</Name>
156     <Billable>T2022</Billable>
157
158     <IncludeFSs>
159         <IncludeFS>1071</IncludeFS>
160     </IncludeFSs>
161
162     <IncludePrograms>
163         <IncludeProgram>2530</IncludeProgram>
164         <IncludeProgram>2330</IncludeProgram>
165         <IncludeProgram>3330</IncludeProgram>
166         <IncludeProgram>4330</IncludeProgram>
167         <IncludeProgram>5330</IncludeProgram>
168         <IncludeProgram>6330</IncludeProgram>
169         <IncludeProgram>7330</IncludeProgram>
170         <IncludeProgram>8330</IncludeProgram>
171     </IncludePrograms>
172
173     <Rules>
174         <FTF>0</FTF>
175         <Total>1</Total>
176         <OverrideStaff>9995</OverrideStaff>
177         <ServiceType>3</ServiceType>
178     </Rules>
179 </PreprocessorRule>
180
181
182
183 </PreprocessorRules>
```