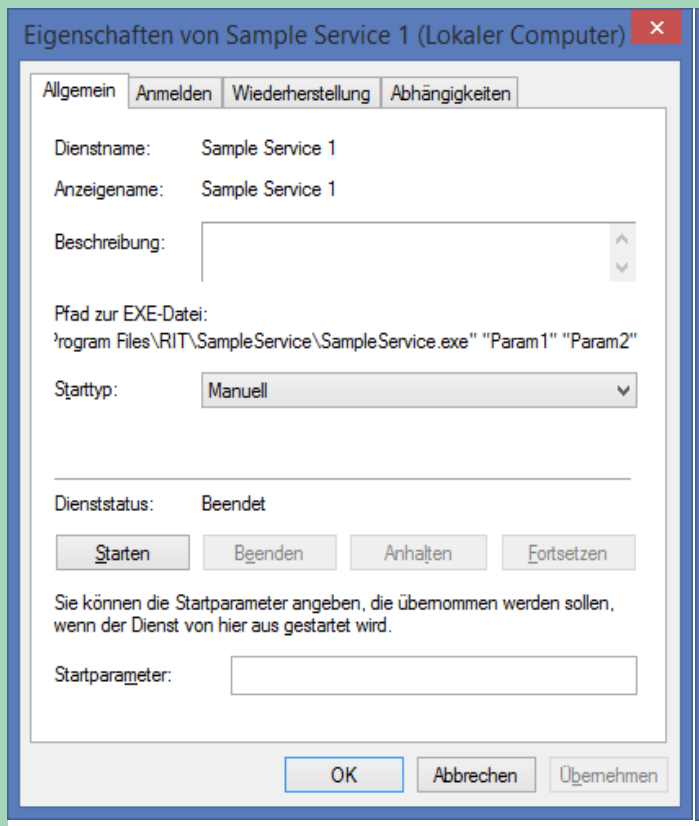


Abbildung 4: Die in der Registry ergänzten Parameter werden im Service Control GUI angezeigt

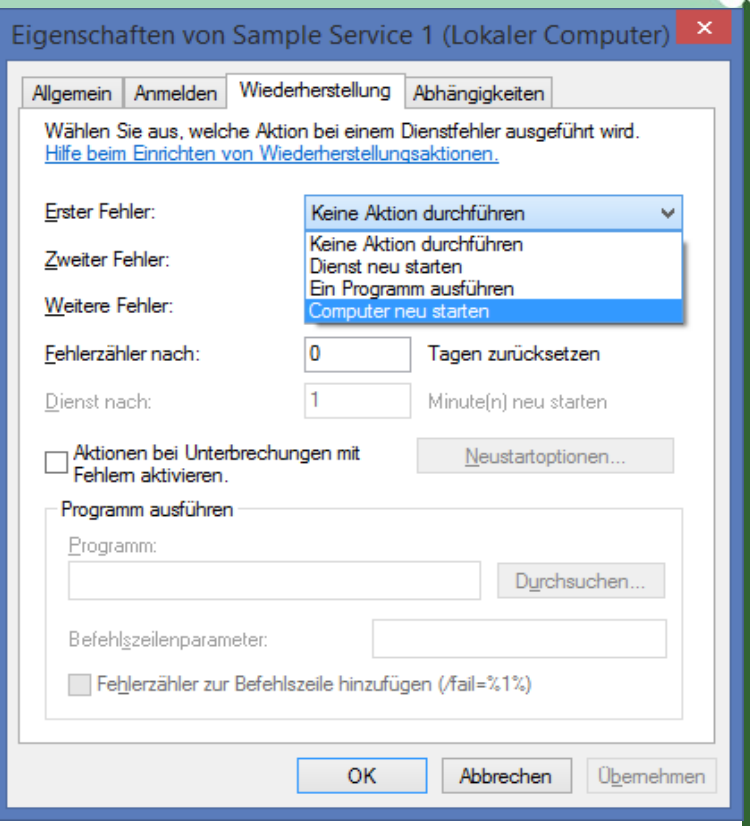


Abbildung 5: Service Wiederherstellungsmöglichkeiten wie sie das Betriebssystem bereitstellt

Also bleibt nichts anderes übrig, als direkt in der Registry den Wert zu ergänzen und somit den Parameter zu persistieren.

An jene Stelle, an der der physische Pfad zum Dateisystem eingetragen ist, kann man nun seine Start Parameter manuell ergänzen.

Achtung: Der Pfad, als auch jeder Parameter müssen unter Hochkomma „“ gesetzt werden. Jeder weitere Parameter durch ein Blank getrennt sein. Ansonsten kann das Service nicht mehr gestartet werden.

Damit die Parameter im Service auch ankommen, wird hier der Code abermals erweitert, in dem man `static void Main (string[] args)` erweitert.

Service Priority festlegen

Die Prozesse die im Service gestartet werden, können einer Prozess Priorität zugeordnet werden.

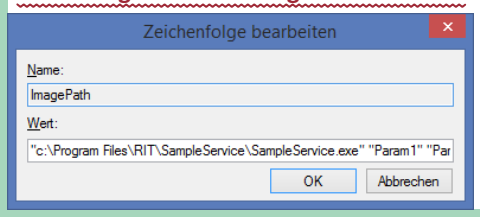
Service recovery

Treten während der Laufzeit des Service Probleme auf, bietet das Betriebssystem Wiederherstellungsmöglichkeiten an. Dazu ist im Code nichts zu ergänzen.

Fazit

Windows Services werden nicht aussterben, es werden immer Lösungen benötigt, die auf einem lokalen Environment oder nur im Intranet laufen werden. Beginnend von systemnahen Prozessen bis zu Importvorgängen. Vor allem dann, wenn man nicht in die Cloud gehen kann oder möchte, bleiben Services weiterhin sinnvoll. Anscheinend ist das der Hintergrund, warum nicht mehr in die Weiterentwicklung von Windows Services unternommen wird. Ein Windows Service implementiert man inzwischen sicherlich nicht mehr so häufig wie ein .net WebService oder eine asp.net WebSeite. Daher auch der Grund meiner Zusammenfassung von Tipps zu Windows Services, die mir während meiner Projekttätigkeit untergekommen sind.

Abbildung 3: Erweiterung um Parameter



Code Sample:

```
static void Main(string[] args)
{
  try
  {
    Console.WriteLine(string.Format("Argument items: {0}", args.Length));
    // Set priority
    System.Diagnostics.Process p = System.Diagnostics.Process.GetCurrentProcess();
    switch (Config.ProcessPriority.ToLower())
    {
       case "abovenormal":
         p.PriorityClass = System.Diagnostics.ProcessPriorityClass.AboveNormal;
         break;
       case "belownormal":
         p.PriorityClass = System.Diagnostics.ProcessPriorityClass.BelowNormal;
         break;
       case "high":
         p.PriorityClass = System.Diagnostics.ProcessPriorityClass.High;
         break;
       case "normal":
         p.PriorityClass = System.Diagnostics.ProcessPriorityClass.Normal;
         break;
       case "realtime":
         p.PriorityClass = System.Diagnostics.ProcessPriorityClass.RealTime;
         break;
       }
       Console.WriteLine(string.Format("KAV.IHE.Service: ProcessPriority: {0}",
p.BasePriority));
```